BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ ALANINDA AKADEMİK TARTIŞMALAR

Editör: Doç.Dr. Fatih YÜCALAR

yaz
yayınları

Bilgisayar Bilimleri ve Mühendisliği Alanında Akademik Tartışmalar

Editör

Doç.Dr. Fatih YÜCALAR

yaz yayınları

2026

yaz
yayınları

Bilgisayar Bilimleri ve Mühendisliği Alanında Akademik Tartışmalar

Editör: Doç.Dr. Fatih YÜCALAR

E_ISBN 978-625-8996-64-7

Haziran 2026 – Afyonkarahisar

Dizgi/Mizanpaj: YAZ Yayınları

Kapak Tasarım: YAZ Yayınları

YAZ Yayınları. Yayıncı Sertifika No: 73086

M.İhtisas OSB Mah. 4A Cad. No:3/3
İscehisar/AFYONKARAHİSAR

www.yazyayinlari.com

yazyayinlari@gmail.com

İÇİNDEKİLER

TIME OFFSET ESTIMATION IN UNMANNED AERIAL VEHICLE (UAV) SENSOR DATA USING DEEP LEARNING-BASED FEATURES

Özlem ÖRNEK[1]

Efnan ŞORA GÜNAL[2]

1. INTRODUCTION

Technological advancements have influenced the development of robotic systems, diversifying and expanding their application areas and tasks (Ryalat, Almtireen, Al-refai, ElMoaqet and Rawashdeh, 2025). Robotic systems utilize sensor systems for both situational awareness and environmental perception. Sensors can provide data of varying qualities regarding the state of the environment or the system. The acquired data is used for robotic systems to successfully execute their operational and task processes. Looking at the historical development of robotic systems, it can be summarized as human-operator managed systems, automatic systems operating under human supervision in a fixed location, unmanned autonomous systems operating in a fixed location, and unmanned fully autonomous systems. The goal is the creation of fully independent (autonomous) systems. In this context, one of the fundamental components of a robotic system is its ability to perceive its surroundings and localize itself within this environment.

[1] Res. Asst., Eskisehir Osmangazi University, Faculty of Engineering and Architecture, Dept. of Computer Engineering, ORCID: 0000-0002-8775-8695.

[2] Assoc. Prof. Dr., Eskisehir Osmangazi University, Faculty of Engineering and Architecture, Dept. of Computer Engineering, ORCID: 0000-0001-6236-174X.

For this purpose, many different methods are utilized in the literature (Alipourvarmazabadi and Ebner, 2026). These methods utilize data obtained from robotic system sensors. The types of data used (visual, radar, etc.) may vary depending on the method. Some methods collectively use the same or different types of data obtained from multiple sensors to achieve better performance (Ušinskis, Nowicki, Dzedzickis and Bučinskas, 2025). To achieve accurate results, data obtained from different sensors must be synchronized. There are various approaches in the literature to ensure the synchronization of sensors (Sivrikaya and Yener, 2004; Dammert, Thalmann, Monetti, Neuner, and Mandlburger, 2025). These approaches are generally aimed at ensuring that timestamps arrive simultaneously during the data collection phase. However, due to an unexpected error or malfunction in the system, even if the timestamps appear synchronized, the data content may not be collected simultaneously (Lyu, Liu, Qiao, Jiang and Wang, 2025). The error occurring in this situation can affect the systems where this data is used, negatively impacting the operational performance and accuracy of the robotic system.

In this study analyzes proposed approaches to time offset estimation under declining time offset to improve system performance and accuracy through time synchronization. The study proposes a time-series-focused approach using deep learning-based feature extraction to estimate the time offset between data obtained from two different sensors for a drone. To test the proposed method, data obtained from the sensor system on the drone and from the optical motion capture system located in the drone's environment during the drone's movement in a laboratory test environment are used.

2. PROPOSED METHOD

This section provides information regarding the dataset, preparation, method stages, and the application of the method for the proposed approach. The flowchart of the proposed method for ensuring the synchronization of time-series data to contribute to the accuracy or stability of positioning systems of robotic systems is given in Figure 1. The processes within the flow were implemented using the Python programming language.

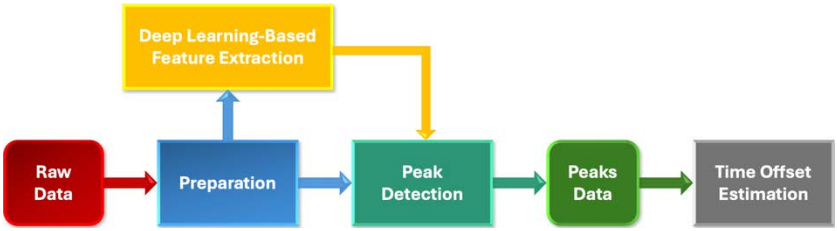


Figure 1. Flowchart of The Proposed Method

2.1. Dataset

The synchronization of information obtained during the operational processes of robotic systems is important for the decision stages in the systems where this data is used. In this context, data was collected to be used in the proposed method for controlling the synchronization of data obtained from robotic systems and estimating time offset. Figure 2 presents a view of the laboratory environment where the dataset was collected and the drone present in the environment.

Figure 2. Laboratory Environment

Datasets were collected using an experimental setup encompassing a drone and an optical motion tracking system (OptiTrack). OptiTrack sensors are a system used in motion tracking and positioning technologies, typically utilized for three-dimensional motion tracking. OptiTrack can provide real-time data with high accuracy and precision. Datasets were acquired under five different scenarios:

- 0 milliseconds (ms) delay, square tour at a single altitude
- 0 ms delay, square tour at two altitudes
- 5 to 0 ms declining delay over time, square tour at a single altitude
- 15 to 9 ms declining delay over time, square tour at a single altitude
- 30 to 19 declining ms delay over time, square tour at a single altitude

The delays are present in the OptiTrack data. Therefore, the OptiTrack data arrives later than the drone data by the amount of the delay value. Each dataset contains data coming from the drone (timestamp, acceleration, velocity, altitude, height, position, angular velocity, and quaternion) and data coming from the optical motion tracking system (timestamp and position).

Values are present in three axes (x, y, and z) for fields other than timestamps.

2.2. Preparation

The acceleration unit was used in time offset estimation. The purpose of this is to distinctly see the changes in the data resulting from the drone's movements. To achieve this, velocity data obtained from the drone and position data obtained from the optical motion tracking system were selected from the acquired datasets. These two selected raw data, along with timestamp information, are first transmitted to the preparation stage, and these processes are as follows:

- Analysis and Preprocessing
- Derivative Process (Unit conversion)
- Norm Process (Single-dimension Representation)

During the analysis phase, the data is checked for dimensions and for the presence of null or duplicate values. In the preprocessing stage, if null or repeating values are found during the analysis phase, they are removed. Subsequently, a beginning check, an interval check, and a time unit conversion (conversion to seconds) are performed to ensure the chronological ordering among data timestamps. The velocity and position data are in different units. They need to be converted to a common unit to be evaluated together. For this purpose, the first derivative of the velocity and the second derivative of the position data are taken, respectively, converting them into acceleration units. Derivative calculations were performed using the "gradient" function of the "NumPy" library in Python (Harris et al., 2020; NumPy Developers, 2025).

Thus, data obtained from two different sources are represented with the same unit. The data contain information across three axes (x, y, and z). To represent the information in

these axes in a single dimension, the Euclidean norm value of the acceleration data obtained from velocity and the acceleration data obtained from position is calculated. In the calculation, the "linalg.norm" function of the "NumPy" library was utilized (Harris et al., 2020). As a result of these operations, the acceleration norm value and timestamp values derived from velocity, as well as the acceleration norm value and timestamp values derived from position, are obtained. As an example, in the graph provided in Figure 3, the norm values obtained for the "0 ms delay, square tour at two altitudes" dataset are plotted overlaid on a single timeline after normalization between 0 and 1. Through this visualization, the characteristic peak points seen in the graph are expected to show similarity to one another.

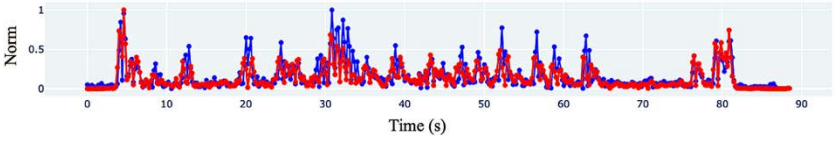


Figure 3. Comparison of Normalized Position-Acceleration Norm (Red) and Velocity-Acceleration Norm (Blue) Data

2.3. Peak Detection

Following data analysis and preprocessing stages, norm and timestamp data from two different sources are obtained. The peak detection stage aims to identify the peak points in the data for these two different norm datasets and to acquire information about the timestamps at which these peak points occur. Within this scope, two different approaches were used for peak detection. One involves using norm data directly in the peak detection method. In the second approach, norm data is used in deep learning feature extraction, and the obtained features (data) are used in the peak detection method.

In summary, during the peak detection phase, the norm transmitted from the previous stage or features derived from the norm, along with the timestamp data belonging to this data, are

used. The peak detection method provides the timestamp information as output of the peak points identified for the given input (data). At this stage, the "find_peaks" function given in the Python "SciPy" library is employed (Virtanen et al., 2020).

2.4. Deep Learning-Based Feature Extraction

In this phase, "untrained convolutional neural networks" are used to obtain time-varying characteristic features from time series data. For this purpose, a feature-extracting approach based on the "InceptionTime" model (Fawaz et al., 2020) with random weights was employed. "InceptionTime" is a highly powerful model for time series; by using filters of different sizes simultaneously, it captures both short- and long-term variations in the signal (Fawaz et al., 2020). There is no training phase in the created model. The model's classification layer (head) is not used; the "backbone" section is utilized. Thus, enriched vectors of local temporal patterns are obtained from the data. The flowchart of deep learning-based feature extraction and the details of the deep learning model's architecture are shown in Figures 4 and 5, respectively.

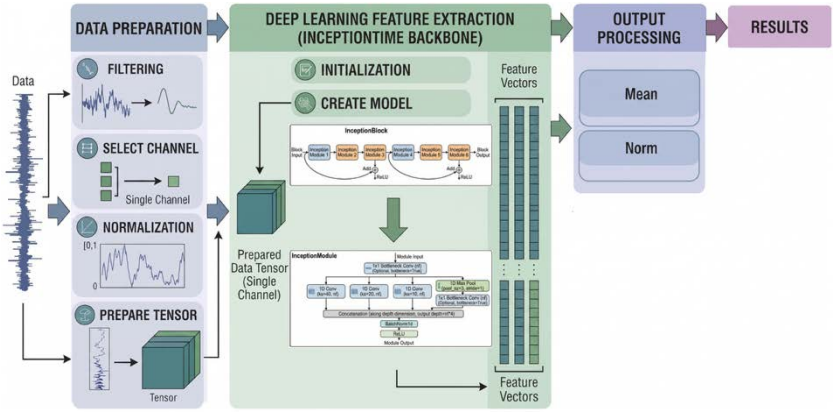


Figure 4. Flowchart of Deep Learning-based Feature Extraction

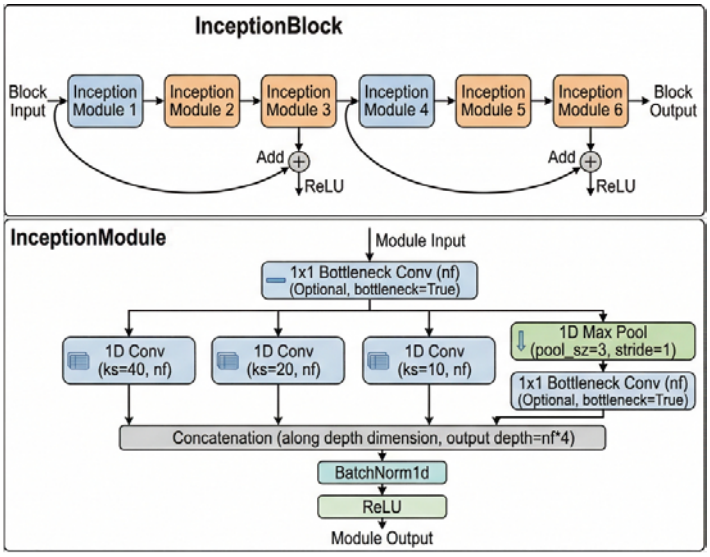


Figure 5. Architecture of the Deep Learning Model

Deep learning models in the literature are generally used after a training phase. However, they can also be used with random weights without training (Li, Xi, and Lin, 2024). In this study, the "InceptionTime" model is used for feature extraction with random weights without training. An architecture with random weights can be a powerful feature extractor (Li et al., 2024). Models like "InceptionTime" utilize convolutional layers. These layers have the capability to capture local correlations, abrupt changes, and frequency components within the data, even if their weights are random (Fawaz et al., 2020). The input signal is one-dimensional (1 channel). Random filters project this data into a higher-dimensional space (e.g., 128 channels). In this high-dimensional space, the non-linear features of the data can become more pronounced. Additionally, since there is no training phase for the model, there is no need for large datasets and time-consuming training. A model built on random features can sometimes exhibit performance rivaling fully trained giant models (Li et al., 2024; Ramanujan, Wortsman, Kembhavi, Farhadi and Rastegari, 2020).

The stages of feature extraction with deep learning are as follows: Data Preparation, Feature Extraction, and Output Processing. Data preparation consists of the stages of filtering, channel selection, normalization, and tensor conversion:

- **Filtering:** The target field within the data is selected (norm data).
- **Channel Selection:** The model operates on single-channel (univariate) signals
- **Normalization:** Z-score normalization on a per-channel basis is performed to reduce noise levels (James et al., 2013). The computation was carried out using "NumPy" (Harris et al., 2020).
- **Tensor Conversion (Batch, Channel, Time Step):** Normalized data are converted into "PyTorch" tensors (Paszke et al., 2019).

  Batch = 1: The number of data inputs fed into the model simultaneously.

  Channel = 1: Single-channel (univariate) signal, solely the norm data.

  Time Step = T: T indicates the number of data points contained in the norm data.

The feature extraction phase includes initialization, model creation, and output stages. Since no training will be conducted during the definition phase, gradient computation in Python is disabled (Paszke et al., 2019). In this way, memory is freed up, and the process is accelerated. In the model creation phase, the "InceptionTime" backbone model is created with random weights. The backbone, that is the "Inception Block", consists of a sequential arrangement of six "Inception Modules" in total and residual connections (see Figure 5). Each "Inception module"

uses a bottleneck (1x1 convolution) layer, parallel 1-dimensional (1D) convolutional filters, 1D max pooling, concatenation, 1D batch normalization (BatchNorm1D), and ReLU (Rectified Linear Unit) to capture features at different scales.

The bottleneck (1x1 convolution) layer lowers computational cost by reducing the number of input channels. It essentially performs a kind of "dimension compression". The kernel sizes (ks) of the parallel 1D convolutional filters are 40, 20, and 10, respectively, and the base number of filters (channels) used in each convolutional layer (nf) is 32. 1D max pooling is used to highlight significant features in the data. The max pool window size (pool_sz) is three meaning that at each step, three consecutive data points are taken side-by-side and evaluated. The window slides by one unit each time (stride = 1). The subsequent bottleneck (1x1 convolution) aligns the channel count with the other branches. Concatenation is executed to merge data coming from all different paths (3 different convolutions and 1 pooling). BatchNorm1D normalizes the data, making training faster and more balanced. ReLU zeroes out negative values, moving the network away from linearity, which allows complex relationships to be learned. To ensure layers use constant values rather than random statistics, the model is used in evaluation mode in Python. The input tensor (the tensor obtained in the data preparation stage) passes through the backbone, forming different representations across time steps and feature dimensions.

The output format obtained in the output phase is as follows (Batch, Feature, Time):

- Batch = 1
- Feature (F) = 128: The representation learned by the model.
- Time (T): The time axis of the input signal.

This format represents the feature vectors obtained for each time step of the time series data. In other words, 128 different features are extracted at each time step for a signal. Finally, the obtained output is processed with two different strategies: Mean Dimensional Reduction and Norm Analysis. In the mean dimensional reduction strategy, the mean across all feature dimensions (F) is calculated for each time step (T) of the output tensor (T, F). The result is a time series of size T. This operation produces a single numerical value representing the center or overall activation level of the signal for each time step. In the norm analysis strategy, the Euclidean magnitude of the feature vector is calculated for each time step of the output tensor. The result is again a time series of size T. This value represents the magnitude or amplitude/power of the signal for each time step and can be used to detect changes in the signal's behavior.

In summary, the deep learning feature extraction phase passes the signal (norm data) through a convolutional "filter", turning it into a richer, multi-dimensional representation.

2.5. Time Offset Estimation

The timestamp information of the peak points identified for each input, as a result of peak detection, is transmitted to the time offset estimation stage. The Nearest Advocate (NAd) method (Schranz, Mayr, Bernhart and Halmich, 2024) was used for time offset estimation. NAd was developed in the literature for event-based time delay estimation in signals obtained from wearable health technologies. NAd was adapted and utilized for the proposed method. In this manner, the NAd method is used to determine the relationship between the timestamp data of the peak points obtained from two different sources and these data in terms of time offset. These relationships could be:

- No delay between the two data,
- One data is ahead of the other, or
- One data is behind the other.

3. RESULTS AND DISCUSSION

Table 1 presents the time offset results obtained by using the timestamp data of peak points acquired from two different sources via the NAd method for five different datasets, utilizing two different proposed approaches and different strategies.

Table 1. Time Offset Estimation Results

Dataset		**Approach: 1**	**Approach: 2 (Feature Extraction)**	
		Norm Data	Mean	Norm
Name	Mean (s)	Estimated Time Offset (s)		
0 ms delay (single altitude)	0.000	**0.000**	**0.000**	**0.000**
0 ms delay (two altitudes)	0.000	**-0.001**	**-0.001**	**-0.001**
5 to 0 ms delay	0.003	-0.005	-0.001	**-0.003**
15 to 9 ms delay	0.013	-0.011	**-0.012**	**-0.012**
30 to 19 ms delay	0.024	-0.021	0.577	**-0.023**

Data obtained from OptiTrack was selected as the reference for the tests. The time information for the data obtained from OptiTrack arrives later than the drone data by the amount of delay value. Therefore, the test results are expected to yield negative values in offset situations. The time offset values estimated in Table 1 are expected to be compatible with the declining delay mean value from the dataset.

In the test results across all approaches, "0 ms delay (single altitude)" was estimated flawlessly (exactly), while "0 ms delay (two altitudes)" was estimated with an error of 1 ms. In estimating a "5 to 0 ms delay", a 5 ms estimate was achieved with the norm data approach, whereas an estimate with a 3 ms (equal to mean) was obtained with the deep learning feature extraction norm option. For the estimation of "15 to 9 ms delay", the closest value was obtained with an error of 1 ms (compared to the mean) using the deep learning feature extraction options, followed by the norm data option with an error of 2 ms. In the estimation of "30 to 19 ms delay," the closest value was obtained with an error of 1 ms (compared to the mean) via the norm option from the deep learning feature extraction options, followed by the norm data option with a 3 ms error. For the "30 ms delay" estimation, the mean dimensional reduction option from deep learning feature extraction yielded a rather poor result.

When all test results are examined generally, the values obtained from the norm data and the norm option of deep learning feature extraction are competitive. However, between the two approaches, the deep learning feature extraction norm option demonstrates better performance in five datasets.

4. CONCLUSION

The proposed method was tested and analyzed on five different datasets obtained in a laboratory environment. The

process is identical up to peak detection. The data to be used in peak detection comes from two different approaches. The first approach involves transferring the norm data directly to peak detection. In the second approach, deep learning-based feature extraction is performed to obtain two different feature outputs. In total, data obtained through three different methods is forwarded to peak detection. In peak detection, the time information corresponding to the peak details is found for each data. In the final stage, the obtained time information is used within the NAd method to perform time offset estimation.

The output of the NAd method is a neutral, negative, or positive result. A negative result indicates that the other data is behind relative to the reference data, while a positive result indicates the exact opposite, that the data is ahead of the reference data. If the result is zero (neutral), no time offset is detected between the two data.

In conclusion, the values obtained in the executed processes have shown that the proposed deep learning-based feature extraction approach estimates time offset more approximate in declining time offset compared to the direct use of norm data.

DECLARATION OF GENERATIVE AI USE

Google Gemini was used to improve the grammar, readability, and fluency of this manuscript and to assist in composing figure structures. The authors subsequently reviewed and edited the content as needed.

REFERENCES

Alipourvarmazabadi, M., & Ebner, M. (2026). Robot localization: a comprehensive survey from classical methods to intelligent autonomy. International Journal of Intelligent Robotics and Applications. https://doi.org/10.1007/s41315-026-00528-9

Dammert, L., Thalmann, T., Monetti, D., Neuner, H., & Mandlburger, G. (2025). A review on UAS trajectory estimation using decentralized Multi-Sensor Systems based on robotic total stations. Sensors, 25(13), 3838. https://doi.org/10.3390/s25133838

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., . . . Petitjean, F. (2020). InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery, 34(6), 1936–1962. https://doi.org/10.1007/s10618-020-00710-y

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., . . . Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning. Springer texts in statistics. https://doi.org/10.1007/978-1-4614-7138-7

Li, X., Xi, W., & Lin, J. (2024). Randomnet: clustering time series using untrained deep neural networks. Data Mining and Knowledge Discovery, 38, 3473–3502. https://doi.org/10.1007/s10618-024-01048-5

Lyu, X., Liu, S., Qiao, R., Jiang, S., & Wang, Y. (2025). Camera, LIDAR, and IMU Spatiotemporal Calibration: Methodological review and research Perspectives. Sensors, 25(17), 5409. https://doi.org/10.3390/s25175409

NumPy Developers. (2025). NumPy v1.23 reference manual. https://numpy.org/doc/1.23/reference/index.html

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems (Article 721, pp. 1–12). Red Hook, NY: Curran Associates Inc.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., & Rastegari, M. (2020). What's hidden in a randomly weighted neural network? Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA.

Ryalat, M., Almtireen, N., Al-refai, G., ElMoaqet, H., & Rawashdeh, N. (2025). Research and education in robotics: A comprehensive review, trends, challenges, and future directions. Journal of Sensor and Actuator Networks, 14(4), 76. https://doi.org/10.3390/jsan14040076

Schranz, C., Mayr, S., Bernhart, S., & Halmich, C. (2024). Nearest advocate: a novel event-based time delay estimation algorithm for multi-sensor time-series data synchronization. EURASIP Journal on Advances in Signal Processing, 2024(1), 46. https://doi.org/10.1186/s13634-024-01143-1

Sivrikaya, F., & Yener, B. (2004). Time synchronization in sensor networks: a survey. IEEE Network, 18(4), 45–50. https://doi.org/10.1109/mnet.2004.1316761

Ušinskis, V., Nowicki, M., Dzedzickis, A., & Bučinskas, V. (2025). Sensor-fusion based navigation for autonomous

mobile robot. Sensors, 25(4), 1248. https://doi.org/10.3390/s25041248

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., . . . SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 17(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

PROMPT ENGINEERING IN LARGE LANGUAGE MODELS: A SYSTEMATIC REVIEW OF METHODS, APPLICATIONS, AND ETHICAL CONSIDERATIONS

Sara NAGHIB ZADEH[1]

Eyüp FİDAN[2]

1. INTRODUCTION

Human–computer interaction has evolved from punched cards and command-line systems to graphical interfaces and, more recently, Large Language Models (LLMs) (Shneiderman, 2016; Stephanidis, 2024). A major milestone in this transformation was the release of ChatGPT by OpenAI in 2022, which rapidly became one of the fastest-adopted technologies worldwide (Macagnano, 2025). This development demonstrated that effective interaction with artificial intelligence systems requires new skills beyond traditional software usage, leading to the emergence of prompt engineering as a distinct research field (Y. Qian, 2025).

Prompt engineering is the process of designing and optimizing textual instructions (prompts) to obtain accurate and meaningful outputs from LLMs (Schulhoff et al., 2024). This field combines elements of linguistics, cognitive science, software engineering, and communication theory. Studies show that factors such as instruction clarity, information order, and

[1] Dr. Lecture, Halic University, Vocational School, Department of Computer Programming, ORCID: 0009-0005-6959-1165.

[2] Halic University, Vocational School, Department of Artificial Intelligence Operator, ORCID: 0009-0000-8389-2793

wording significantly affect model performance (Choi & Chang, 2025). Well-structured prompts can improve the quality and reliability of AI-generated responses, while poorly designed prompts may produce inaccurate or misleading outputs (Sahoo et al., 2024; Naser, 2025).

Recent advances in transformer-based LLMs, including ChatGPT, Claude, and Gemini, have expanded the use of AI across education, software development, healthcare, and business applications (Brown et al., 2020). Consequently, prompting techniques such as zero-shot, few-shot, and chain-of-thought prompting have become important research topics in recent years (Debnath et al., 2025).

This study aims to review the fundamental concepts and techniques of prompt engineering and examine its applications in large language models. The research is based on a qualitative literature review of scientific articles, technical reports, and official publications published between 2020 and 2026, including sources from ACM Digital Library, Google Scholar, arXiv, and leading AI companies such as Anthropic, Google AI, and OpenAI.

2. PROMPT ENGINEERING

Prompt engineering refers to the process of designing and optimizing textual inputs (prompts) to obtain accurate and task-oriented outputs from large language models (Schulhoff et al., 2024). The field gained significant importance with the widespread adoption of systems such as OpenAI's ChatGPT and has become an essential skill in the effective use of artificial intelligence (Zhang & Shao, 2024). Although prompts were previously discussed in GPT-3 research, the rapid growth of generative AI during 2022–2023 established prompt engineering as an independent research and application domain (Brown et al., 2020). Reports by Stanford HAI also show a substantial increase

in the academic use of the term during this period (Xu, 2026). As shown in Figure 1, prompt engineering involves organizing components such as context, instructions, examples, constraints, and output formatting to guide models toward accurate and meaningful responses.

The significance of prompt engineering lies in the strong dependence of model outputs on how user requests are formulated (Liu et al., 2023). Even within the same model, minor variations in prompt design can result in substantially different output quality. Well-structured prompts that provide sufficient context and clear instructions tend to produce more coherent, analytical, and useful responses (Geroimenko, 2025). According to the McKinsey Global Institute, in many organizational applications, the quality of the prompt can have a greater impact on output performance than the selection of the model itself (Jain & Kansal, 2025).

An effective prompt typically consists of several key components, including context, task or instruction, input data, output format, constraints, and in some cases, illustrative examples (Schulhoff et al., 2024). These elements help the model better interpret user intent and generate more relevant responses. Although not all components are required in every task, the absence of critical elements may lead to a decline in output quality. Therefore, prompt engineering has become a fundamental factor in enhancing the efficiency and effectiveness of large language models (Chen et al., 2025).

3. LARGE LANGUAGE MODELS (LLMS)

Large Language Models (LLMs) are deep learning–based neural networks trained on massive volumes of textual data, enabling them to understand, generate, and process natural language (Brown et al., 2020). The primary advancement in this

field began with the introduction of the Transformer architecture in 2017, followed by the development of influential models such as BERT and GPT (Devlin et al., 2019). Subsequently, the emergence of systems such as ChatGPT significantly accelerated the global adoption of generative artificial intelligence (Storey et al., 2025).

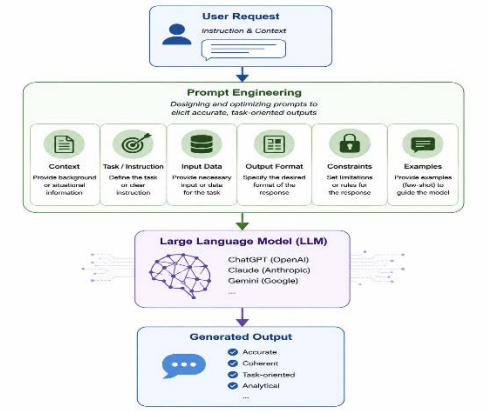


Figure 1. Conceptual framework of prompt engineering and its influence on large language model outputs.

Within the Transformer architecture, the self-attention mechanism enables the modeling of relationships between words regardless of their positional distance in a sequence. The training process of these models typically consists of three main stages: pre-training, fine-tuning, and reinforcement learning from human feedback (RLHF) (Bommasani et al., 2021). These stages collectively enable the model not only to learn linguistic structures but also to generate outputs that are better aligned with user intentions (C. Qian et al., 2024).

In recent years, various organizations have developed advanced LLMs with distinct capabilities. As illustrated in Table 1, models such as GPT-5.2, Claude Opus 4.5/4.6, and Gemini 3 Pro demonstrate strong performance in areas including reasoning, programming, long-context processing, and multimodal

capabilities (Wang et al., 2026). In addition, open-weight models such as Llama 4 and DeepSeek V4-Pro have significantly increased accessibility for researchers and organizations. Among the most prominent current systems are ChatGPT by OpenAI, Claude by Anthropic, and Gemini by Google DeepMind (Bai et al., 2022).

Table 1. Prominent Large Language Models and Their Key Characteristics

Model	Developer	Year	Key Feature
GPT-5.2	OpenAI	2025	400K token context window, strong mathematical reasoning
Claude Opus 4.5/4.6	Anthropic	2025	Leading performance in coding benchmarks, strong safety alignment
Gemini 3 Pro	Google DeepMind	2025	1M token context window, “Deep Think” mode
DeepSeek V4-Pro	DeepSeek	2026	Open-weight, low-cost and highly accessible
Llama 4	Meta AI	2025	Open-weight model with multimodal capabilities
Mistral Large 2	Mistral AI	2025	Optimized for enterprise applications
Grok 4.1	xAI	2025	Focus on reducing hallucination rate

LLMs are capable of performing a wide range of tasks, including text generation, translation, summarization, question answering, code generation, and logical reasoning. However, they still face important limitations such as hallucination, context window constraints, knowledge cutoff, and biases inherited from training data. Despite these challenges, LLMs have become one of the most transformative technologies in the field of artificial intelligence (Brown et al., 2020).

4. CORE TECHNIQUES OF PROMPT ENGINEERING

4.1. Zero-Shot Prompting

Zero-shot prompting represents the most basic form of prompt engineering, in which a task is directly assigned to the

model without providing any prior examples (Brown et al., 2020). This approach relies on the knowledge acquired during the training phase and generally performs well in common tasks such as translation, summarization, and sentiment analysis. However, its performance may degrade in more complex or domain-specific problems where additional guidance is required (Almeida, 2025).

4.2. Few-Shot Prompting

In few-shot prompting, a small number of input–output examples are provided before the main query in order to guide the model toward the desired pattern (Brown et al., 2020). This technique leverages the model's in-context learning capability and improves performance in structured and classification-based tasks. Empirical studies indicate that few-shot prompting can significantly outperform zero-shot approaches in various applications (Liu et al., 2023).

4.3. Chain-of-Thought (CoT) Prompting

Chain-of-thought prompting encourages the model to generate intermediate reasoning steps before producing a final answer (Wei et al., 2022). This method is particularly effective in mathematical, logical, and multi-step reasoning tasks, as it improves the model's ability to handle complex problem-solving processes. A simplified variant, known as zero-shot Chain-of-Thought, can be activated by incorporating phrases such as "think step by step" (Kojima et al., 2022).

4.4. Role-Based Prompting

Role-based prompting assigns a specific role or expertise to the model, such as "cybersecurity expert" or "primary school teacher" (Schulhoff et al., 2024). This technique influences the tone, depth, and style of the generated responses, enabling the model to adapt its outputs according to the assigned role. As a

result, it is widely used in educational systems, technical consulting, and customer service applications (Y. Qian, 2025).

4.5. Advanced Techniques

In recent years, more advanced prompt engineering methods have been introduced. Tree-of-Thought (ToT) enables the exploration of multiple reasoning paths simultaneously, making it suitable for complex problem-solving tasks (Yao et al., 2023). Retrieval-Augmented Generation (RAG) enhances model responses by integrating external knowledge sources, resulting in more up-to-date and accurate outputs. In addition, Self-Consistency leverages multiple reasoning paths to select the most consistent answer, while Meta-Prompting guides the model to iteratively improve its own prompts (Han et al., 2024).

4.6. Comparative Analysis of Techniques

As shown in Table 2, each prompt engineering technique is suited to specific types of tasks. Methods such as zero-shot and role-based prompting are computationally efficient and appropriate for general-purpose applications. In contrast, more advanced approaches such as Tree-of-Thought and RAG require higher computational resources but provide superior performance in complex and knowledge-intensive tasks.

Table 2. Comparison of Core Prompt Engineering Techniques

Technique	Number of Examples	Computational Cost	Best Application Domain
Zero-Shot	0	Low	Simple and general tasks
Few-Shot	1–5	Medium	Structured tasks
Chain-of-Thought	0–5	Medium	Mathematics and reasoning
Role Prompting	0	Low	Tone and expertise control
Tree-of-Thought	0	High	Complex problem solving
RAG	0	High	Knowledge-intensive and up-to-date tasks

Recent studies suggest that prompt engineering is gradually evolving from an empirical practice into a structured scientific discipline. The comprehensive "The Prompt Report," which analyzed more than 1,500 scientific papers, categorizes a wide range of prompt engineering techniques and highlights the rapid expansion and formalization of this research field (Schulhoff et al., 2024).

5. APPLICATIONS OF PROMPT ENGINEERING ACROSS DIFFERENT DOMAINS

5.1. Education

Education is one of the major application areas of prompt engineering and large language models. These technologies support personalized learning and intelligent tutoring systems by adapting explanations to learners' needs, evaluating knowledge levels, and providing targeted feedback through well-designed prompts (Kasneci et al., 2023; Hayati et al., 2025).

For educators, prompt engineering also facilitates automation of instructional tasks such as lesson plan generation, creation of assessment questions with varying cognitive levels, rubric design, and feedback production. For example, structured prompts can quickly generate complete and standardized lesson plans (Paz et al., 2024).

Empirical studies show that LLM-based personalized learning can improve student performance (HGSE, 2024). However, challenges such as academic dishonesty and potential reduction in critical thinking skills remain important concerns (Kasneci et al., 2023).

5.2. Software Development

Software development is among the fastest-growing fields in the adoption of large language models (Noy & Zhang, 2023).

Tools such as GitHub Copilot have significantly accelerated coding workflows, where prompt engineering plays a crucial role in improving output quality (Nabi et al., 2025).

Key applications include code generation, debugging, code review, and unit test generation. Well-designed prompts can produce complete software components (e.g., APIs), detect errors and suggest fixes, identify security or performance issues, and generate edge-case test scenarios (Q. Zhang et al., 2023).

Industry reports show widespread adoption of such tools in organizations, with noticeable productivity gains in software engineering tasks (Deloitte, 2025). Nevertheless, human oversight remains essential due to ongoing concerns about the reliability and quality of AI-generated code (Abbas et al., 2025).

5.3. Healthcare

In healthcare, large language models are mainly used as decision-support systems rather than replacements for clinicians (Singhal et al., 2023). Prompt engineering enables efficient summarization of medical literature, generation of patient-friendly explanations, and improved clinical documentation (Patil et al., 2024).

These applications reduce the time required for medical data processing and enhance workflow efficiency in healthcare systems (J. Wang et al., 2026). Moreover, institutions such as Mayo Clinic and Cleveland Clinic have conducted pilot studies using LLM-based systems (Xu, 2026). The increasing approval of AI-based medical devices by the U.S. FDA further demonstrates the growing role of these technologies in healthcare (Zand et al., 2026).

5.4. Business and Organizational Applications

In business contexts, prompt engineering is widely used to improve productivity in marketing, customer service, data analysis, and legal operations (Chemingui & Ahmad, 2025).

In marketing, it supports the creation of targeted content such as advertisements, social media posts, and product descriptions. In customer service, Retrieval-Augmented Generation (RAG) systems enable fast and knowledge-grounded responses. In data analysis, natural language queries can be transformed into structured database queries, while in legal domains, contract analysis and documentation processes are significantly accelerated (Cheng et al., 2025; Kobeissi et al., 2023).

Industry reports indicate that many global organizations are integrating LLMs into their workflows and creating new AI-related job roles (Deloitte, 2025). Research also confirms that these technologies significantly enhance productivity across multiple business functions, including marketing, customer service, and software development (Lakhwani & Omkar, 2020).

6. CURRENT STATUS OF PROMPT ENGINEERING IN THE GLOBAL AND TURKISH CONTEXT AND ITS ETHICAL AND LEGAL DIMENSIONS

6.1. Global Perspective

Between 2023 and 2026, prompt engineering has become increasingly institutionalized in both academic and industrial settings as a key component of large language model (LLM) development (Schulhoff et al., 2024). Leading universities such as MIT, Stanford, Carnegie Mellon, and Oxford have incorporated courses on LLM applications and AI governance

into their curricula (Xu, 2026). Additionally, the "ChatGPT Prompt Engineering for Developers" course by DeepLearning.AI, supervised by Andrew Ng, attracted over one million learners, reflecting the rapid global expansion of interest in this field (Ma et al., 2024). In industry, platforms such as PromptBase have emerged as marketplaces for prompt exchange and monetization (Schneider & Abraham, 2025), while companies like OpenAI, Anthropic, and Google have published official guidelines for prompt engineering. According to Gartner (2025), the role of "Prompt Engineer" is among the fastest-growing technology occupations (George et al., 2023). The Stanford AI Index (2026) further indicates that generative AI adoption has reached 88% in organizations and has rapidly spread to more than half of the global population, with the United States and China leading development, the European Union focusing on regulatory frameworks such as the AI Act, and countries like India leveraging generative AI for competitiveness in software services (Xu, 2026).

6.2. Developments in Türkiye

In Türkiye, prompt engineering is currently in a growth phase characterized by increasing adoption and structural limitations (Üyesi et al., 2024). Major companies such as Türk Telekom, Turkcell, and Garanti BBVA, along with several public institutions, have initiated pilot projects based on large language models (Nabiyev et al., 2025). The Ministry of Health also tested an LLM-based healthcare assistant integrated into the e-Nabız system in 2025. On the academic side, universities including ODTÜ, Boğaziçi, Bilkent, and Sabancı are actively conducting research in natural language processing and LLM technologies (Şenyüz et al., 2025), while TÜBİTAK has prioritized Turkish language model development in its 2024–2026 research agenda. In addition, both academic and industry-driven open-source initiatives, such as those led by Trendyol, are expanding.

However, a major limitation remains the lack of high-quality Turkish-language datasets, which reduces model performance compared to English. At the same time, this gap represents a strategic opportunity for developing indigenous language models and strengthening technological sovereignty in Türkiye (Zümberoğlu et al., 2025).

6.3. Ethical and Legal Dimensions

The rapid expansion of prompt engineering has introduced significant ethical, legal, and security challenges (Geroimenko, 2025a). One of the key risks is prompt injection attacks, where malicious inputs manipulate model behavior, as well as jailbreaking techniques that bypass safety mechanisms (Perez & Ribeiro, 2023). In addition, issues related to intellectual property and ownership of AI-generated content remain unresolved across legal systems, creating regulatory uncertainty (Srivastava & Tripathi, 2025). On the policy side, the European Union AI Act, which entered into force in August 2024, introduced phased regulations including restrictions on high-risk systems and transparency requirements for general-purpose AI models (van Kolfschooten & van Oirschot, 2024). In Türkiye, efforts to develop a comprehensive national AI legal framework continue alongside existing KVKK-related considerations (Galandarli, 2025). Moreover, reports indicate declining transparency in foundation models and a growing number of AI-related incidents, suggesting that regulatory and accountability mechanisms are not keeping pace with technological advancement (Bommasani et al., 2023).

7. CONCLUSION AND FUTURE OUTLOOK

This study demonstrates that prompt engineering is not merely a technical skill but a fundamental capability for effectively interacting with large language models (Schulhoff et

al., 2024; Y. Qian, 2025). The analysis of different prompting strategies, ranging from zero-shot and few-shot methods to advanced approaches such as Chain-of-Thought, Tree-of-Thought, and RAG, shows that prompt design significantly influences model performance and output quality (Wei et al., 2022; Yao et al., 2023; Liu et al., 2023).

The findings further indicate that aligning prompting techniques with task requirements is essential for optimizing performance, as prompt quality can in some cases have a greater impact than model selection itself (Jain & Kansal, 2025). Evidence from multiple domains—including education, software development, healthcare, and business—confirms that effective prompt engineering improves productivity, decision-making, and task automation (Kasneci et al., 2023; Noy & Zhang, 2023; Singhal et al., 2023; Chemingui & Ahmad, 2025). However, ethical, security, and governance concerns such as prompt injection, intellectual property ambiguity, and regulatory gaps remain critical challenges (Perez & Ribeiro, 2023; Srivastava & Tripathi, 2025; Geroimenko, 2025).

Recent advances in large language models demonstrate strong progress in reasoning and multimodal capabilities, yet issues such as reduced transparency and increasing AI-related incidents highlight the need for stronger accountability mechanisms (Bommasani et al., 2023; Xu, 2026).

Looking forward, prompt engineering is expected to evolve from manual instruction design toward autonomous, agent-based systems where prompts function as modular components in complex workflows. In addition, multimodal prompting integrating text, image, audio, and video will further expand human–AI interaction capabilities (Schulhoff et al., 2024). Overall, these developments suggest that prompt engineering will not only enhance model performance but also

fundamentally reshape the structure of human–AI collaboration in the coming years.

REFRENCES

Abbas, T., Rathore, S. A., Turki, A., Khan, S., Alghushairy, O., & Daud, A. (2025). Enhancing software engineering with AI: Innovations, challenges, and future directions. *IET Software, 2025*(1), 5691460.

Almeida, J. (2025). Prompt engineering: A comparative study of prompting techniques in AI language models. *2025 15th IEEE Integrated STEM Education Conference (ISEC 2025).*

Bommasani, R., Klyman, K., Longpre, S., Kapoor, S., Maslej, N., Xiong, B., Zhang, D., & Liang, P. (2023). *The foundation model transparency index.* arXiv.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., et al. (2020). *Language models are few-shot learners*. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Chemingui, H., & Ahmad, M. (2025). *Advanced AI and prompt engineering techniques and resources.*

Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2025). Unleashing the potential of prompt engineering for large language models. *Patterns, 6*(6), 101260.

Cheng, M., Luo, Y., Ouyang, J., Liu, Q. I., Liu, H., Li, L. I., Yu, S., Zhang, B., Cao, J., Ma, J., Wang, D., Chen, E., Liu, Q., & Li, L. (2025). *A survey on knowledge-oriented retrieval-augmented generation.*

Choi, W. C., & Chang, C. I. (2025). *A survey of techniques, key components, strategies, challenges, and student perspectives on prompt engineering for large language models (LLMs) in education.*

Debnath, T., Siddiky, M. N. A., Rahman, M. E., Das, P., Guha, A. K., Rahman, M. R., & Kabir, H. M. D. (2025). *A comprehensive survey of prompt engineering techniques in large language models.*

Galandarli, A. (2025). Mitigating AI risks: A comparative analysis of data protection impact assessments under GDPR and KVKK. *Journal of Data Protection and Privacy, 7*(3), 252–273.

George, A. S., & George, A. S. H. (2023). The emergence of prompt engineering in India: Assessing the potential for a new generation of AI talent. *Partners Universal International Innovation Journal, 1*(6), 1–18.

Geroimenko, V. (2025a). Key challenges in prompt engineering. In *Prompt engineering and generative AI* (pp. 85–102).

Geroimenko, V. (2025b). Key techniques for writing effective prompts. In *Prompt engineering and generative AI* (pp. 37–83).

Han, B., Susnjak, T., & Mathrani, A. (2024). Automating systematic literature reviews with retrieval-augmented generation: A comprehensive overview. *Applied Sciences, 14*(19), 9103.

Hayati, L., Suyidno, S., Sauqina, S., Khairunnisa, Y., & NF, A. R. (2025). The effect of Socratic questioning-based student worksheets on improving students' critical thinking skills. *Jurnal Pendidikan MIPA, 26*(4), 2542–2556.

Jain, N., & Kansal, J. (2025). Application of McKinsey 7S framework as a strategic tool for a knowledge-based organizational development. *IEEE Engineering Management Review, 53*(2), 14–26.

Kobeissi, M., Assy, N., Gaaloul, W., Defude, B., Benatallah, B., & Haidar, B. (2023). Natural language querying of

process execution data. *Information Systems, 116,* 102227.

Lakhwani, M., & Omkar, D. (2020). The impact of technology adoption on organizational productivity. *Journal of Industrial Distribution & Business, 11*(4), 7–18.

Ma, A., Ong, J., & Tan, S. S. (2024). *AI for humanity: Building a sustainable AI for the future.*

Macagnano, V. (2025). *Leveraging artificial intelligence in B2B sales: Challenges and opportunities.*

Nabi, Z. M., Jaman, I., Bose, U., Hossain, T. T., & Kamal, D. (2025). *AI-assisted code generation tools: A new frontier in software development.*

Nabiyev, A. B., & Ovenc, G. (2025). *The state of the Turkish fintech ecosystem and case studies on bank-fintech collaborations.*

Naser, M. Z. (2025). A review of prompt engineering techniques for large language models. *International Journal of Human–Computer Interaction.*

Patil, R., Heston, T. F., & Bhuse, V. (2024). Prompt engineering in healthcare. *Electronics, 13*(15), 2961.

Paz, L., Berg, M., Kreibich, S., & Werth, D. (2024). Assessing the impact of prompting techniques on short-term curriculum design: A comparative approach. *ICERI2024 Proceedings*, 4865–4872.

Qian, C., He, B., Zhuang, Z., Deng, J., Qin, Y., Cong, X., Zhang, Z., Zhou, J., Lin, Y., Liu, Z., & Sun, M. (2024). Tell me more! Towards implicit user intention understanding of language model driven agents. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1088–1113.

Qian, Y. (2025). Prompt engineering in education: A systematic review of approaches and educational applications. *Journal of Educational Computing Research, 63*(7–8), 1782–1818.

Schneider, J., & Abraham, R. (2025). *AI exchange platforms.* arXiv.

Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., et al. (2024). *The prompt report: A systematic survey of prompting techniques. arXiv preprint arXiv:2406.06608.*

Şenyüz, B., Özgen, E., & Oğuzcan, A. U. (2025). Integrating GenAI into communication education for "Generation Prompt": An exploration of academics' perspectives on its benefits, challenges, and future prospects in Türkiye. *Encontros Bibli, 30,* e103465.

Srivastava, B., & Tripathi, Y. (2025). The legal and ethical disruption of authorship: Navigating copyright infringement and originality in generative artificial intelligence and large language models. *SSRN Electronic Journal.*

Stephanidis, C. (2024). The HCI discipline past, present and future. In *Foundations and fundamentals in human-computer interaction* (pp. 1–54).

Storey, V. C., Yue, W. T., Zhao, J. L., & Lukyanenko, R. (2025). Generative artificial intelligence: Evolving technology, growing societal impact, and opportunities for information systems research. *Information Systems Frontiers, 27*(5), 2081–2102.

Üyesi, Ö., Üniversitesi, K. D., Sosyal, K., Myo, B., & Yönetimi, İ. (2024). Prompt engineering awareness: A study on

Google Trends data. *Uluslararası Sosyal ve Ekonomik Çalışmalar Dergisi, 5*(2), 248–268.

van Kolfschooten, H., & van Oirschot, J. (2024). The EU Artificial Intelligence Act (2024): Implications for healthcare. *Health Policy, 149*, 105152.

Wang, J., Shi, E., Yu, S., Wu, Z., Hu, H., Ma, C., Dai, H., Yang, Q., Kang, Y., Wu, J., Yue, C., Zhang, H., Liu, Y., Pan, Y., Liu, Z., Sun, L., Li, X., Ge, B., Jiang, X., & Zhang, S. (2026). Prompt engineering for healthcare: Methodologies and applications. *Meta-Radiology, 4*(1), 100190.

Wang, Q., Ye, H., Kim, J., Ke, J., Wang, Y., Kuo, M., Shao, Z., Li, D., Lin, Y., Jiang, T., Wei, C., Qian, Q., Wen, W., Li, H., & Chen, Y. (2026). *T2S-Bench & structure-of-thought: Benchmarking and prompting comprehensive text-to-structure reasoning*. arXiv.

Xu, W. (2026). *Human-centered artificial intelligence (HCAI): Foundations and approaches.* arXiv.

Zand, J., Suvakov, M., Overgaard, S. M., Kozikowski, C. G., Jimison, R. C., Nelson, K. B., Silvano, C. J., Meiners, L. M., Fan, J. W., Schultz, K. E., Carter, R. E., Wieland, M. L., Patten, C. A., Dugani, S. B., Weis, J. A., Poe, J. D., Holmes, D. R., Griffiths, L. G., Dieter, H. L., et al. (2026). Building artificial intelligence and data literacy across the health care research workforce in the Mayo Clinic Center for Clinical and Translational Science. *Mayo Clinic Proceedings, 101*(1), 17–25.

Zhang, H., & Shao, H. (2024). Exploring the latest applications of OpenAI and ChatGPT: An in-depth survey. *CMES - Computer Modeling in Engineering and Sciences, 138*(3), 2061–2102.

Zhang, Q., Zhang, T., Zhai, J., Fang, C., Yu,B., Sun, W., & Chen, Z. (2023). A critical review of large language model on software engineering: An example from ChatGPT and automated program repair. *Proceedings of the Conference on Automated Program Repair.*

Zümberoğlu, K. B., Dik, S. Z., Karadeniz, B. S., & Sahmoud, S. (2025). Towards better sentiment analysis in the Turkish language: Dataset improvements and model innovations. *Applied Sciences, 15*(4), 2062.

DEEP LEARNING-BASED DRUG–TARGET INTERACTION PREDICTION: DATASETS, MODELS, AND APPLICATIONS

Sara NAGHIBZADEH[1]

Mustafa YILMAZ[2]

1. INTRODUCTION

The process of drug discovery and development is considered one of the most complex and costly procedures in biomedical sciences. The development of a new drug typically requires 10 to 15 years and involves investments amounting to billions of dollars. One of the most critical stages of this process is the accurate identification of interactions between drug molecules and target proteins. These interactions determine the extent to which a drug can be effective, safe, and clinically applicable for disease treatment (Berdigaliyev & Aljofan, 2020).

Over the past decade, remarkable advances in artificial intelligence, particularly in deep learning, have brought significant transformations to pharmaceutical sciences and biotechnology. Deep learning models are capable of analyzing massive volumes of biological, chemical, and molecular data and extracting complex patterns that are difficult to identify using traditional approaches. This capability has made drug–target interaction (DTI) prediction one of the most important

[1] Dr. Lecture, Halic University, Vocational School, Department of Computer Programming, ORCID: 0009-0005-6959-1165.

[2] Halic University, Vocational School, Department of Big Data Analytics, ORCID: 0009-0008-6883-6946.

applications of artificial intelligence in biomedicine (Gupta et al., 2021).

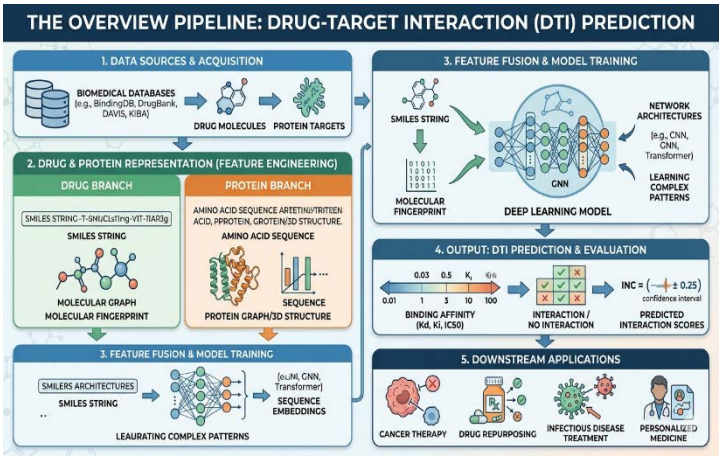


Figure 1. A comprehensive overview pipeline of deep learning-based drug–target interaction (DTI) prediction.

Drug–Target Interaction (DTI) refers to the process through which a drug molecule interacts with a specific biological target such as a protein, receptor, enzyme, or nucleic acid structure. Accurate prediction of these interactions is essential not only for the development of novel therapeutics, but also for identifying adverse drug effects, drug repurposing, and designing personalized treatment strategies (Suruliandi et al., 2024).

Traditional approaches such as molecular docking, pharmacophore modeling, and physics- and quantum chemistry-based simulations have long been utilized in this field. Although these methods provide acceptable accuracy in certain applications, they suffer from several limitations, including high computational cost, long execution times, and strong dependence on structural information. In contrast, machine learning and deep learning models enable faster and more accurate predictions through direct learning from data (Abbasi et al., 2020).

In recent years, advanced architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Transformers, and Graph Neural Networks (GNNs) have been widely applied to DTI prediction tasks . These models are capable of accurately modeling molecular structures, protein sequences, and complex biological features (Panahandeh & Mansouri, 2025) .The overall workflow and pipeline of deep learning-based DTI prediction, from data acquisition and feature representation to model training and downstream applications, are schematically illustrated in Figure 1.

The aim of this chapter is to provide a comprehensive review of deep learning applications in drug–target interaction prediction, introduce major datasets and computational models, analyze representative case studies, discuss existing challenges, and present future perspectives in this rapidly evolving field.

2. FUNDAMENTALS OF DRUG–TARGET INTERACTIONS

Drug–target interaction (DTI) is considered one of the most fundamental concepts in pharmaceutical and biomedical sciences. The effectiveness of any drug depends on the manner in which it interacts with a specific biological target. These targets generally include proteins, receptors, enzymes, or nucleic acids (Yu et al., 2025).

When a drug molecule binds to its target, a series of biological reactions is initiated, which may lead to inhibition, activation, or modulation of biological functions. Therefore, the accurate prediction of these interactions plays a decisive role in the success of the drug discovery and development process (Mak et al., 2024).

Traditional drug discovery approaches are primarily based on laboratory experiments and chemical compound screening. In addition to being highly expensive, these procedures require extensive time and substantial resources. Moreover, many experimentally tested compounds fail during later development stages and never reach the pharmaceutical market (Yu et al., 2025).

In response to these limitations, computational or in silico approaches have been developed. These methods utilize molecular and biological data to predict potential interactions. Molecular docking is one of the most widely used traditional computational techniques that models the binding behavior between a drug molecule and a target protein. However, these approaches often require high computational power and, in many cases, are unable to fully represent the dynamic nature of biological systems (Ru et al., 2025).

In contrast, deep learning models are capable of learning complex patterns directly from data without relying on explicit physical rules. This capability has transformed deep learning into a powerful tool for DTI prediction (Abbasi et al., 2020).As shown in Table 1, artificial intelligence–based approaches, particularly deep learning methods, provide important advantages over traditional computational techniques in terms of pattern learning and large-scale data analysis.

This comparison demonstrates why deep learning has emerged as one of the most important approaches in modern drug discovery research.

Table 1. Comparison of Traditional and Artificial Intelligence-Based Methods

Method	Advantages	Limitations	Applications
Molecular Docking	High structural accuracy	High computational cost	Molecular binding analysis
Pharmacophore Modeling	Fast execution	Limited structural diversity	Initial virtual screening
Machine Learning	Large-scale data analysis	Feature dependency	Interaction classification
Deep Learning	Learning complex patterns	Requires large datasets and GPU resources	DTI prediction

3. DATASETS USED IN DRUG–TARGET INTERACTION PREDICTION

High-quality datasets are essential for the success of deep learning models in Drug–Target Interaction (DTI) prediction. These datasets generally include information about drug molecules, protein targets, and experimentally measured binding affinities. Drug data often contain molecular structures and chemical properties, while protein data include amino acid sequences and structural information (X. Chen et al., 2016).

Several benchmark datasets are widely used in DTI research, including BindingDB, DrugBank, DAVIS, and KIBA (Debnath et al., 2025a).Figure 2 illustrates examples of commonly used datasets and their role in DTI prediction studies.

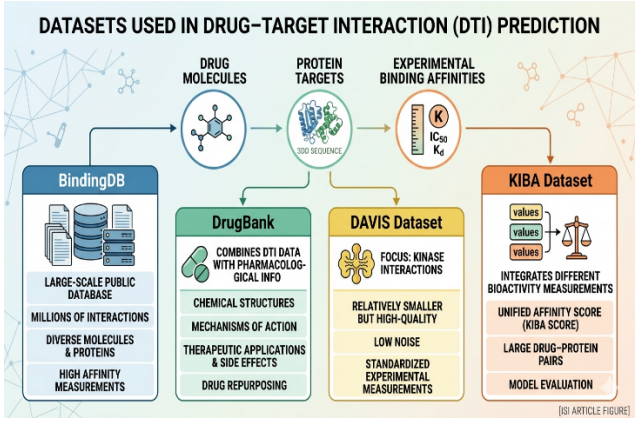


Figure 2. Datasets used in drug target interaction prediction

3.1. BindingDB

BindingDB is one of the largest public databases for DTI studies. It contains millions of experimentally validated interactions between drug molecules and proteins, along with binding affinity measurements. Due to its large scale and molecular diversity, it is commonly used for training deep learning models (Alhadethy et al., 2025).

3.2. DrugBank

DrugBank combines drug–target interaction data with pharmacological and biological information such as chemical structures, mechanisms of action, therapeutic applications, and side effects. It is widely used in drug repurposing and biomedical AI research (X. Chen et al., 2016).

3.3. DAVIS Dataset

The DAVIS dataset focuses mainly on kinase-related drug–target interactions. Although relatively smaller than other datasets, it is considered a high-quality benchmark because of its low noise and standardized experimental measurements (Salem, 2024).

3.4. KIBA Dataset

KIBA integrates different bioactivity measurements into a unified affinity score known as the KIBA score. It contains a large number of drug–protein interaction pairs and is frequently used for evaluating deep learning models in DTI prediction (Barlow et al., 2022).

3.5. Dataset Challenges

Biomedical datasets often contain noisy data, missing values, and class imbalance problems, which may affect model performance. Therefore, preprocessing techniques such as normalization, data cleaning, and resampling are important steps in DTI prediction studies (Mirza et al., 2019).

4. FEATURE ENGINEERING IN DRUG–TARGET INTERACTION PREDICTION

Feature engineering is a crucial step in Drug–Target Interaction (DTI) prediction because raw biological and chemical data must be converted into numerical representations that can be processed by machine learning and deep learning models. Effective feature representation improves prediction accuracy and model performance (Ru et al., 2025).

4.1. Representation of Drug Molecules

Drug molecules are commonly represented using SMILES strings, which describe molecular structures as text sequences. These representations are widely used in deep learning models such as CNNs and Transformers (Hu et al., 2024).

In addition, graph-based representations have become increasingly popular. In molecular graphs, atoms are represented as nodes and chemical bonds as edges. This approach enables

Graph Neural Networks (GNNs) to learn structural relationships directly from molecular structures (Reiser et al., 2022).

4.2. Representation of Protein Targets

Proteins are generally represented as amino acid sequences. Deep learning models such as CNNs and RNNs can analyze these sequences to identify important biological patterns and interaction regions (Karimi et al., 2019).

Recent studies also use protein embeddings and structural information obtained from databases or prediction systems such as AlphaFold to improve prediction performance (Karimi et al., 2019).

4.3. Molecular Fingerprints and Physicochemical Features

Molecular fingerprints encode chemical substructures into numerical vectors and are widely used in drug discovery research. Common fingerprint methods include ECFP, Morgan fingerprints, and MACCS keys (Pant et al., 2026).

Physicochemical descriptors such as molecular weight, hydrophobicity, and polar surface area are also important because they provide biologically meaningful information related to molecular behavior and binding affinity (Davis & Leeson, 2023).

4.4. Sequence-Based and Graph-Based Features

Sequence-based descriptors such as Amino Acid Composition (AAC), Dipeptide Composition (DPC), and Position-Specific Scoring Matrix (PSSM) are commonly used to represent protein information (Pande et al., 2019).

Graph-based feature learning has become one of the most effective approaches in modern DTI prediction. GNN models automatically learn molecular representations from graph structures and can capture complex chemical relationships more

effectively than traditional handcrafted features (O. Zhang et al., 2025).

4.5. Challenges in Feature Engineering

Feature engineering in biomedical AI still faces challenges such as high-dimensional data, heterogeneous biological information, and limited interpretability. To address these issues, researchers increasingly use automated representation learning, self-supervised learning, and multi-modal deep learning techniques (Alqudah & Moussavi, 2025).

Overall, feature engineering plays a major role in improving the accuracy and robustness of deep learning-based DTI prediction systems.

5. DEEP LEARNING MODELS IN DRUG–TARGET INTERACTION PREDICTION

Deep learning models are widely used in Drug–Target Interaction (DTI) prediction because they can automatically learn complex relationships from biological and chemical data and significantly improve prediction accuracy (Hu et al., 2024). Commonly used architectures in DTI studies include CNNs, RNNs, GNNs, and Transformer-based models, and their main characteristics, advantages, and limitations are summarized in Table 2 (Alhadethy et al., 2025).

Convolutional Neural Networks (CNNs) are frequently applied to protein sequences and SMILES representations to automatically extract important local features from biological data (Hu et al., 2024). CNN models are computationally efficient and effective in feature extraction; however, they may have limitations in capturing long-range dependencies within sequences (Hu et al., 2024).

Recurrent Neural Networks (RNNs), including advanced variants such as LSTM and GRU, are designed for sequential biological data and can effectively model contextual relationships in protein sequences (Barlow et al., 2022). These models are commonly used in DTI prediction because they can capture temporal dependencies and sequential information more successfully than traditional machine learning approaches (Barlow et al., 2022).

Graph Neural Networks (GNNs) represent molecules as graph structures in which atoms are nodes and chemical bonds are edges, enabling models to preserve molecular topology and structural relationships more effectively (Alhadethy et al., 2025). As a result, GNN-based approaches generally provide higher predictive performance in modern DTI prediction tasks compared to many sequence-based methods (Alhadethy et al., 2025).

Transformer-based models utilize attention mechanisms to capture long-range dependencies and contextual information in biological sequences and molecular data (Choi & Lee, 2023). Due to their strong capability in representation learning and contextual analysis, Transformer architectures have recently become highly popular in drug discovery and DTI prediction research (Choi & Lee, 2023).

Table 2. Comparison of Deep Learning Models Used in Drug–Target Interaction Prediction

Model Type	*Main Application in DTI*	*Advantages*	*Limitations*
CNN	*Feature extraction from protein sequences and SMILES data*	*Automatic feature extraction, computational efficiency*	*Limited ability to capture long-range dependencies*
RNN (LSTM/GRU)	*Sequential biological data analysis*	*Effective modeling of temporal and contextual relationships*	*Higher computational cost and training complexity*
GNN	*Molecular graph representation learning*	*Preserves molecular structure and chemical relationships*	*Requires graph-structured data and complex computation*
Transformer	*Learning contextual relationships in biological sequences*	*Strong long-range dependency modeling and attention mechanism*	*Large data and computational resource requirements*

6. STATE-OF-THE-ART DEEP LEARNING MODELS FOR DRUG–TARGET INTERACTION PREDICTION

Recent advances in deep learning have led to the development of several powerful architectures for Drug–Target Interaction (DTI) prediction, which differ in molecular representation strategies, neural network structures, and predictive performance . A comparison of widely used state-of-

the-art DTI prediction models and their reported performances is presented in Table 3 (Shi et al., 2024).

DeepDTA is one of the earliest successful deep learning models for DTI prediction and utilizes two parallel Convolutional Neural Networks (CNNs) to separately process drug SMILES sequences and protein sequences . One of the main advantages of DeepDTA is its ability to automatically learn interaction patterns directly from sequence data without requiring handcrafted features, and the model achieved strong predictive performance on benchmark datasets such as DAVIS and KIBA (Debnath et al., 2025b).

GraphDTA introduced Graph Neural Networks (GNNs) into DTI prediction by representing drug molecules as molecular graphs in which atoms are treated as nodes and chemical bonds as edges .Compared with sequence-based methods, GraphDTA preserves molecular structural information more effectively and generally achieves higher predictive performance than DeepDTA (Y. Zhang et al., 2023).

Another important framework is DeepPurpose, which is a flexible deep learning platform developed for multiple biomedical prediction tasks, including DTI prediction. DeepPurpose supports different architectures such as CNNs, GNNs, and multilayer perceptrons (MLPs), and its modular structure makes it adaptable to different biological datasets and computational drug discovery applications (Huang et al., 2021).

According to the results summarized in Table 3, graph-based models generally demonstrate superior performance because they can better preserve the structural and chemical properties of drug molecules compared with traditional sequence-based approaches (Y. Zhang et al., 2023).

Table 3. Performance Comparison of DTI Models

Model	Dataset	Metric	Performance
DeepDTA	BindingDB	AUC	0.89
GraphDTA	BindingDB	AUC	0.92
DeepPurpose	KIBA	CI	0.85

7. APPLICATIONS OF DEEP LEARNING IN DRUG DISCOVERY

Deep learning has become an important tool in modern drug discovery because of its ability to analyze large-scale biological and chemical data and identify complex molecular relationships. Deep learning-based Drug–Target Interaction (DTI) models are widely applied in cancer research, neurological disorders, infectious diseases, and drug repurposing (H. Chen et al., 2018).

7.1. Cancer Treatment

Deep learning models help identify drugs targeting cancer-related proteins by accurately predicting drug–target interactions. Graph-based approaches such as GraphDTA have shown strong performance in predicting interactions between therapeutic compounds and oncogenic proteins, supporting faster and more cost-effective drug discovery (Debnath et al., 2025b).

7.2. Neurological Disorders

Diseases such as Alzheimer’s and Parkinson’s involve complex biological mechanisms. Deep learning models can identify hidden molecular patterns and predict therapeutic targets associated with neurodegenerative disorders (Vicidomini et al., 2024).

7.3. Infectious Diseases

During the COVID-19 pandemic, deep learning-based DTI models were widely used to identify potential antiviral

compounds targeting viral proteins. These approaches accelerated the prioritization of candidate drugs for experimental validation (Abdel-Basset et al., 2020).

7.4. Drug Repurposing

Drug repurposing involves identifying new therapeutic uses for existing drugs. Deep learning models can analyze large biomedical datasets to discover unknown drug–disease relationships. This approach reduces development costs and time while benefiting from the known safety profiles of approved drugs (Pan et al., 2022).

8. CONCLUSION

Deep learning has become one of the most important approaches in Drug–Target Interaction (DTI) prediction because it can learn complex relationships between drugs and biological targets more effectively than traditional computational methods (Chen et al., 2018; Abbasi et al., 2020). The performance of DTI models strongly depends on data representation techniques, where drug molecules are commonly represented using SMILES sequences, molecular fingerprints, and graph-based structures, while protein targets are encoded through amino acid sequences and structural embeddings (Hu et al., 2024; Karimi et al., 2019; Reiser et al., 2022).

Large-scale datasets such as BindingDB, DrugBank, DAVIS, and KIBA have significantly contributed to the development of deep learning-based DTI prediction by providing valuable experimental interaction data (Chen et al., 2016; Debnath et al., 2025a). However, challenges including noisy data, class imbalance, missing values, and limited interpretability still affect model performance and reliability (Mirza et al., 2019; Alqudah & Moussavi, 2025).

Different deep learning architectures have been widely applied in DTI prediction studies, including CNNs, RNNs, Graph Neural Networks (GNNs), and Transformer-based models (Hu et al., 2024; Barlow et al., 2022; Choi & Lee, 2023). Among these approaches, GNN-based models generally achieve superior performance because they preserve molecular structural information more effectively than sequence-based methods (Zhang et al., 2023; Ru et al., 2025). In addition, hybrid frameworks such as DeepPurpose improve model flexibility and generalization across different datasets (Huang et al., 2021).

Deep learning-based DTI prediction has important applications in cancer treatment, infectious disease research, neurological disorders, and drug repurposing studies (Chen et al., 2018; Pan et al., 2022; Vicidomini et al., 2024). During the COVID-19 pandemic, these models played an important role in accelerating drug repurposing and identifying potential antiviral compounds (Abdel-Basset et al., 2020).

Despite significant progress, limitations such as insufficient labeled data, high computational cost, and lack of explainability remain major research challenges (Alqudah & Moussavi, 2025; Ru et al., 2025). Future studies are expected to focus on explainable artificial intelligence, multimodal learning, self-supervised learning, and multi-omics integration to improve the robustness and interpretability of DTI prediction systems (Alhadethy et al., 2025; Zhang et al., 2025). Overall, deep learning-based DTI prediction is expected to play a central role in precision medicine and modern drug discovery by reducing experimental costs and accelerating the identification of effective drug candidates (Gupta et al., 2021; Chen et al., 2018).

REFRENCES

Abbasi, K., Razzaghi, P., Poso, A., Ghanbari-Ara, S., & Masoudi-Nejad, A. (2020). Deep learning in drug target interaction prediction: Current and future perspectives. *Current Medicinal Chemistry, 28*(11), 2100–2113. https://doi.org/10.2174/0929867327666200907141016

Abdel-Basset, M., Hawash, H., Elhoseny, M., Chakrabortty, R. K., & Ryan, M. (2020). Deeph-DTA: Deep learning for predicting drug-target interactions: A case study of COVID-19 drug repurposing. *IEEE Access, 8*, 170433–170451. https://doi.org/10.1109/ACCESS.2020.3024238

Alhadethy, N. H., Hashim, A. N., & Al-Rashid, S. Z. (2025). Deep learning approaches for drug–target interaction prediction: Comprehensive examination of methods, data resources, challenges, and future. *SUMMA 2025 Proceedings*, 763–772.

Alqudah, A. M., & Moussavi, Z. (2025). Bridging signal intelligence and clinical insight: A comprehensive review of feature engineering, model interpretability, and machine learning in biomedical signal analysis. *Applied Sciences, 15*(22), 12036. https://doi.org/10.3390/APP152212036

Barlow, D., et al. (2022). A novel deep neural network technique for drug–target interaction. *Pharmaceutics, 14*(3), 625. https://doi.org/10.3390/PHARMACEUTICS14030625

Berdigaliyev, N., & Aljofan, M. (2020). An overview of drug discovery and development. *Future Medicinal Chemistry, 12*(10), 939–947. https://doi.org/10.4155/FMC-2019-0307

Chen, H., et al. (2018). The rise of deep learning in drug discovery. *Drug Discovery Today, 23*(6), 1241–1250. https://doi.org/10.1016/J.DRUDIS.2018.01.039

Chen, X., et al. (2016). Drug–target interaction prediction: Databases, web servers and computational models. *Briefings in Bioinformatics, 17*(4), 696–712. https://doi.org/10.1093/BIB/BBV066

Choi, S. R., & Lee, M. (2023). Transformer architecture and attention mechanisms in genome data analysis. *Biology, 12*(7), 1033. https://doi.org/10.3390/BIOLOGY12071033

Davis, A. M., & Leeson, P. D. (2023). Physicochemical properties. In *The Handbook of Medicinal Chemistry*.

Debnath, K., et al. (2025). A survey on deep learning for drug-target binding prediction. *Briefings in Bioinformatics*, *26*(5), 491.

Gupta, R., et al. (2021). Artificial intelligence to deep learning: Machine intelligence approach for drug discovery. *Molecular Diversity, 25*(3), 1315–1360. https://doi.org/10.1007/S11030-021-10217-3

Hu, W., et al. (2024). Deep learning methods for small molecule drug discovery: A survey. *IEEE Transactions on Artificial Intelligence, 5*(2), 459–479. https://doi.org/10.1109/TAI.2023.3251977

Huang, K., et al. (2021). DeepPurpose: A deep learning library for drug–target interaction prediction. *Bioinformatics, 36*(22–23), 5545–5547. https://doi.org/10.1093/BIOINFORMATICS/BTAA1005

Karimi, M., et al. (2019). DeepAffinity: Interpretable deep learning of compound–protein affinity. *Bioinformatics*, *35*(18), 3329–3338. https://doi.org/10.1093/BIOINFORMATICS/BTZ111

Mak, K. K., et al. (2024). Artificial intelligence in drug discovery and development. In *Drug Discovery and Evaluation*.

Mirza, B., et al. (2019). Machine learning and integrative analysis of biomedical big data. *Genes, 10*(2), 87. https://doi.org/10.3390/GENES10020087

Pan, X., et al. (2022). Deep learning for drug repurposing. *Wiley Interdisciplinary Reviews: Computational Molecular Science, 12*(4), e1597.

Panahandeh, F., & Mansouri, N. (2025). A comprehensive review of neural network-based approaches for drug–target interaction prediction. *Molecular Diversity, 30*(2), 1647–1694.

Pande, A., et al. (2019). Computing wide range of protein/peptide features. *BioRxiv*.

Pant, M., et al. (2026). *Machine Learning and Its Applications to Healthcare*. Springer.

Reiser, P., et al. (2022). Graph neural networks for materials science and chemistry. *Communications Materials, 3*(1), 93.

Ru, X., et al. (2025). In silico methods for drug-target interaction prediction. *Cell Reports Methods, 5*(10), 101184.

Salem, A. (2024). AI-enabled system to predict drug-target binding affinity.

Shi, W., et al. (2024). A review of machine learning-based methods for predicting drug–target interactions. *Health Information Science and Systems, 12*(1), 30.

Suruliandi, A., et al. (2024). Drug target interaction prediction using machine learning techniques. *International Journal of Interactive Multimedia and Artificial Intelligence, 8*(6), 86–100.

Vicidomini, C., et al. (2024). Computational methods in drug discovery for neurodegenerative diseases. *Biomolecules, 14*(10), 1330.

Yu, J., et al. (2025). A review of drug-target interaction prediction methods. *CCIS*, 161–195.

Zhang, O., et al. (2025). Graph neural networks in modern AI-aided drug discovery. *Chemical Reviews, 125*(20), 10001–10103.

Zhang, Y., et al. (2023). A survey of drug-target interaction prediction via graph neural networks. *Computers in Biology and Medicine, 163*, 107136.

N-GRAM BASED FEATURE ENGINEERING IN NEWS TEXTS AND ITS EFFECT ON CLASSIFICATION PERFORMANCE

Tolga MAHRAMANLIOĞLU[1]

Zülfikar ASLAN[2]

1. INTRODUCTION

The rapid growth of digital news content has steadily increased the need for efficient and interpretable systems that can automatically classify large volumes of text. Text classification is a core part of many natural language processing tasks such as spam detection, sentiment analysis, and news categorization.

In traditional machine learning approaches, n-gram based feature representation is a critical design choice that directly affects model performance. However, the question of which n-gram granularity gives better results for which dataset and task remains an open one in the literature. Considering the high computational cost, the large labeled data requirement, and the limited interpretability of deep learning models, traditional methods supported by sound feature engineering choices are still a valid option for resource-limited applications where explainability matters.

The main goal of this study is to systematically compare how different n-gram configurations and the related feature engineering choices affect classification performance in news text

[1] Yüksek Lisans Öğrencisi, Gaziantep Islam Science and Technology University, ORCID: 0009-0005-7459-1901.

[2] Assoc. Prof. Dr., Gaziantep Islam Science and Technology University, ORCID: 0000-0002-2706-5715.

classification. In line with this, four research questions are addressed:

- How does classification performance change when uni-gram, bi-gram, and tri-gram features are used alone?
- Do n-gram combinations give significantly better results than single n-gram types?
- To what extent does vectorizing and weighting the title and description fields separately improve overall performance?
- What is the effect of the k value in chi-square feature selection, and which k value gives the best balance?

This study does not only compare accuracy rates but also offers a complete evaluation that includes feature selection sensitivity, hyperparameter optimization, feature importance, and cross-validation analyses. The rest of the paper is organized as follows. Section 2 reviews the related literature. Section 3 explains the dataset, the preprocessing steps, and the feature engineering process in detail. Section 4 presents the experimental results, including sensitivity analyses, per-class performance evaluation, feature importance, and cross-validation findings. The best configuration, Uni+Bi+Tri (1,3), achieved 92.325% test accuracy and a 92.305% F1-Macro score on the AG News dataset. Section 5 concludes with the main findings and directions for future work.

2. LITERATURE REVIEW

N-gram based approaches are among the well-established methods in text classification. Cavnar and Trenkle (1994) carried out one of the early studies that applied n-gram statistics to the language identification problem and showed the language-independent representation capacity of character n-grams.

Joachims (1998) demonstrated the effectiveness of Support Vector Machines on high-dimensional and sparse text vectors using the Reuters dataset; this approach was widely accepted as the industry standard in the following decades.

The TF-IDF weighting proposed by Salton and Buckley (1988) normalizes raw term frequencies by document frequency, which reduces the effect of low-information terms and helps classifiers focus on more discriminative features. Yang and Pedersen (1997) made a wide comparison of different feature selection methods and showed that the chi-square statistic is among the most effective methods for text classification. Wang and Manning (2012) showed that the NB-SVM model, which combines SVM with Naive Bayes weights, provides a strong and interpretable baseline for short text classification.

Zhang et al. (2015), in their work that turned the AG News dataset into a standard benchmark, showed that character-level convolutional neural networks (Char-CNN) provide significant gains over traditional BoW models; the reference accuracy for traditional models was reported as 88–90%. Johnson and Zhang (2017) reached 91.5% accuracy on this dataset with a deep pyramid convolutional neural network architecture. Joulin et al. (2017) showed that the FastText model, which uses subword information, can reach success rates close to deep models in large-scale text classification at a much lower computational cost. When fine-tuned, BERT based models reach 94–95% accuracy on AG News (Devlin et al., 2019).

Sebastiani (2002), in a wide survey of machine learning based text categorization, showed that combining structural metadata features such as title and source information with the raw text increases classification performance. This finding provides the theoretical basis for vectorizing the title field separately in our study.

3. METHOD

This section explains in detail the dataset used, the preprocessing steps applied, the feature engineering strategy, the classifier design, and the evaluation protocol. All experiments were run on Google Colab in a Python 3.x environment using the scikit-learn library.

3.1. Dataset

In this study, the AG News (AG's News Topic Classification Dataset) dataset was used. Made publicly available by Zhang et al. (2015) as a standard benchmark in text classification, this dataset consists of four categories: world news (World), sports (Sports), business (Business), and science/technology (Sci/Tech). The dataset was downloaded programmatically from Kaggle through the kagglehub library.

Table 1. AG News Dataset Properties

Property	Value
Training set	120,000 news articles (30,000 per class)
Test set	20% of the training set — 24,000 samples (hold-out split, 6,000 per class)
Number of classes	4 — fully balanced distribution
Fields	label (0–3), title, description
Language	English
Average token length	37–39 words (the difference between classes is not statistically significant)

The fully balanced structure of the dataset removes the need for class weighting and makes the interpretation of evaluation metrics such as the macro average F1-score easier. The Sci/Tech class showed the highest standard deviation in token length (13.2), which suggests that technology news has a more variable structure in terms of content length.

3.2. Preprocessing

The raw text data was passed through the following steps before being fed into the classification process. As required by the feature engineering strategy explained in Section 3.3, the title and description fields were cleaned and stored separately:

- URL and HTML entity cleaning: patterns such as 'https://' and 'www.', together with HTML character entities like '&#…;' and '&', were replaced with spaces.
- Lowercasing: all text was converted to lowercase.
- Non-alphanumeric character cleaning: all characters except letters and spaces (punctuation, digits, and so on) were removed using regex.
- Whitespace normalization: multiple consecutive spaces were reduced to a single space.

3.3. Feature Engineering

The most critical design decision in this study is how to handle the title and description fields during text representation. In the standard approach, both fields are merged into a single TF-IDF vector; however, this method ignores the different information density and lexical structure of the title and the description. In this study, three separate TF-IDF matrices were built and combined using scipy.sparse.hstack:

Table 2. Hstack Feature Blocks

Block	Source	Analyzer	N-gram	Notes
A	Title	Word	(1,2)	× α weight coefficient
B	Description	Word	(1,2)	Standard TF-IDF
C	Full text (A+B)	Character	(3,5)	analyzer='char_wb'; word boundary padding

In all TF-IDF vectorization steps, the sublinear_tf = True parameter was used; this parameter transforms the TF value as 1

+ log(tf) and prevents high-frequency terms from gaining a disproportionate weight. With the min_df = 2 parameter, only terms that appear in at least two documents were included, so hapax legomena (terms appearing only once) features were filtered out. max_features was set to 100,000 for the word n-gram blocks and 50,000 for the character n-gram block.

The character n-gram block (Block C) was added to address the weakness of word-level representations against misspellings and morphological variations. The 'char_wb' mode applies space padding to the start and end of words; in this way, subword patterns that cross word boundaries can be captured as features.

3.4. Feature Selection: Chi-Square

The size of the raw feature matrix combined by hstack can exceed 250,000. To make this dimensionality manageable and to filter out noisy features, chi-square (χ^2) based feature selection was applied. For each feature, the χ^2 score is a measure of the deviation between observed (O) and expected (E) frequencies and is calculated using the following formula:

$$\chi^2(t, c) = \Sigma \left[(O - E)^2 / E\right]$$

The sklearn.feature_selection.SelectKBest(chi2, k = 30,000) configuration was used. To determine the optimal k value, a systematic sensitivity analysis was carried out from 500 to 30,000 (see Section 4.1).

3.5. Classifier: LinearSVC

LinearSVC (Linear Support Vector Classifier) was chosen as the classifier. LinearSVC, which learns a maximum-margin decision boundary using hinge loss, is especially effective on high-dimensional and sparse text vectors; thanks to the liblinear solver, it converges quickly on large datasets. The one-vs-rest strategy was used for the multi-class setup.

3.6. Hyperparameter Optimization

Two critical hyperparameters were optimized through sequential grid search. In the first stage, $\alpha \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0\}$ values were tested while keeping $C = 1.0$ fixed. In the second stage, the best α value was fixed and a search was carried out over $C \in \{0.01, 0.05, 0.1, 0.3, 0.5, 1.0, 2.0, 5.0\}$.

3.7. N-gram Configurations and Evaluation

For a systematic comparison, six different n-gram ranges were tested: Unigram (1,1), Bigram (2,2), Trigram (3,3), Uni+Bi (1,2), Uni+Bi+Tri (1,3), and Bi+Tri (2,3). The same feature engineering pipeline was applied in all configurations; only the word n-gram range of Block A and Block B was changed. Block C (character n-gram) was kept fixed across all experiments.

Model evaluation was carried out in two stages. In the primary evaluation, 80% of the dataset (96,000 samples) was used for training and 20% (24,000 samples) for testing; the split was made while preserving the class distribution. To measure the generalization capacity, 5-Fold Stratified K-Fold Cross-Validation was applied on a stratified subset of 40,000 samples. The evaluation metrics used were accuracy, per-class precision, recall, and F1-score, together with the macro average F1-score.

4. RESULTS

4.1. Chi-Square K Value Sensitivity Analysis

A separate sensitivity analysis was carried out to determine the optimal k value. In this analysis, the n-gram configuration and other parameters were kept fixed, and only the number of features selected by chi-square was systematically changed from 500 to 30,000. The 91.871% value obtained at k = 30,000 represents the optimal balance between information and

computational cost, and all configuration comparisons were carried out at this value.

Table 3. Chi-square k Value Sensitivity Analysis

k	500	1,000	2,000	5,000	10,000	30,000
Accuracy	85.696%	88.292%	89.704%	91.233%	91.675%	**91.871%**

4.2. Title Weight Coefficient (α) And C Optimization

The α optimization results showed that giving an increasing weight to the title monotonically reduces accuracy, and the highest value (92.208%) was obtained with α = 0.5. This finding suggests that the description field carries a richer semantic signal for classification compared to the title.

Table 4. Title Weight Coefficient (α) Optimization

α	0.5	1.0	1.5	2.0	3.0	4.0
Accuracy	**92.208%**	91.992%	91.688%	91.546%	90.921%	90.188%

After fixing α = 0.5 and applying k = 30,000, the LinearSVC C parameter was optimized. C = 0.3 gave the best result both in accuracy (92.325%) and F1-Macro (92.305%). The performance drop as C goes above 0.3 shows that the tendency toward overfitting becomes clearer in the high-dimensional TF-IDF space.

Table 5. LinearSVC C Parameter Optimization

C	0.01	0.05	0.1	0.3	0.5	1.0	2.0 / 5.0
Accuracy	90.058%	91.588%	91.992%	**92.325%**	92.271%	92.208%	91.858% / 91.271%

4.3. Comparison Of N-gram Configurations

The comparative results of the six configurations run with optimized parameters (α = 0.5, C = 0.3, k = 30,000) are presented below. Figure 1 visualizes the performance difference between configurations and also includes the α optimization curve, the C parameter search, and the gain heatmap.

Table 6. N-gram Configuration Comparison Results (α=0.5, C=0.3, k=30K)

Configuration	Vocab	Accuracy (%)	F1-Macro (%)	Time (s)
Unigram (1,1)	100,000	92.183	92.163	58.6
Bigram (2,2)	100,000	92.062	92.038	67.8
Trigram (3,3)	100,000	91.929	91.909	69.4
Uni+Bi+Tri (1,3)	100,000	**92.325**	**92.305**	70.8
Uni+Bi+Tri (1,3)	100,000	92.308	92.288	80.5
Bi+Tri (2,3)	100,000	92.029	92.006	79.4

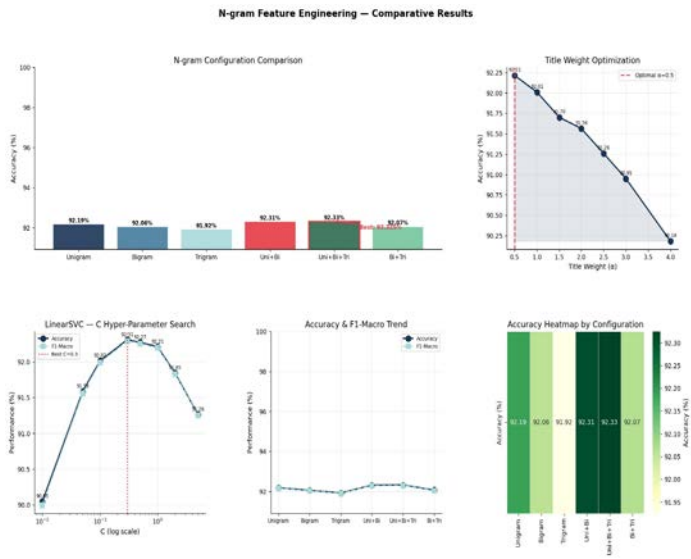


Figure 1. N-gram Configurations, α Optimization, C Parameter Search, and Gain Heatmap

Note. [a] Vocab values belong only to the word n-gram block (Block A and B); the max_features value of the character n-gram block (Block C) was set separately to 50,000.

When Table 6 and Figure 1 are considered together, all configurations cluster within a narrow band between 91.9 and 92.3. This finding suggests that the main drivers of performance are system-level feature engineering decisions (hstack, character n-gram, optimized α and C) rather than individual n-gram selection. By combining uni-gram, bi-gram, and tri-gram

information, Uni+Bi+Tri (1,3) provided the best balance between word-level specificity and broader context cues.

4.4. Per-Class Performance And Confusion Matrix

For the best model (Uni+Bi+Tri, 1,3), the detailed per-class evaluation is presented in Table 7, and the corresponding confusion matrix is presented in Figure 2.

Table 7. Per-Class Classification Report — Uni+Bi+Tri (1,3), Test Set

Class	Precision	Recall	F1-Score	Support
World	0.9380	0.9047	0.9210	6,000
Sports	0.9537	0.9858	**0.9695**	6,000
Business	0.8993	0.8945	0.8969	6,000
Sci/Tech	0.9019	0.9083	0.9051	6,000
Accuracy	—	—	**0.9233**	24,000
Macro Avg.	0.9232	0.9233	**0.9231**	24,000

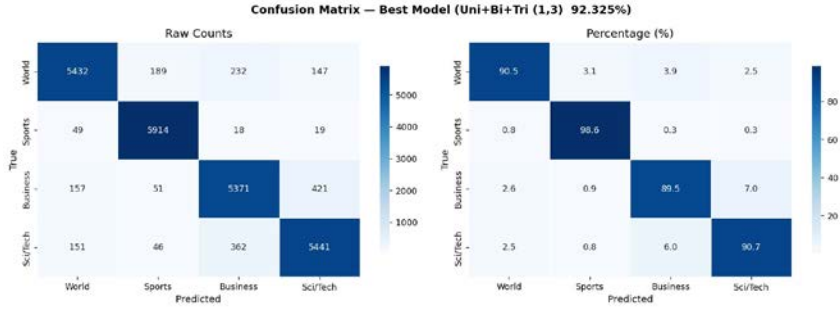


Figure 2. Confusion Matrix — Best Model Uni+Bi+Tri (1,3), 92.325% Accuracy (Raw Counts and Percentages)

The Sports class achieved the best result with 98.58% recall and 96.95% F1-score. The domain-specific rich lexical content of sports news, such as 'cricket', 'football', 'cup', and 'coach', makes this class clearly separable from the others. As seen in the confusion matrix (Figure 2), Sports has only 86 misclassifications, while the Business class incorrectly predicted 421 samples as Sci/Tech. This overlap comes from business news

sometimes containing technology-related content and points to a semantic boundary that the model cannot cross with a bag-of-words representation.

4.5. Feature Importance Analysis

The most determining features for each class were identified through the LinearSVC coefficients. Figure 3 shows the top 15 features per class and the sources of these features (Title, Description, Character).

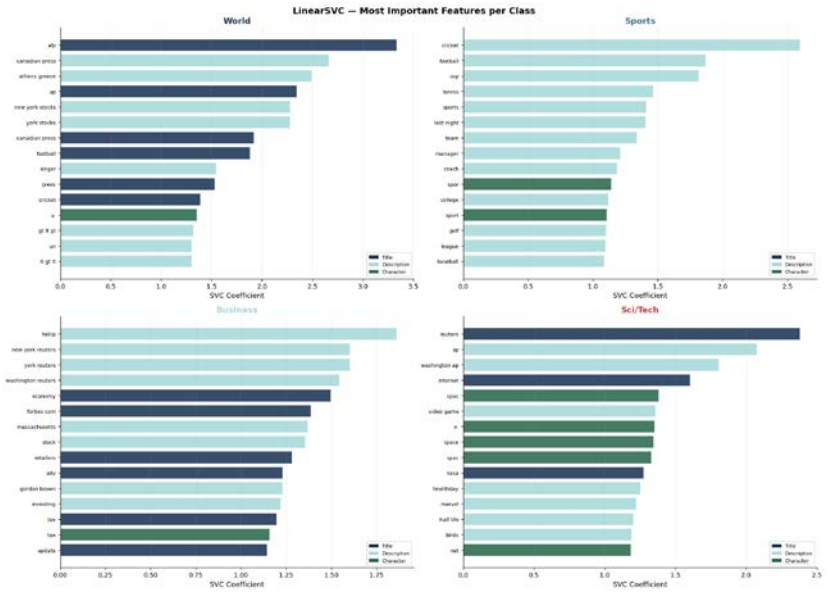


Figure 3. LinearSVC Top 15 Features per Class (Color-Coded by Title / Description / Character Block)

The key findings from Figure 3 are as follows. In the World class, news agency features such as 'york stocks', 'afp', 'ap', and 'canadian press' came to the front; this result shows that agency information is strongly correlated with the geographic and diplomatic news category. In the Sports class, words like 'cricket', 'football', 'cup', and 'tennis' have the highest coefficients as expected, while in the Business class features such as 'york reuters', 'economy', and 'investing' stand out. In the Sci/Tech

class, features like 'reuters', 'internet', 'video game', and 'nasa' were the deciding ones. The fact that features from the character n-gram block appear in every class's list confirms that this block provides complementary subword information.

4.6. 5-Fold Cross-Validation

The results of the 5-Fold Stratified K-Fold Cross-Validation, which was applied to evaluate the generalization capacity of the model, are presented in Table 8.

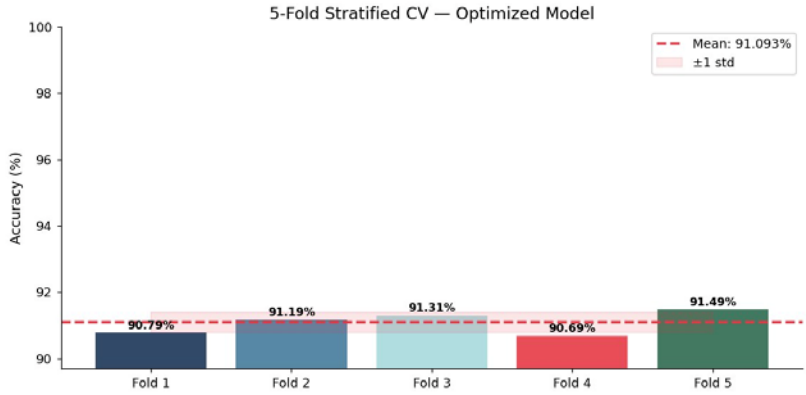


Figure 4. 5-Fold Stratified Cross-Validation — Per-Fold Accuracy Scores

Table 8. 5-Fold Stratified Cross-Validation Results

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean ± Std.
90.79%	91.24%	91.30%	90.66%	91.50%	**91.10% ± 0.32%**

The fact that the standard deviation stays as low as ±0.32 shows that the model behaves stably across different data subsets. The roughly 1.2-point gap between the test set accuracy (92.325%) and the cross-validation mean (91.10%) can be explained by the sampling variance of the 20% test split and is not considered a meaningful sign of overfitting.

5. CONCLUSION

This study examined the systematic effect of n-gram based feature engineering on news text classification using the AG News dataset. The experimental findings clearly show that the main factors driving model performance are system-level feature engineering decisions rather than individual n-gram selection.

The clustering of all configurations in the 91.9–92.3 range supports this observation and shows that the chosen feature engineering strategy overshadows possible gains from n-gram selection. Vectorizing the title and description fields separately, applying an $\alpha = 0.5$ weight coefficient, adding the character n-gram (3–5) block, using the LinearSVC classifier, and the systematic hyperparameter optimization ($C = 0.3$, $k = 30{,}000$) form the main components of this strategy.

The 92.325% test accuracy obtained clearly exceeds the 88–90% reference value reported for traditional bag-of-words models (Zhang et al., 2015). BERT based models report 94–95% accuracy on this dataset (Devlin et al., 2019); the roughly 2-point gap to these models can be considered an acceptable trade-off given the advantages in computational cost, training time, and interpretability.

The main conclusions of the study can be summarized as follows:

- The Uni+Bi+Tri (1,3) n-gram combination forms the optimal structure for this dataset; by combining uni-gram, bi-gram, and tri-gram information, it offers the best balance between word-level specificity and broader context.

- Vectorizing and weighting the title and description fields separately contributes positively to classification performance.
- Character n-gram (3–5) features provide subword information that complements word n-grams and bring particular robustness against morphological variations.
- In chi-square feature selection, k = 30,000 keeps the balance between information and computational cost at its optimum point for AG News.
- LinearSVC, with C = 0.3 regularization, is an effective classifier for high-dimensional and sparse text vectors.
- 5-Fold cross-validation (± 0.32 std.) confirmed that the model is reliable and stable across different data splits.

Future work can move forward in the following directions. Adding metadata features based on named entity recognition (NER), the hybrid use of BERT embedding layers with traditional TF-IDF features, deepening feature interpretability with explainable AI tools such as SHAP and LIME, and cross-comparisons on datasets from different languages and domains stand out as promising research paths in this area.

REFERENCES

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. In Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (pp. 161–175). UNLV Publications.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Vol. 1, pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), Proceedings of the 10th European Conference on Machine Learning (ECML-98) (LNCS 1398, pp. 137–142). Springer. https://doi.org/10.1007/BFb0026683

Johnson, R., & Zhang, T. (2017). Deep pyramid convolutional neural networks for text categorization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1, pp. 562–570). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1052

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Vol. 2, pp. 427–431). Association for Computational Linguistics. https://doi.org/10.18653/v1/E17-2068

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1–47. https://doi.org/10.1145/505282.505283

Wang, S., & Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Vol. 2, pp. 90–94). Association for Computational Linguistics.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In D. H. Fisher (Ed.), Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97) (pp. 412–420). Morgan Kaufmann.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), Advances in Neural Information Processing Systems 28 (NIPS 2015) (pp. 649–657). Curran Associates.

FROM REACTIVE SECURITY TO CONTROLLED OPERATIONS: LEAN SIX SIGMA AND SPC FOR CYBERSECURITY WORKFLOWS

Gulser OZ[1]

Fesih KESKIN[2]

1. INTRODUCTION

Cybersecurity operations often break down at handoff points. A security information and event management (SIEM) or endpoint detection and response (EDR) platform may raise an alert correctly, yet the alert sits in a triage queue. A high-severity incident may be recognized, but containment stalls because no one is sure who can isolate a production system. A vulnerability may be scored and given a deadline while infrastructure, application, and vendor teams remain uncertain about ownership. The weakness in each case is not the security control itself but the process that carries it into daily work. As attack cycles shorten and infrastructure grows more distributed, organizations have less operational margin to absorb these delays, and cloud, IoT, and operational-technology dependencies mean an unclear queue or a missing owner can become real exposure before anyone notices (National Institute of Standards and Technology [NIST], 2018; Pemmasani, Gudepu, Gonugunta, & Jegajothi, 2025; Tariq, Chhetri, Vo, & Abuadbba, 2025; Tokgöz, 2026).

[1] Arş. Gör. Dr., Iğdır Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği, ORCID: 0000-0002-9620-4598.

[2] Dr. Öğr. Üyesi, Iğdır Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği, ORCID: 0000-0002-3798-2912.

Compliance can reinforce the problem when it is treated mainly as evidence production. An organization may show that a control exists, has an owner, and was reviewed for an assessment, none of which reveals whether the work stays stable during an ordinary operating week. In a SOC that distinction surfaces under routine load: analysts juggle alert volume, recurring false positives, repeated enrichment, and movement across disconnected tools. When pressure rises, consistency is the first thing lost. Escalation rules get read differently shift to shift, evidence is recorded unevenly, and containment decisions lean too heavily on whoever happens to be on duty (Dhirani & Newe, 2024; Scholl, 2011; Tariq et al., 2025).

This chapter treats cybersecurity operations as controllable systems rather than collections of tools and policies. Security outcomes emerge from connected workflows such as detection, triage, investigation, containment, recovery, and review. When one stage is unstable, the whole chain weakens even when the formal architecture looks complete (Scholl, 2011). Four ideas are brought together to address this: Lean Six Sigma to make the workflow visible and strip out work that does not reduce risk; statistical process control to separate routine variation from meaningful change; risk frameworks to guide prioritization; and standard work to preserve gains after a project ends (Anand, Ward, & Tatikonda, 2012; Dempsey, Ross, McEvilley, Orebaugh, & Riddle, 2011; Montgomery, 2013; NIST, 2018, 2020).

Systematic mapping work has established that Lean Six Sigma can be applied to cybersecurity operations (Tissir, Lahmadi, & Festor, 2023), building on a longer record in manufacturing and related domains (Antony, Sony, & McDermott, 2022). The aim here is to operationalize it: to show how SOC and incident-response workflows can be mapped as value streams, translated into critical-to-quality metrics,

monitored with SPC, and held in place by risk-aligned standard work. The discussion is conceptual and practitioner-oriented rather than an empirical study of one organization. Its central claim is that repeatable cybersecurity depends not only on tools and documentation but on measurable workflows, controlled variation, and routines that keep working under pressure.

2. BACKGROUND AND CORE CONCEPTS

Lean and Six Sigma address two recurring operational questions in security work. Lean applies when work slows because flow is poorly organized, and in a SOC this should not mean telling analysts to work faster. It means finding where time is lost without any reduction in risk. These losses cluster around handoffs, approval points, tool boundaries, and thin knowledge bases, and removing them shortens response cycles without weakening the investigation (Antony et al., 2022).

Six Sigma addresses variation. A process can look acceptable on a monthly average while still producing a handful of cases that run long, need rework, or stay open past the expected threshold, and in incident response and vulnerability remediation those long-tail cases carry most of the operational risk. The DMAIC cycle (Define, Measure, Analyze, Improve, Control) gives teams a structured way to trace that variation to its causes, such as inconsistent severity rules, differences in analyst judgment, tool latency, missing asset context, or escalation paths that become inconsistent under pressure. Read this way, Lean Six Sigma is not another administrative layer. It makes work faster where it is needlessly slow and more consistent where it is needlessly uneven (Antony et al., 2022; Prado, Barbosa, & Fernandes, 2024; Tissir et al., 2023; Tokgöz, 2026).

Improvement only becomes useful once broad goals are translated into critical-to-quality (CTQ) characteristics

observable in daily work. A goal such as "reduce cyber risk" tells a shift lead nothing. Measures such as time to detect, time to triage, time to contain, false-positive rate, evidence completeness, patch latency, or the share of cases closed within an agreed threshold tell them where to look. These CTQs tie stakeholder expectations to the parts of the workflow teams can actually move, and they narrow the gap between compliance reporting and operational control. A metric produced only for an audit confirms that something happened. A metric watched while the work is still live can guide decisions and reveal drift (Dempsey et al., 2011; Dhirani & Newe, 2024).

Measurement quality is therefore central. Small definitional choices change what the data mean. A response-time metric shifts depending on whether the clock starts at alert generation, ticket creation, analyst acceptance, or containment. Severity ratings vary between analysts, and exceptions are logged carefully in one team and loosely in another. Left uncontrolled, these differences let a change in labeling masquerade as a real gain. That risk sharpens when AI-supported tools enter the workflow, since models trained on inconsistent labels and incomplete telemetry tend to reproduce the weaknesses of the measurement system rather than correct them (Fu, Mostafa, Iosifidis, & Nair, 2025). Useful metrics need clear definitions and meaningful segmentation, with results examined by severity, asset criticality, environment, attack type, and ownership boundary, so risk shows up where it concentrates instead of hiding inside an average (NIST, 2018).

3. LEAN SIX SIGMA IN SECURITY OPERATIONS

Lean Six Sigma becomes practical in cybersecurity once security work is made visible as an end-to-end value stream. A

SOC incident is not just an alert, a ticket, or a containment decision. It is a sequence of work that moves through tools, teams, approval points, evidence requirements, and risk decisions before it closes. SIPOC mapping gives a first view of that sequence by naming suppliers, inputs, process steps, outputs, and customers. In cybersecurity these categories run wider than in a conventional service process. Adapting SIPOC to security operations, suppliers include IT teams, application owners, managed security service providers, vendors, and threat intelligence feeds, with adversarial activity treated as an external trigger that initiates the stream rather than a conventional supplier. Customers include system owners, executives, auditors, regulators, and affected users.

Value stream mapping extends the SIPOC view by showing where work waits, repeats, or moves without adding security value. This matters in incident response because elapsed time usually creates more risk than analyst effort does. A containment decision may take minutes, but the incident may have waited far longer for ownership, enrichment, approval, or access. A SOC value stream typically runs from intake through triage, investigation, containment, eradication, recovery, and review, and each step connects to critical-to-quality characteristics such as time to triage, time to contain, diagnostic accuracy, evidence completeness, and closure reliability (Lameijer, Den Heijer, & Wang, 2024; Tissir et al., 2023). In an illustrative incident stream, value-added work might total around 78 minutes against roughly 240 minutes of waiting, giving a total lead time near 318 minutes and a process cycle efficiency of about 25 percent. Most of the incident lifetime is therefore not active investigation but queue time and handoff delay. The same logic governs vulnerability management, where the costly delay is rarely discovery itself but the gap between discovery, ownership assignment, remediation, and verification. Mapping helps just as much in cloud and hybrid environments, where responsibility is

split among platform teams, application owners, vendors, and security teams (Dhirani & Newe, 2024; NIST, 2018, 2020).

With the workflow visible, waste becomes a process problem rather than a personal failing. The recurring forms are waiting (cases sitting for access, approvals, enrichment, or ownership decisions), handoff loss (context rewritten or dropped as evidence moves between tools), rework (the same false positive investigated repeatedly), overprocessing (reports that satisfy a request but do not control the shift), and underused expertise (skilled analysts absorbed in routine copying that could be automated once the process is stable). Together these stretch lead time, drive fatigue, and make response quality hinge on the analyst, shift, or context available now (Antony et al., 2022; Lameijer et al., 2024; Morato & Ferreira, 2024; NIST, 2025; Tariq et al., 2025).

DMAIC turns the map into structured improvement. Define ties the project to a specific risk scenario, such as delayed containment of severity-one incidents or excessive delay between vulnerability discovery and remediation, rather than a vague "improve SOC performance." Measure builds a trustworthy baseline by fixing timestamp rules, severity definitions, inclusion criteria, and exception handling before the data are read, since an apparent gain can easily be a relabeling. Analyze tests what drives delay or variation, from enrichment outages and rule deployments to approval paths, staffing, and shifts in incident mix. Improve matches countermeasures to verified causes rather than symptoms, including triage templates, alert tuning, pre-approved containment for high-confidence signatures, better enrichment, stronger runbooks, or targeted automation. Control protects the gains through standard work, monitoring, and out-of-control action plans that tell analysts what to check first when performance drifts (Anand, Ward, & Tatikonda, 2012; Dempsey et al., 2011; Montgomery, 2013; NIST, 2018).

Project selection decides whether Lean Six Sigma becomes a useful discipline or more paperwork. Projects should be chosen because they reduce operational risk, not because they are easy to measure, and the strongest candidates are high-value assets, critical business processes, and workflows where delay or inconsistency widens exposure. Because security value streams cross team boundaries, sponsorship is essential. Without shared accountability, a bottleneck just gets better documented instead of removed. With it, Lean Six Sigma works as a practical implementation layer for security frameworks, translating requirements into measurable targets, control plans, and repeatable routines (NIST, 2018, 2020; Channappagoudar, Lakshmana Gowda, & Thimmappa, 2025).

The link between improvement work and governance should be explicit, with each DMAIC phase producing both an operational output and a risk-management artifact. Table 1 summarizes how the phases connect to governance, risk-based prioritization, continuous monitoring, and standard work (Anand et al., 2012; Dempsey et al., 2011; Montgomery, 2013; NIST, 2018).

Table 1. Alignment of DMAIC phases with risk-management and standard-work outputs

DMAIC	Cybersecurity	Risk-management	Standard work output
Define	Select SOC, incident-response, or vulnerability workflow	Link the project to high-value assets, threat scenarios, and business impact	Project charter and process boundary
Measure	Define CTQs and collect baseline data	Measure exposure, control performance, and evidence quality	Metric dictionary and data collection rules
Analyze	Identify bottlenecks and variation drivers	Prioritize causes by risk severity and operational impact	Root-cause record and Pareto analysis
Improve	Test countermeasures	Reduce likelihood, impact, exposure time, or uncertainty	Updated runbooks and escalation rules
Control	Monitor stability and sustain gains	Support continuous monitoring and ongoing authorization	Control plan, OCAP, and review cadence

4. STATISTICAL PROCESS CONTROL FOR SECURITY METRICS

Statistical process control matters in cybersecurity because not every change in a metric calls for the same response. A SOC may see more alerts early in the week, slower triage during a staffing gap, or a brief rise in false positives after a detection rule changes. Some of that movement is routine. Some of it signals that the operating environment has shifted. SPC separates common-cause variation, the normal behavior of a stable process, from special-cause variation tied to identifiable events such as enrichment outages, major rule deployments, analyst shortages, tooling failures, or new attack campaigns. Treating every routine fluctuation as a special cause invites tampering, where thresholds are adjusted too often, work is rerouted needlessly, and rules are retuned before the process settles. The result is often more variation, not less (Montgomery, 2013; Wankhede, Joshi, & Jain, 2025; Wheeler & Chambers, 2010).

Security dashboards usually miss this distinction. Raw counts and averages are easy to display but say nothing about whether a value is unusual for the process that produced it. A daily alert count can look high simply because the process is naturally volatile, and a containment-time average can look fine while a few severity-one cases drift into the upper tail. Control charts add the missing context by showing the process center, the expected variation, and the decision rules. They do not replace analyst judgment, but they make escalation more disciplined by marking when a change actually deserves investigation (Montgomery, 2013).

Chart choice follows the data type. I-MR charts suit incident-level durations such as time to triage, contain, and patch, especially when cases arrive one at a time. The $\bar{x}$–R and $\bar{x}$–s charts

fit continuous measurements that can be grouped into rational subgroups, such as observations by shift, queue, or service category. The p chart tracks proportions, including the share of incidents that miss an SLA or the share of alerts confirmed as false positives, while c and u charts handle counts such as alerts per unit of traffic. For rare events such as high-severity containment failures, a time-between-events chart is more informative than a count chart dominated by zeros (Montgomery, 2013; Wheeler & Chambers, 2010).

A control chart is only useful once the organization has agreed how to respond to its signals. Signal rules should be set before the chart goes live, and the first response should not depend on who happens to be on shift. Annotations are especially valuable in security operations, since rule changes, staffing changes, detection deployments, outages, and incident surges all explain later movement. Over time the chart becomes an operational record linking process behavior to changes in the environment. An out-of-control action plan should name the first checks to run when a signal appears, among them tooling health, recent detection changes, staffing coverage, queue backlog, incident mix, and external threat activity (Dempsey et al., 2011; Montgomery, 2013).

The notation below summarizes the SPC and capability measures used in this chapter. One-sided requirements are common in security operations, because the practical question is usually whether a time-based metric stays below an upper threshold. For example, a response-time requirement may be expressed as response time $\leq$ USL, where USL is the upper specification limit and LSL is the lower specification limit.

$$\mathrm{CL} = \bar{\mathrm{x}}$$

$$\mathrm{UCL} = \bar{\mathrm{x}} + 3\hat{\sigma}\,,\ \mathrm{LCL} = \bar{\mathrm{x}} - 3\hat{\sigma}$$

$$C_p = \frac{USL - LSL}{6\hat{\sigma}}$$

$$C_{pk} = \min\left\{\frac{USL - \mu}{3\hat{\sigma}}, \frac{\mu - LSL}{3\hat{\sigma}}\right\}$$

$$C_{pk}^{USL} = \frac{USL - \mu}{3\hat{\sigma}}$$

$$PCE = \frac{\text{Value-Added Time}}{\text{Total Lead Time}}$$

Here μ is the process mean, estimated in practice by $\bar{x}$. For individuals and moving-range charts, the process standard deviation is usually estimated from the average moving range,

$$\hat{\sigma} = \frac{\overline{MR}}{d_2}$$

where $\overline{MR}$ is the mean of the moving ranges $MR_i = |x_i - x_{i-1}|$ and (d_2 = 1.128) for moving ranges based on two consecutive observations. The moving-range chart uses

$$UCL_{MR} = D_4\overline{MR}$$

where D_4 = 3.267. The lower control limit for the moving-range chart is fixed at zero because moving ranges cannot be negative.

These limits describe expected variation and flag special-cause signals that warrant operational investigation (Montgomery, 2013; Wheeler & Chambers, 2010).

Capability indices such as C_p and C_{pk} need careful interpretation in security settings. Both are normal-theory measures that assume approximately normal output (Kane, 1986; Montgomery, 2013), an assumption that is often weak for incident-response and containment times. Many cases close quickly while a smaller number stay open far longer because of approval delays, missing access, unavailable system owners, unclear escalation paths, or vendor dependencies. Those long-tail cases are where operational risk concentrates, so a normal-theory capability value can make a process look more capable than it is.

For one-sided requirements such as response time $\leq$ USL, capability assessment should not rest on a single C_{pk} value. SLA pass rates, upper-tail percentiles, transformed data, and distributional models such as lognormal fits are more dependable when the data are strongly skewed (Montgomery, 2013; Thurgood & Ferguson, 2018). Reporting the average response time alone is also too thin. A process can be statistically stable and still miss the required service level, and it can meet the SLA on average while still producing delayed high-impact incidents that create unacceptable exposure.

Control limits and specification limits should stay separate. Control limits describe how the process is expected to behave given its observed variation. Specification limits describe external requirements such as SLA thresholds, regulatory expectations, or risk-tolerance levels. An out-of-control signal and an SLA breach are related but not the same event. A process can stay inside its control limits while consistently missing the service level, and it can stay inside the SLA while showing a special-cause signal worth investigating. The distinction holds

because SPC asks whether the process is behaving unusually, while SLA and risk thresholds ask whether its performance is acceptable (Dempsey et al., 2011; Montgomery, 2013; Wheeler & Chambers, 2010).

Figure 1 illustrates the skewness problem. In the illustrative severity-one (Sev1) response-time sample, both the histogram and the normal probability plot show a long upper tail, which supports using upper-tail percentiles, transformations, or non-normal capability methods rather than normal-theory indices alone (Montgomery, 2013; Thurgood & Ferguson, 2018).

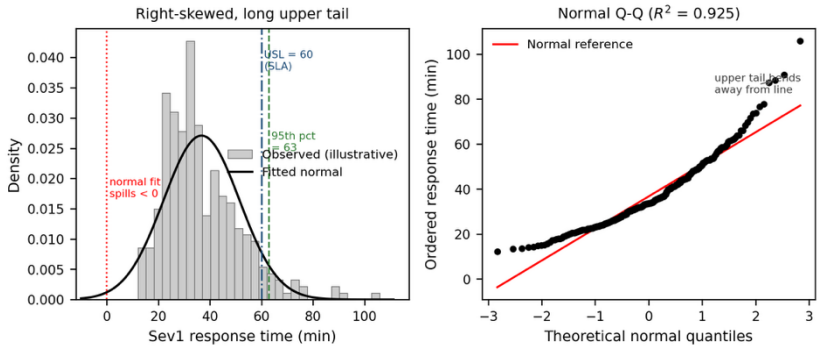


Figure 1. Distribution of severity-one response time using illustrative data (n=300). The histogram and normal probability plot indicate right-skewed response-time behavior, supporting the use of upper-tail or non-normal capability analysis.

5. IMPLEMENTATION PLAYBOOK AND ILLUSTRATIVE SCENARIO

The following walkthrough is an illustrative scenario rather than a study of a real SOC; no actual incident dataset is analyzed, and the steps describe how a team would proceed. Standard work helps keep an improvement from fading once the project ends. In security operations that means making the repeatable parts of the work clear enough that they no longer

depend on memory, habit, or whoever is on shift. Triage templates, evidence checklists, escalation criteria, pre-approved containment actions, post-incident review forms, and out-of-control action plans all serve this purpose. The point is not to remove analyst judgment. It is to cut avoidable variation in routine decisions so experienced analysts can spend their attention on ambiguous incidents, high-impact systems, and cases where business context changes the response (Dempsey et al., 2011; Montgomery, 2013).

This has a boundary. Standard work should stabilize repeatable tasks such as evidence collection, escalation criteria, timestamp rules, and review steps, while leaving novel or high-impact cases to expert judgment.

The playbook and case below use illustrative rather than real data, showing how Lean Six Sigma and SPC translate into security operations without standing in for measurements from any specific organization. A practical rollout can follow a 30/60/90-day plan: establishing governance and mapping a high-pain value stream in the first month, validating definitions and setting a baseline in the second (with process mining surfacing rework loops and queue delays that ticket summaries hide), then deploying countermeasures, control charts, and out-of-control action plans in the third (Anand et al., 2012; Dempsey et al., 2011; Montgomery, 2013; Prado, Barbosa, & Fernandes, 2024). Table 2 is deliberately small. SOCs usually have plenty of dashboards already, but not every displayed metric helps control the work. The aim is a few CTQ areas that expose delay, rework, exposure, and stability, keeping the measurement system practical instead of turning improvement into one more reporting burden.

Table 2. Minimal viable measurement system for security operations.

CTQ Area	Example Metric	Why It Matters	Typical Monitoring Approach
Response timeliness	Time from alert creation to approved containment action	Exposure window; tail cases concentrate risk	I-MR chart for incident-level response times
Detection quality	False positive rate	Rework and overload; reducing it increases capacity	p chart for false positives among investigations
Vulnerability exposure	Time to patch; inventory coverage	Measures exposure and reveals waiting/handoffs	I chart for time-to-patch; p chart for coverage
Stability and capability	% within SLA; capability vs. target	Separates stable-but-incapable from unstable processes	Capability review plus SPC on CTQ

Building on Table 2, the illustrative case targets variation in severity-one response time and is structured around DMAIC. Define frames the problem as excessive variation in that response time and sets a one-sided requirement, response time $\leq$ USL, where the upper specification limit is an agreed SLA or risk-based threshold. In this scenario, response time is defined from alert creation to approved containment action. The project boundary spans intake, triage, investigation, containment, and review, since delay at any step extends the exposure window, which keeps the project tied to the full value stream rather than one isolated activity (Anand et al., 2012; NIST, 2018).

Measure collects several months of incident records and validates system-generated timestamps before any analysis. The team fixes when the response clock starts and stops, which incidents count, how severity-one cases are classified, and how exceptions are handled. These are not administrative details. A response-time improvement means nothing if the measurement rule shifts mid-project, and a disciplined measurement system keeps inconsistent labels, missing timestamps, or altered workflow rules from being mistaken for real gains (Dempsey et al., 2011; Montgomery, 2013).

Analyze segments response times by shift, analyst tier, incident subtype, tooling dependency, approval path, and asset criticality to locate the source of long-tail delay. Delayed containment might trace to missing access on night shifts, waiting on system-owner approval, enrichment outages, unclear escalation criteria, or a vendor dependency. The goal is to move past individual late incidents toward verified causes of delay and variation (Montgomery, 2013; Tissir et al., 2023; Wheeler & Chambers, 2010).

Improve matches countermeasures to those validated bottlenecks, among them pre-approved containment for high-confidence signatures, clearer triage templates, better enrichment, escalation rules, alert tuning, and runbooks that specify eligibility, documentation, and rollback conditions. Changes should be tested in a controlled way, especially when several factors interact, or the team may cut one delay while opening a new risk elsewhere in the response process (Anand et al., 2012; Tissir et al., 2023).

Control monitors incident-level response times with I-MR charts. An out-of-control action plan names the first checks to run when a signal appears, including tooling health, recent rule or workflow changes, staffing coverage, incident mix, queue backlog, and dependency failures, and the phase updates standard work so successful changes become routine. The aim is for future deviations to trigger consistent investigation rather than ad hoc reaction. Figure 2 supports this phase with an I-MR chart tracking both incident-level response times and the short-term moving ranges between consecutive cases (Dempsey et al., 2011; Montgomery, 2013; Wheeler & Chambers, 2010).

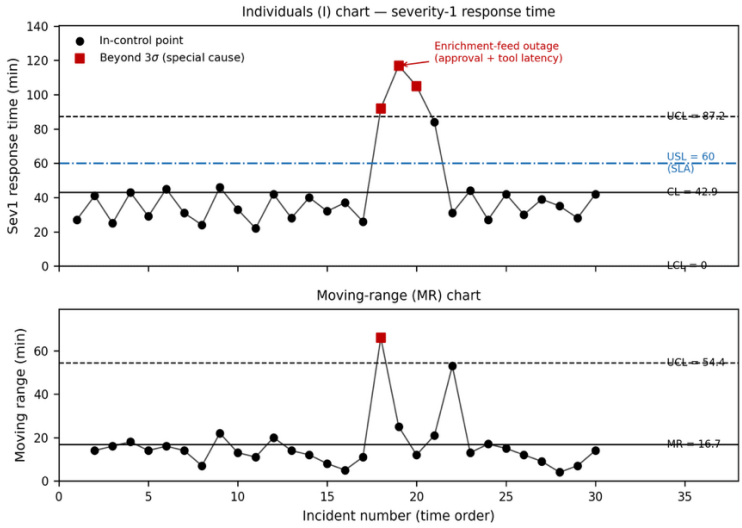


Figure 2. I-MR chart for severity-one response time using illustrative incident-level data.

Figure 2 can be read as a short operational example. Using the first seventeen stable incidents as the baseline, response times remain between roughly 22 and 46 minutes. The center line is the mean response time, $\bar{x}$ = 42.9 minutes. Short-term variation comes from the average moving range of consecutive cases, $\overline{MR}$ = 16.7 minutes, which gives $\hat{\sigma} = \overline{MR} / d_2 = 16.7 / 1.128 \approx 14.8$ minutes. The individuals limit then follow as $UCL = \bar{x} + 3\hat{\sigma} \approx$ 87.2 minutes The calculated lower control limit is negative, so the plotted lower limit is set to zero for practical interpretation because response time cannot be negative. The moving-range chart uses $UCL_{MR} = D_4 \cdot \overline{MR} = 3.267 \times 16.7 \approx 54.4$ minutes. During an enrichment-feed outage, incidents 18, 19, and 20 rise to 92, 117, and 105 minutes, each beyond the upper individuals limit and together forming a clear special-cause signal. Incident 21, at 84 minutes, then falls back just below the upper control limit while still breaching the SLA. On the moving-range chart, the abrupt jump into the outage produces a single point beyond UCL_{MR}, at incident 18, confirming that the shift, not routine

noise, drove the excursion. This is the pattern that should trigger the out-of-control action plan, and in this illustrative stream the first check, recent tooling and enrichment health, locates the cause directly.

The same example shows why control limits and specification limits must be read apart. The agreed SLA threshold here is USL = 60 minutes, well inside the upper control limit of 87.2 minutes. A response time of 70 minutes would therefore fall within the control limits, meaning the process is behaving as expected, while still breaching the SLA. The control chart asks whether the process is stable, and the specification asks whether its output is acceptable. A severity-one stream that routinely lands between 60 and 87 minutes is statistically in control and operationally non-compliant at once, exactly the situation a lone average or a single capability index would conceal.

Several limitations define the scope of this chapter. The value stream example, response-time distribution, and I-MR chart are illustrative rather than empirical, so the numbers should not be read as benchmarks. A full validation would require applying the framework in a live SOC and testing whether variation narrows, SLA performance improves, and the control plan remains usable under pressure. The capability discussion also has a technical boundary: normal-theory indices such as C_p and C_{pk} are weak for right-skewed response-time data, and no single capability measure fully summarizes one-sided, long-tailed security metrics. Finally, the framework assumes reliable timestamps, stable severity definitions, and usable asset context, which many SOCs must first improve.

6. CONCLUSION

Cybersecurity becomes easier to sustain when it is managed as operational work rather than treated as a one-time

architecture decision. Tools, policies, and frameworks are necessary but do not guarantee steady execution. Alerts still need triage, incidents need investigation, containment needs approval, evidence needs recording, vulnerabilities need assignment, and controls need review. When these activities vary widely by shift, analyst, system owner, or tool boundary, the program stays fragile even where the formal design looks complete. The problem is operational as much as technical, and controls have to function as measurable countermeasures that can be tested, monitored, and improved when they fall short (Scholl, 2011).

Lean Six Sigma gives this work a practical structure. DMAIC makes workflows visible, removes avoidable waste, reduces rework, and narrows the variation behind missed detections, delayed containment, and inconsistent assurance (Tissir et al., 2023). SPC adds discipline by helping teams tell routine fluctuation from signals worth investigating, without which organizations either chase noise or miss early signs of decay. The longer-term aim is security processes that are both stable and capable, predictable enough in behavior and capable enough to meet SLAs, risk tolerances, and assurance expectations (Dempsey et al., 2011; Montgomery, 2013; NIST, 2018, 2020; Tokgöz, 2026).

Automation reduces repetitive effort but can also hide weak assumptions, poor labels, unclear ownership, and unstable decision logic, since models trained on inconsistent data reproduce its weaknesses rather than correct them (Fu et al., 2025). Accordingly, models and orchestrated workflows should be treated as operational components subject to measurement and risk management: they need defined CTQs, monitoring, drift checks, escalation rules, and safeguards for high-impact decisions (Fu et al., 2025; National Institute of Standards and Technology, 2023). The central lesson is that cybersecurity improvement is sustainable only while risk governance and operational control

stay connected. Hold that connection, and security work grows more measurable, more repeatable, and more resilient under pressure.

REFERENCES

Anand, S., Ward, P. T., & Tatikonda, M. V. (2012). Role of explicit and tacit knowledge in Six Sigma projects: An empirical examination of differential project success. Journal of Operations Management, 30(3), 158–171.

Antony, J., Sony, M., & McDermott, O. (2022). A systematic review of Lean Six Sigma for the manufacturing industry: Lessons learned and future research directions. International Journal of Quality & Reliability Management, 39(10), 2259–2281.

Channappagoudar, P. P., Lakshmana Gowda, N., & Thimmappa, T. (2025). Integrating Industry 4.0 tools in Six Sigma DMAIC framework: A theoretical roadmap. Processes, 13(3), Article 519.

Dempsey, K. L., Ross, R. S., McEvilley, M., Orebaugh, A. D., & Riddle, M. (2011). Information security continuous monitoring (ISCM) for federal information systems and organizations (NIST Special Publication 800-137). Gaithersburg, MD: National Institute of Standards and Technology. doi:10.6028/NIST.SP.800-137

Dhirani, L. L., & Newe, T. (2024). Enhancing cybersecurity assurance through Lean Six Sigma. Procedia Computer Science, 241, 510–519.

Fu, X., Mostafa, A., Iosifidis, V., & Nair, V. (2025). Mislabeling in cybersecurity vulnerability datasets: An investigation and mitigation study. ACM Transactions on Software Engineering and Methodology, 34(3), Article 115. doi:10.1145/3728800

Kane, V. E. (1986). Process capability indices. Journal of Quality Technology, 18(1), 41–52.

Lameijer, J. E., Den Heijer, A. C., & Wang, X. (2024). Lean and Agile in software development: A systematic literature review. Software Quality Journal, 32(1), 73–106.

Montgomery, D. C. (2013). Introduction to statistical quality control (7th ed.). Hoboken, NJ: Wiley.

Morato, M. L., & Ferreira, K. A. (2024). Value stream mapping application for construction industry loss and waste reduction: A systematic literature review. International Journal of Lean Six Sigma, 15(4), 817–837.

National Institute of Standards and Technology. (2018). Risk management framework for information systems and organizations: A system life cycle approach for security and privacy (NIST Special Publication 800-37 Rev. 2). Gaithersburg, MD: Author. doi:10.6028/NIST.SP.800-37r2

National Institute of Standards and Technology. (2020). Security and privacy controls for information systems and organizations (NIST Special Publication 800-53 Rev. 5). Gaithersburg, MD: Author. doi:10.6028/NIST.SP.800-53r5

National Institute of Standards and Technology. (2023). Artificial intelligence risk management framework (AI RMF 1.0) (NIST AI 100-1). Gaithersburg, MD: Author. doi:10.6028/NIST.AI.100-1

National Institute of Standards and Technology. (2025). Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile (NIST Special Publication 800-61 Rev. 3). Gaithersburg, MD: Author. doi:10.6028/NIST.SP.800-61r3

Pemmasani, P. K., Gudepu, B. K., Gonugunta, K. C., & Jegajothi, B. (2025). Unified AI command console for cybersecurity: Multi-AI integration with minimal manual intervention. In 2025 IEEE Madhya Pradesh Section Conference (MPCON) (pp. 1066–1074). Piscataway, NJ: IEEE.

Prado, C. A., Barbosa, E. F., & Fernandes, A. A. (2024). Process mining in cybersecurity: A systematic literature review. Computers & Security, 139, Article 103700.

Scholl, F. (2011, May). A lean approach to information security. ISSA Journal, 20–24.

Tariq, S., Chhetri, M. B., Vo, B. Q., & Abuadbba, A. (2025). Alert fatigue in security operations centers: A systematic literature review. ACM Computing Surveys, 57(4), Article 93. doi:10.1145/3719758

Thurgood, A., & Ferguson, J. (2018). Process capability analysis for non-normal distributions: A review and practical guidance. Quality Engineering, 30(2), 238–253.

Tissir, F., Lahmadi, A., & Festor, O. (2023). Lean Six Sigma for cybersecurity operations: A systematic mapping study. Computers & Security, 125, Article 102999.

Tokgöz, E. (2026). Six Sigma for continuous improvement in cybersecurity. Cham, Switzerland: Springer Nature.

Wankhede, A. S., Joshi, A., & Jain, S. (2025). Statistical process control techniques for cybersecurity risk monitoring: A survey. International Journal of Quality & Reliability Management, 42(5), 1290–1312.

Wheeler, D. J., & Chambers, D. S. (2010). Understanding statistical process control (3rd ed.). Knoxville, TN: SPC Press.

PREDICTING STUDENT STRESS LEVELS USING MACHINE LEARNING: A COMPARATIVE ANALYSIS OF SVM, MLP, AND RANDOM FOREST

Rıfat AŞLIYAN[1]

1. INTRODUCTION

Academic stress has arisen as a major worldwide issue influencing the students' mental health, psychological quality of life, and higher education students' academic success. Transitioning to universities introduces many problems such as economic difficulties, social pressure, substantial coursework, and unpredictable aspects. Current worldwide data demonstrate that more than 60% of undergraduates report experiencing intense anxiety, chronic stress, or depressive symptoms during their university career. This problem is beyond mental boundaries. It expresses itself biologically through physical symptoms such as poor sleep quality, recurrent headaches, elevated heart rate, and blood pressure volatility. Moreover, community interactions and external factors such as bullying, social pressure from peers, and social isolation heavily intensify the adverse effects of student anxiety. Conventional methods to managing student wellness commonly lack efficacy since they depend on inconsistent subjective data or postponed educational support. Hence, it is imperative for preventative, data-centric applications designed to evaluate diverse variables to identify and categorize academic pressure levels early (Pascoe et al., 2020; Bayram & Bilgel, 2008;

[1] Dr. Öğr. Üyesi, Aydın Adnan Menderes Üniversitesi, Fen Fakültesi, Matematik, ORCID: 0000-0003-1495-713X.

Gollust et al., 2007; Hysenbegasi et al., 2005; Amaral et al., 2017; Walburg, 2014).

Recently, the merging of psychology, computer science, and healthcare has accelerated the utilization of ML methodologies to simulate sophisticated human-centric behaviors. Compared to traditional statistical analysis, ML models can successfully identify non-linear patterns across heterogeneous data containing mental, biological, educational, and environmental features. Researchers have progressively adopted standardized datasets to recognize stress levels can correctly detect indicators of distress before progressing into critical mental health issues. Nevertheless, numerous current papers depend exclusively on narrow parameters such as focusing exclusively on academic performance or biological indicators thereby overlooking the comprehensive reality of daily routines of students. To overcome this problems, this study utilizes a multidimensional dataset which merges social factors, physiological metrics, psychological indicators, physical surroundings, and scholastic strains. By analyzing these features, it can be designed more stable and general prediction models.

This work conducts an evaluation of three machine learning methods to reach superior classification efficiency. The method of Support Vector Machines (SVM) is a powerful supervised classification method that is an efficient in high-dimensional datasets. SVM involves an ideal decision boundary which widens the margin between distinct stress categories (Cortes & Vapnik, 1995). Multilayer Perceptron (MLP) is an Artificial Neural Networks (ANN) consists of layers as an input, hidden, and an output. MLP is equipped to model complex, interconnected interactions within diverse stress features by adjusting weights (Rumelhart et al., 1986). Random Forest (RF) is an ensemble technique that generates many decision trees for training. By merging the estimations of these trees through

majority voting, RF minimizes overfitting, processes without difficulty varying data formats, and offers direct feature rankings (Breiman, 2001).

Recent studies on the classification and detection of student stress levels have been summarized below: Rani et al. (2025) uses the CatBoost method to develop a model that categorize stress levels. Achieving an accuracy of 85% the system shows high reliability and detects weak time management and poor sleep quality for the primary stress predictors. Sulistya et al. (2025) utilize both Gaussian Mixture Models and Principal Component Analysis to segment student stress into four distinct, multidimensional profiles, offering crucial insights. Yamasari et al. (2024) have developed an ANN method optimized with SMOTE-N oversampling to categorize student stress levels, showing the performance testing and ablation studies. Zahra & Kalifia, (2025) propose "StressPredict", a web-based system with Random Forest method to categorize university student stress into five distinct levels. The model provides a high accuracy of 87.93%, proving its effectiveness for early screening in higher education.

In this study, many Random Forest, SVM and MLP machine learning models have been implemented to predict student stress levels. The student stress level dataset includes three levels as 0 (Eustress), 1 (Distress), and 2 (No Stress). The stress level dataset has been partitioned into train and test sets. The dataset consists of 20 features and 1100 samples from college students. The machine learning models have been implemented with their parameters which have been decided by Grid Search to detect the best parameters. To compare each other, the performances of these models have been evaluated with ROC curves, Recall, F1-Score, Precision, and Accuracy.

2. MATERIALS AND METHODS

SVM, MLP, and RF machine learning systems have been developed to decide the stress levels of college students in this work. The systems try to predict the student stress level using key features about stress. This section provides details about the dataset and methods as SVM, MLP, and RF.

2.1. Student Stress Level Dataset

This dataset (Student Stress Level Dataset, 2026) investigates the root causes and multi-faceted impacts of stress among higher education students, capturing survey responses from the college students aged 18-21 via Google Forms. This dataset includes 20 features structured with five dimensions. These features are psychological such as depression, anxiety; physiological such as headaches, sleep quality; environmental such as living conditions, safety; academic such as study load, teacher-student relationships; and social factors such as peer pressure, bullying. The target feature categorizes the student stress levels into three distinct classes as “Eustress”, “Distress”, and “No Stress”.

As displayed in Table 1, out of total 1100 samples in the stress level dataset, 880 have been utilized for train and 220 for test. Whereas the train subset of the dataset in CSV format needs 39921 bytes of space, the test dataset has a disk storage requirement of 10224 bytes.

Table 2 provides the dataset feature statistics, including their minimum, maximum, standard deviation, and mean. And Figure 1 shows distribution of student stress levels in the dataset.

Table 1. Some features of Student Stress Level Dataset.

Features	**Train Dataset**	**Test Dataset**
Sample Size	880	220
Number of Bytes	39921	10224

Table 2. Some statistics of Student Stress Level Dataset.

Features	**Mean**	**Std.**	**Min**	**Max**
AnxietyLevel	11.06	6.11	0	21
SelfEsteem	17.77	8.94	0	30
MentalHealth History	0.49	0.50	0	1
Depression	12.55	7.72	0	27
Headache	2.50	1.40	0	5
BloodPressure	2.18	0.83	1	3
SleepQuality	2.66	1.54	0	5
BreathingProblem	2.75	1.40	0	5
NoiseLevel	2.64	1.32	0	5
LivingConditions	2.51	1.11	0	5
Safety	2.73	1.40	0	5
BasicNeeds	2.77	1.43	0	5
AcademicPerformance	2.77	1.41	0	5
StudyLoad	2.62	1.31	0	5
TeacherStudentRelationship	2.64	1.38	0	5
FutureCareerConcerns	2.64	1.52	0	5
SocialSupport	1.88	1.04	0	3
PeerPressure	2.73	1.42	0	5
ExtracurricularActivities	2.76	1.41	0	5
Bullying	2.61	1.53	0	5
Stress Level	0.99	0.82	0	2

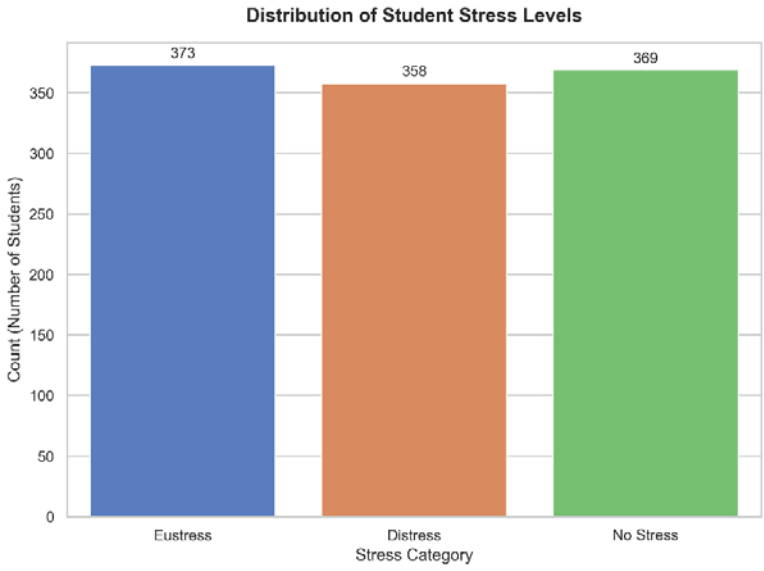


Figure 1. The distribution of student stress levels.

2.2. Support Vector Machines (SVM)

SVM is a supervised ML technique extensively used for both regression and classification problems. In general, it provides high performance in many complex datasets. The main objective of SVM is to establish an optimal decision boundary, known as a hyperplane which separates classes in the data. Rather than selecting an arbitrary boundary, SVM detects the hyperplane which maximizes the margin distance between the closest samples from every category and the hyperplane. By achieving a maximum-margin separation, the algorithm enhances its generalization capability on unseen data, thereby significantly reducing the risk of overfitting (Burges, 1998; Vapnik, 1995; Cortes & Vapnik, 1995; Chang & Lin, 2011; Guyon et al., 2002; Schölkopf et al., 2001).

As seen in Figure 2, the samples of classes A and B are identified as support vectors, and the classes are optimally separated from each other by a line or a curve by maximizing the

margin between them. Thus, the two classes can be classified using this line or curve.

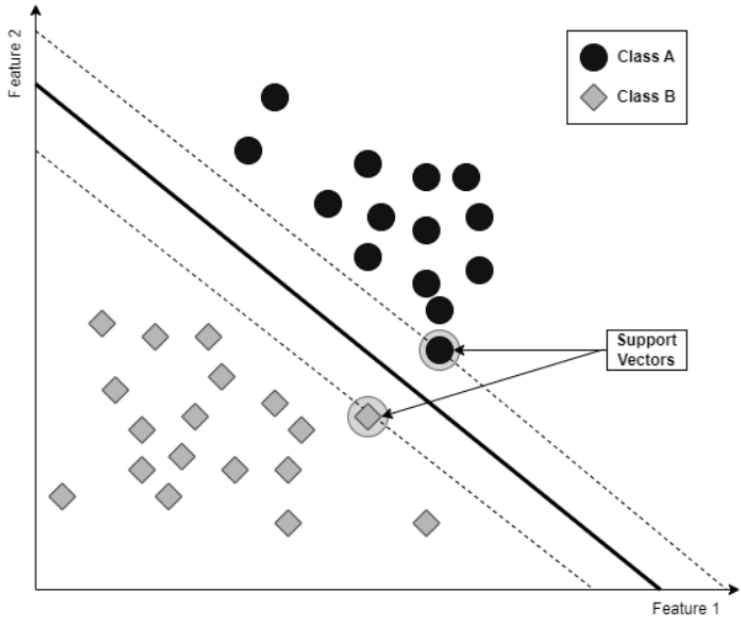


Figure 2. Classification with SVM method using support vectors.

In real-world scenarios such as predicting student stress levels, the data is rarely linearly separable due to the intricate and non-linear relationships between academic, social, and psychological factors. SVM utilizes a kernel that maps the inputs into higher-dimensional spaces. Depending on dataset and hyperparameter, non-linear kernel functions as RBF and Polynomial kernels have been used to capture the patterns of student stress levels.

Since recognizing student stress levels involves multi-class categorization, the traditionally binary SVM algorithm must be adapted for multi-class frameworks. This can be achieved by One-vs-One or One-vs-Rest strategies. SVM is a robust method in dense feature spaces, memory efficiency, and the ability of achieving a global optimum.

2.3. Multi-Layer Perceptron (MLP)

MLP is a feedforward ANN which maps numerical inputs onto numerical outputs. It consists of one input layer, one or more hidden layers, and one output layer. Every neuron utilizes a non-linear function. MLP uses a supervised backpropagation algorithm. MLP's power lies in its architectural complexity, capable of learning highly complex, non-linear relationships (Riedmiller & Heinrich, 1993; Bishop, 1995; Cybenko, 1989; Hornik, 1991; Hornik et al., 1989; LeCun et al., 2012; Minsky & Papert, 1969).

Figure 3 illustrates the structure of a general MLP network. In this network, the inputs are received at the input layer and transferred directly to hidden layers. In hidden and output layers, the sample values and weights are multiplied and summed, and then passed through activation functions, such as Sigmoid, to obtain the outputs.

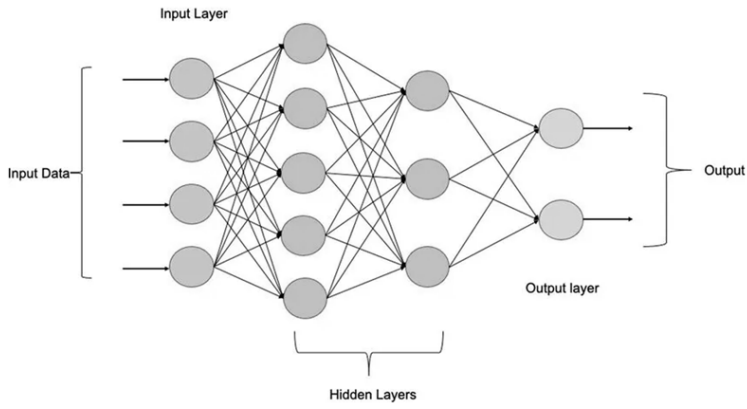


Figure 3. The general structure of MLP network.

During the training phase, data flows forward through the network, where every neuron calculates the sum of multiplication input values and weights, adds a bias, and uses activation functions, such as a Sigmoid or Tanh. The network's final

prediction is compared against the actual target values using loss functions such as cross-entropy loss for categorical stress levels. To minimize this loss, the backpropagation algorithm calculates the error of the network's weights and adjusts them in reverse to the input layer using an optimization algorithm such as Adam optimizer.

In the context of predicting multi-class student stress levels, the output layer of the MLP is typically configured with a Softmax function, which yields a probability distribution across the defined stress categories (e.g., Eustress, Distress, No Stress). MLP does not need rigid assumptions about the distributions of the data, making it highly flexible in capturing latent patterns. However, due to many parameters, MLP is prone to overfitting and needs regularization.

2.4. Random Forest (RF)

RF is a quite successful ensemble technique. It operates by generating many decision trees in the training of the systems. It can be said that RF is an extension of bootstrap aggregating. Hence, it can handle the limitation of decision trees. Their variance and tendency to overfit the training data can be reduced with it. Instead of depending on only one system, it merges the results of many decision trees. For categorization problems, the final results have been determined with majority voting of the trees. It mostly enhances the model’s accuracy, and performance (Breiman, 2001; Breiman et al., 1984; Cutler et al., 2007; Diaz-Uriarte & De Andres, 2006; Genuer et al., 2010; Ho, 1998; Rodriguez et al., 2006).

Figure 4 displays the general structure of the random forest. The train set has been split into many subsets, utilized to train and construct the decision trees. These decision trees can be used to handle categorization and regression issues with majority voting or averaging.

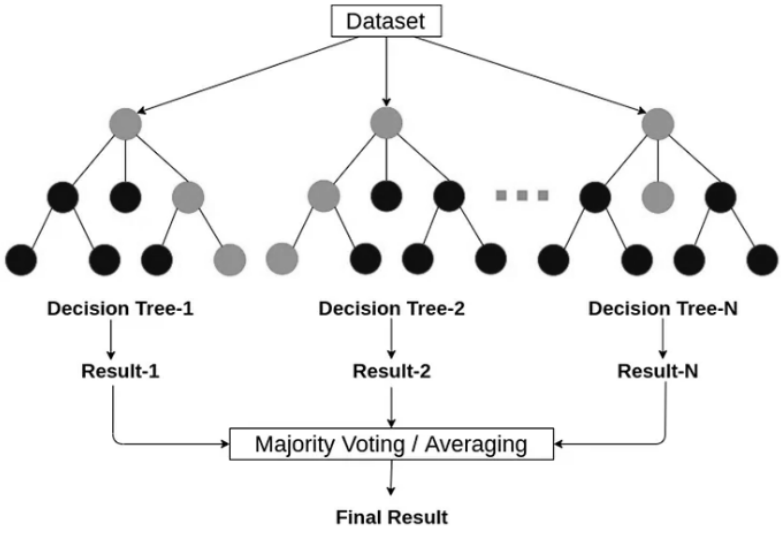


Figure 4. The general structure of Random Forest.

The performance of RF depends mostly on two layers of randomness introduced during its construction. Firstly, each decision tree has been trained using bootstrap samples randomly drawn from the dataset with replacement. Secondly, when dividing a node in a tree, the method does not consider all features instead, it chooses a random feature subset to decide the best part. This randomness decorrelates the trees. As a result, even if certain features are heavily dominant predictors of stress, the method prevents all trees from looking identical.

RF presents some distinct algorithmic advantages over SVM and MLP. RF is capable of handling mixed data types and is resilient to outliers and missing values. In addition, RF provides a mechanism for evaluating feature importance, allowing researchers to quantify exactly which features contribute most significantly to stress levels.

3. RESULTS AND DISCUSSION

In this research, the performances of the systems of MLP, SVM, and Random Forest methods have been evaluated and compared with the metrics of Recall, F1-Score, Precision, Area Under the Curve (AUC), Receiver Operating Characteristic (ROC) curve, and Accuracy. As displayed in Equations 1-4, this work utilizes the metrics: Recall (Rec.), Precision (Pre.), F1-Score, and Accuracy (Acc.).

$$\text{Rec.} = \text{TP}/(\text{FN} + \text{TP}) \quad (1)$$

$$\text{Pre.} = \text{TP}/(\text{FP} + \text{TP}) \quad (2)$$

$$\text{Acc.} = (\text{TN} + \text{TP})/(\text{TN} + \text{TP} + \text{FN} + \text{FP}) \quad (3)$$

$$\text{F1} - \text{Score} = (2 \text{ x Pre. x Rec. l})/(\text{Pre.} + \text{Rec.}) \quad (4)$$

To assess visually the performances of the systems, the ROC curve has been employed, while the AUC has been used to compare the efficiency of the systems. The entire implementation and experimental testing have been carried out with the programming environment of Python.

The systems of Random Forest, MLP, and SVM methods have been trained and tested on a computer with an Intel Core i7 2.7 GHz CPU, 32 GB of RAM, and Windows 11 OS.

The optimal parameters of the models developed using the SVM, MLP, and RF methods have been optimized using the Grid Search algorithm. For the SVM model, the best parameters were identified via Grid Search as “C: 10, gamma: scale, kernel: linear”. The optimal parameters for the RF model were found to be “max_depth: 10, min_samples_split: 3, n_estimators: 50”. The MLP model consists of three hidden layers composed of 16, 3, and 3 neurons, respectively. The first hidden layer utilizes the Sigmoid activation function, while the subsequent layers employ a linear activation function. Through Grid Search, the learning

rate, batch size, and epoch size were determined to be 0.005, 16, and 250, respectively.

In Table 3, the best results of Accuracy, Precision, Recall, F1-Score and AUC using the systems of SVM, MLP and Random Forest have been demonstrated. This MLP model outperforms both SVM and RF models across all evaluation metrics. Using Grid Search algorithm, the MLP model achieved the highest metrics with an Accuracy of 90.5%, a Precision of 90.6%, a Recall of 90.5%, and an F1-Score of 90.5%. These performances suggest that the MLP network successfully captured non-linear features in the student stress dataset. It provides an effective balance between recall and precision.

As demonstrated by Table 3 and Figure 5, the most outstanding finding of this study is the high AUC values obtained from all three models. Whereas SVM indicates quite robust discriminative power at 95.4%, Random Forest and MLP reach an identical, approximately peak of 98.5%.

Although RF yielded the lowest F1-Score metric of 86.8%, its AUC of 98.5% points out that the system has an important inherent ability to separate stress categories. The difference between its AUC and F1-Score shows that the tuning of the baseline categorization threshold could generate much higher success for the RF system.

Table 3. The best results of Accuracy, Precision, Recall, F1-Score and AUC using the systems of SVM, MLP and RF.

	Accuracy	**Precision**	**Recall**	**F1-Score**	**AUC**
SVM	0.891	0.892	0.891	0.891	0.954
RF	0.868	0.870	0.869	0.868	0.985
MLP	0.905	0.906	0.905	0.905	0.985

Figures 5, 6, and 7 illustrate the confusion matrices and ROC curves of the SVM, RF, and MLP methods, respectively. In the confusion matrices, classes 0, 1, and 2 represent “Eustress”,

"Distress", and "No Stress", respectively. For instance, regarding class 0, i.e., the "Eustress" class in the confusion matrix of Figure 5, there are 74 samples in the test dataset, and 61 samples were correctly classified. However, 5 samples were misclassified as "Distress" and 8 samples as "No Stress".

Examination of the confusion matrices reveals that the total number of correctly classified samples by the MLP model is 199 (67 + 66 + 66). At the same time, the total number of correct classifications for the SVM and RF models is observed to be 196 (61 + 68 + 67) and 191 (62 + 66 + 63), respectively. In addition, considering the ROC curves and AUC data, it is evident that the MLP and RF models are substantially more successful compared to the SVM model.

Figure 8 shows training loss curve of the best MLP system. As illustrated in the loss curve, the MLP model has achieved a steady and rapid decline in training error. Moreover, the convergence of the training curves at a low loss value confirms that the MLP model has been successfully optimized without overfitting.

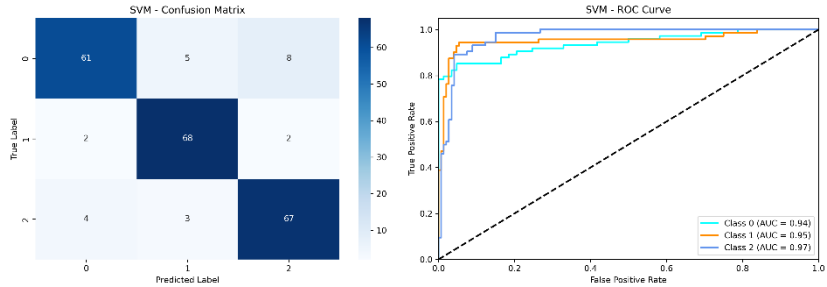


Figure 5. The confusion matrix and ROC curve of the best SVM system.

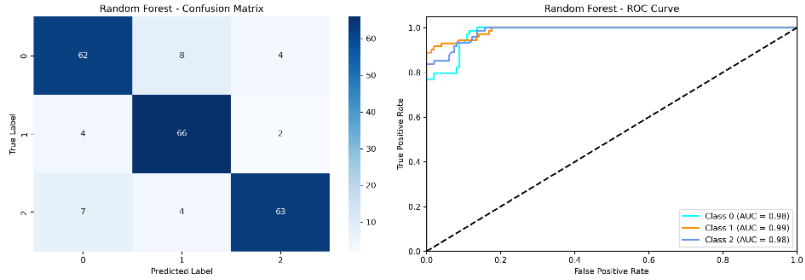


Figure 6. The confusion matrix and ROC curve of the best Random Forest system.

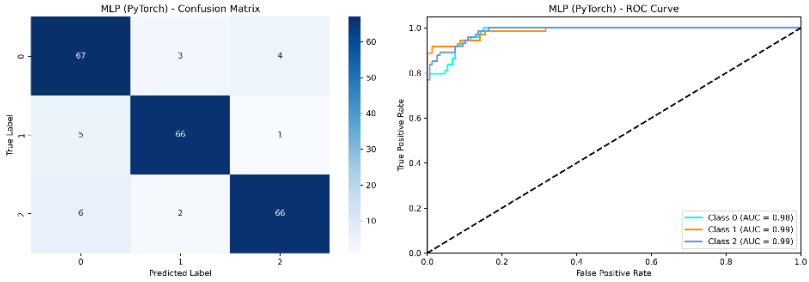


Figure 7. The confusion matrix and ROC curve of the best MLP system.

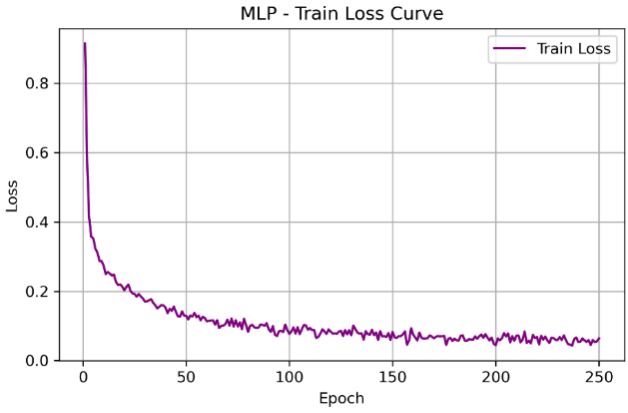


Figure 8. The training loss curve of the best MLP system.

4. CONCLUSION

In this study, the systems have been developed using machine learning methods as SVM, MLP, and RF to detect student stress levels. The parameters of these systems have been optimized using the Grid Search algorithm to identify the most successful configurations. Firstly, the student stress level dataset has been split into 80% training and 20% testing sets. After training the models with the training dataset, they were compared using the test dataset based on Accuracy, Precision, Recall, F1-Score, AUC, and ROC curves. Accordingly, the most successful model was the one developed with MLP, achieving an accuracy of 90.5%, precision of 90.6%, recall of 90.5%, F1-Score of 90.5%, and an AUC value of 98.5%. The optimal parameters for this model were determined via Grid Search and achieved with a learning rate of 0.005, a batch size of 16, and 250 epochs. The model built with SVM became the second most successful model with an F1-Score of 89.1%, while the RF model was the least successful with an F1-Score of 86.8%.

In future work, it is planned to develop new models using different machine learning methods, ensemble approaches, and hybrid methods to detect student stress levels more effectively.

REFERENCES

Amaral, A. P., Soares, M. J., Pinto, A. M., Pereira, A. T., Madeira, N., Bos, S. C., . . . Macedo, A. (2017). Sleep difficulties in college students: The role of stress, affect and cognitive processes. *Psychiatry Research*, 260, 331-337.

Bayram, N., & Bilgel, N. (2008). The prevalence and socio-demographic correlations of depression, anxiety and stress among a group of university students. *Social Psychiatry and Psychiatric Epidemiology*, 43(8), 667-672.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Routledge. https://doi.org/10.1201/9781315139470

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 121–167.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 1-27.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273-297.

Cutler, D. R., Edwards, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). Random forests for classification in ecology. *Ecology*, *88*(11), 2783-2792. https://doi.org/10.1890/07-0539.1

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2*(4), 303-314.

Diaz-Uriarte, R., & De Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics, 7,* Article 3. https://doi.org/10.1186/1471-2105-7-3

Eisenberg, D., Gollust, S. E., Golberstein, E., & Hefner, J. L. (2007). Prevalence and correlates of depression, anxiety, and suicidality among university students. *The American Journal of Orthopsychiatry*, 77(4), 534-542.

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters, 31*(14), 2225-2236. https://doi.org/10.1016/j.patrec.2010.03.014

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1), 389-422.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832-844. https://doi.org/10.1109/34.709601

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4*(2), 251-257.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366.

Hysenbegasi, A., Hass, S. L., & Rowland, C. R. (2005). The impact of depression on the academic productivity of university students. *The Journal of Mental Health Policy and Economics*, 8(3), 145-151.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In G. Montavon, G. B. Orr, & K. R. Müller (Eds.), *Neural networks: Tricks of the trade* (pp. 9-48). Springer.

Minsky, M., & Papert, S. A. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press.

Pascoe, M. C., Hetrick, S. E., & Parker, A. G. (2020). The impact of stress on students in secondary school and higher education. *International Journal of Adolescence and Youth*, *25*(1), 104-112. https://doi.org/10.1080/02673843.2019.1596823

Rani, P. M., & Zufria, I. (2025). Predicting Student Stress Levels Based on Lifestyle Factors Using the Catbost Algorithm. *ZERO: Jurnal Sains, Matematika Dan Terapan*, 9(1), 264. https://doi.org/10.30829/zero.v9i1.24537

Riedmiller, M., & Heinrich, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *IEEE International Conference on Neural Networks*, 586-591.

Rodriguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1619-1630. https://doi.org/10.1109/TPAMI.2006.211

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-536.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a

high-dimensional distribution. *Neural Computation*, 13(7), 1443-1471.

Student Stress Level Dataset. (2026). Student Stress Datasets, https://www.kaggle.com/datasets/mdsultanulislamovi/student-stress-monitoring-datasets/dataasets, Last Accessed in 18.05.2026.

Sulistya, Y. I., Istighosah, M., Setiono, N. H., Putri, L. R., & Lestari Ma'Ruf, A. U. (2025). Segmentation of Student Stress Levels Using Gaussian Mixture Models. 175–180. https://doi.org/10.1109/icaifi66942.2025.11326310

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.

Walburg, V. (2014). Burnout among high school students: A literature review. *Children and Youth Services Review*, 42, 28-33.

Yamasari, Y., Harahap, S. B., Qoiriah, A., Prihanto, A., Sooai, A. G., & Nurhidayat, A. I. (2024). Optimizing ANN Architecture for Classifying Student Stress Levels. 1-5. https://doi.org/10.1109/icoiact64819.2024.10913405

Zahra, R. I., & Kalifia, A. D. (2025). Prediction Of Student Stress Levels Based on Random Forest and The Dass-21 Questionnaire. *Bit-Tech*, *8*(2), 2113–2124. https://doi.org/10.32877/bt.v8i2.3208

PROTEİN DİL MODELLERİ İLE AMİNO ASİT DİZİLİMLERİNİN KARŞILAŞTIRMALI ANALİZİ

Nazan KEMALOĞLU ALAGÖZ[1]

1. GİRİŞ

Moleküler biyoloji ve genetikteki teknolojik devrim, özellikle yüksek başarımlı dizileme (HTS/NGS) ile biyolojik veriyi katlanarak artırmış, bu da biyoinformatiğin ayrı bir disiplin olarak kurumsallaşmasını hızlandırmıştır. Günümüzde DNA, RNA ve protein dizileri ile çok ölçekli "omik" verilerin anlamlandırılması, büyük ölçüde hesaplamalı yöntemlere ve yazılımlara dayanır (Gauthier vd., 2018). Günümüz teknolojileri, genomik, transkriptomik, epigenomik ve metagenomikte devrim yaratarak, geleneksel yöntemlerden çok daha hızlı ve ucuz şekilde muazzam hacimde dizi verisi üretir (Zhao, vd., 2013; Galperin, vd., 2017; Liu, 2019; Branco ve Choupina, 2021).

DNA/RNA/protein dizilerinin gen tanımlama, fonksiyon atama ve sınıflandırması için pek çok web sunucusu ve yazılım geliştirilmiştir (Lee, 2023; Dongare, vd., 2025). NGS ve RNA-seq verileri için hizalama, varyant saptama, ifade analizi gibi adımları kapsayan biyoinformatik yöntemler standart hale gelmiştir (Asım, vd., 2025). Yeni platformlar, makine öğrenmesi ve doğal dil işleme kavramlarını kullanarak biyolojik diziler arası benzerlik, yapı/fonksiyon tahmini ve hastalık ilişkilerini daha isabetli biçimde çözümlemektedir (Lee, 2023; Lightbody, vd.,

[1] Dr. Öğr. Üyesi, Isparta Uygulamalı Bilimler Üniversitesi, Uluborlu Selahattin Karasoy Meslek Yüksekokulu, Bilgisayar Teknolojileri Bölümü, ORCID: 0000-0002-6262-4244.

2023). Veri hacmi ve araç çeşitliliği, biyoinformatiğin biyoloji müfredatına tam entegrasyonunu ve disiplinlerarası ekipleri zorunlu kılmıştır (Gauthier, vd., 2018; Asım, vd., 2025). Proteinlerin 3B yapısını ve fonksiyonlarını deneysel olarak çözmek yavaş ve maliyetlidir. bu açığı kapatmak için derin öğrenme ve diğer makine öğrenmesi yöntemlerine dayalı in silico yaklaşımlar artık modern proteomik araştırmalarının merkezindedir.

Birçok çalışma proteinleri, 20 harfli bir alfabe ile yazılmış cümleler olarak ele alarak; dizideki amino asitler veya k-mer'ler "kelime", tüm protein ise "cümle/doküman" kabul etmektedir (Heinzinger, vd., 2019; Chowdhury, vd., 2022). Büyük, etiketsiz dizi derlemleri üzerinde dil modeli eğitimi, doğal dillerde olduğu gibi bağlama duyarlı gömmeler (embeddings) üretmeyi sağlamaktadır (Rives, vd., 2019; Iuchi, vd., 2021). Bu gömmeler, amino asitlerin kimyasal özelliklerini, protein ailelerini ve uzak akrabalıkları, harici etiket verilmeden ayırt edebilmektedir (Rives, vd., 2019; Liu, vd., 2023). ProGen ve ProGen2 gibi büyük modeller, doğal dillerde cümle üretir gibi yeni protein dizileri üretmektedir. Bu protein dizilerinin önemli bir kısmı deneysel olarak fonksiyonel enzimler olarak doğrulanmaktadır (Elnaggar, vd., 2020; Zhang, vd., 2025).

Kanser immün terapisi ve hücresel iletişim süreçlerinde hayati roller üstlenen protein gruplarının başında sitokinler ve hücresel reseptörler gelmektedir (Yi, vd., 2024; Singh, vd., 2024). Sitokinler, bağışıklık sistemi hücreleri tarafından üretilen, hücreler arası sinyal iletimini sağlayan ve inflamatuar süreçleri koordine eden küçük moleküllü çözünür proteinlerdir (Lee ve Meyerson, 2021). Reseptörler ise bu sinyalleri algılayarak hücre içerisine aktaran, adeta hücrenin dış dünyaya açılan kapıları işlevini gören membran proteinleridir (Singh, vd., 2024).

Bu çalışmada, kanser immünolojisi ve hücresel sinyal iletim mekanizmalarında hayati roller üstlenen insan sitokin ve reseptör proteinlerinin amino asit dizilerinden (sekans) hareketle bilgisayar ortamında yüksek doğrulukla sınıflandırılmasını sağlamak amacıyla NCBI (National Center for Biotechnology Information) veri tabanından kurumsal protokoller aracılığıyla canlı olarak derlenen insan sitokin ve reseptör protein dizileri üzerinde derin öğrenme tabanlı modern bir sınıflandırma metodolojisi yürütülmüştür. Çalışma kapsamında, geleneksel frekans tabanlı n-gram yöntemleri (K-mer + TF-IDF) ile en güncel transformatör (transformer) tabanlı protein dil modelleri olan Meta ESM-2 ve Rostlab ProtBERT mimarilerinin performansı kafa kafaya yarıştırılmıştır. Geliştirilen modeller, veri sızıntısı (data leakage) risklerinden tamamen arındırılmış, bağımsız yapay sinir ağları (ANN) ile eğitilerek iki farklı ölçekteki veri kümesi üzerinde test edilmiştir.

2. MATERYAL VE YÖNTEM

2.1. Frekans Tabanlı Kelime ve Doküman Modelleme Yaklaşımı (K-Mer ve TF-IDF)

Bilgisayar bilimlerinde ve doğal dil işlemede (DDİ), kesikli karakter dizilimlerinin matematiksel olarak ifade edilmesi amacıyla tarihsel olarak istatistiksel frekans modellerinden yararlanılmıştır. Biyoinformatik literatüründe "K-mer" (veya n-gram) yaklaşımı, belirli bir alfabe dizisindeki ardışık alt dizilimlerin metinsel birer kelime öbeği olarak kabul edilmesine dayanmaktadır. Mikrobiyom verilerinde, her dizi “doküman”, her k-mer “terim” kabul edilerek scikit-learn’ün TfidfVectorizer sınıfı ile k-mer’ler TF-IDF ağırlıklarıyla temsil edilebilmiştir (Bokulich, 2024). Bu TF-IDF k-mer matrisleri, mikrobiyal çeşitlilik ölçümü ve denetimli sınıflandırma için kullanılabilmiş; k-mer tabanlı metriklerin filogenetik farkındalığı yüksek

metriklerle yakın korelasyon gösterdiği bildirilmiştir (Bokulich, 2024). BioSet2Vec, TF-IDF benzeri ölçülerle her dizi setini karakterize eden anlamlı k-mer kümeleri ("k-mer sözlükleri") çıkarmakta ve bunu büyük veriye uyumlu bir Spark altyapısıyla ölçeklendirmektedir (Galluzzo, vd., 2025). Matematiksel olarak formüle edildiğinde, bir t teriminin (k-mer) belirli bir d dokümanındaki (dizi) lokal ağırlığı TF, o terimin doküman içindeki frekansıyla doğru orantılıdır. Ancak, tüm veri derleminde (korpus) çok yaygın olan terimlerin ayırt edicilik gücü düşüktür. Bu gürültüyü cezalandırmak amacıyla Ters Belge Frekansı IDF devreye girer. Toplam doküman sayısının N, ilgili terimi barındıran doküman sayısına DF oranının logaritması alınarak hesaplanır (Ozcan, vd., 2019). Nihai TF-IDF ağırlığı, bu iki bileşenin çarpımıyla elde edilir $TF\ X\ IDF$. Bu sayede, tüm dizilerde ortak bulunan ve genel yapısal nitelik taşıyan kalıpların ağırlığı törpülenmekte; sadece belirli alt sınıflara veya fonksiyonel gruplara özgü spesifik sinyal motifleri ön plana çıkarılmaktadır (Cong, vd., 2016).

$$TF(d,t) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t,d}};\quad IDF(t,d) = \log\left(\frac{1}{1+|\{d \in D : t \in d\}|}\right)$$

2.2. Evrimsel Ölçekli Protein Dil Modelleri (ESM Mimarisi)

Doğal dil işlemedeki büyük dil modellerinin (LLM) başarısı, biyolojik dizilerin analizinde "Protein Büyük Dil Modelleri" (pLLM) yaklaşımını doğurmuştur. Meta AI laboratuvarları tarafından geliştirilen ESM (Evolutionary Scale Modeling) mimarisi, biyolojik sekans analizinde transformatör (transformer) tabanlı öz-denetimli (self-supervised) öğrenmenin en gelişmiş teorik örneklerindendir. ESM modelleri, yüz milyonlarca protein dizisi üzerinde "Maskelenmiş Dil

Modellemesi" (Masked Language Modeling - MLM) protokolü uyarınca eğitilir. Bu eğitim sürecinde, dizi içerisindeki belirli amino asit harfleri rastgele gizlenir ve modelden, etraftaki bağlamsal verilerden hareketle gizlenen harfi tahmin etmesi istenir (Rives, vd., 2019; Lin, vd., 2022).

Doğal dil işlemedeki büyük dil modellerinin (LLM) başarısı, biyolojik dizilerin analizinde "Protein Büyük Dil Modelleri" (pLLM) yaklaşımını doğurmuştur. Meta AI laboratuvarları tarafından geliştirilen ESM (Evolutionary Scale Modeling) mimarisi, biyolojik sekans analizinde transformatör (transformer) tabanlı öz-denetimli (self-supervised) öğrenmenin en gelişmiş teorik örneklerindendir. Protein dil modellerinde transformer mimarisi, dikkat haritaları ve havuzlama stratejileri üzerinden diziden yapı ve işlev bilgisi çıkarır. ESM ailesi bu yaklaşımın en gelişmiş örneklerindendir (Rao, vd., 2020).

2.3. Çift Yönlü Entegre Transformatör Modelleri (BERT) ve Boyut Regülasyonu

Biyolojik dil modelleri literatüründeki bir diğer baskın ekol, Google'ın kurucu BERT (Bidirectional Encoder Representations from Transformers) altyapısının amino asit korpusu üzerinde sıfırdan eğitilmesiyle inşa edilen ProtBERT mimarisidir. ESM modellerinden farklı olarak ProtBERT, proteinleri karakter bazlı uzun cümleler olarak ele alır ve çift yönlü (bidirectional) transformatör katmanları sayesinde bir amino asidin hem solundaki hem de sağındaki komşuluk ilişkilerini eş zamanlı olarak modeller (Kang vd., 2022; Ghazikhani ve Butler, 2023). ProtTrans projesi kapsamında, BERT ve türevleri, 393 milyar amino asit içeren devasa protein korpuslarında eğitilmiş; ham embedding'lerin bile yapısal ve fonksiyonel özellikleri yansıttığı gösterilmiştir (Elnaggar vd., 2020). BERT tabanlı yaklaşımlar protein aile sınıflandırmada

%97–99 doğruluklar bildirmiştir (Taju vd., 2021; Guntuboina vd., 2023; Balamurugan vd., 2023).

3. BULGULAR

Bu çalışmada yürütülen tüm analizler Google Colab (Colab Ortamı) üzerinde gerçekleştirilmiştir. Bu doğrultuda, NCBI veri tabanına kurumsal API protokolleri üzerinden bağlanarak sitokin ve reseptör protein dizilimlerinin canlı olarak çekilmesi, FASTA formatındaki ham metinlerin ayrıştırılması ve amino asit harflerinin söz dizimi kontrolünden geçirilmesi ve analiz süreçlerinde Biopython (Bio.Entrez ve Bio.SeqIO modülleri), TensorFlow ve Keras kütüphaneleri kullanılmıştır. Meta ESM-2 ve Rostlab ProtBERT gibi milyarlarca parametreye sahip önceden eğitilmiş protein dil modellerinin tokenizasyon işlemlerini yürütmek ve amino asit dizilimlerini vektörleştirmek amacıyla Transformers, Scikit-Learn ve Torch kütüphanelerinden yararlanılmıştır.

3.1. NBCI Veri Setinin Derlenmesi

Çalışma kapsamında, geliştirilen modellerin ölçeklenebilirlik, genellenebilirlik ve farklı veri hacimlerindeki kararlılık sınırlarını tarafsız bir şekilde test etmek üzere NCBI Protein Veri Tabanı (NCBI Protein Database) üzerinden iki farklı büyüklükte biyolojik veri kümesi canlı olarak derlenmiştir. Bu doğrultuda "Veri Seti 1" olarak adlandırılan pilot ölçekli birinci veri kümesini NCBI veri tabanından çekebilmek için sitokin sınıfında "human cytokine protein", reseptör sınıfında ise "human receptor protein" resmi arama sorguları kullanılmıştır. Arama motorundan sınıf başına en fazla 150 ham kayıt çekilmiş; uzunluk ve karakter filtrelerinden geçen temiz protein dizileri %80 eğitim ve %20 test kümesi olarak rastgele bölünmüştür. Bu küçük ölçekli veri seti, dil modellerinin ürettiği yüksek boyutlu özniteliklerin kısıtlı örneklemlerdeki aşırı öğrenme eğilimlerini

ve boyut indirgeme algoritmalarının matematiksel sınırlarını ölçmek adına oluşturmuştur.

Geniş ölçekli mimariyi test etmek amacıyla kurgulanan "Veri Seti 2" havuzunu derlemek için ise, yine aynı sorgu parametreleri ile sınıf başına 1200 ham kayıt çekilmiştir. Veri Seti 1 kapsamında çekilen ilk 150 proteinin tamamı, Veri Seti 2'deki 1200 proteinlik havuzun ilk kesitini oluşturmaktadır. Dolayısıyla Veri Seti 2, ilkinin basit bir kopyası veya tamamen farklı bir popülasyon değil; ilk verileri de bünyesinde barındıran ancak üzerine 1050 adet yeni, farklı uzunluk varyasyonlarına sahip ve gürültü oranı yüksek protein dizisi eklenmiş genişletilmiş halidir.

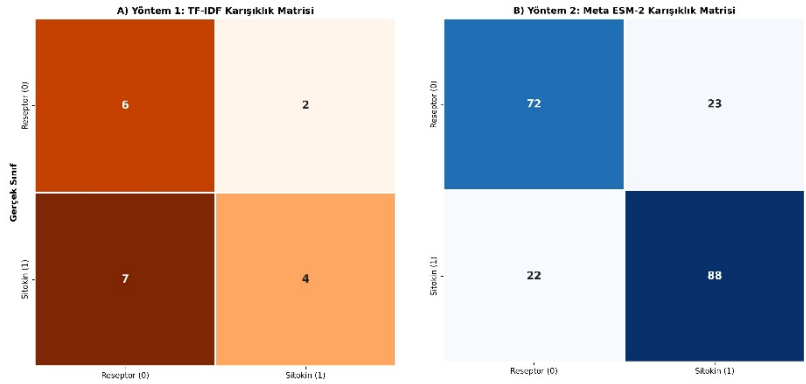
3.2. Model Eğitimi ve Test Sonuçları

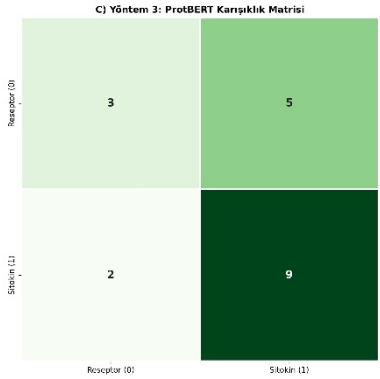
Modelin eğitimi için kullanılan yapay sinir ağları, her iki veri seti için de üç farklı öznitelik uzayında sıfırdan eğitilmiştir. Modellerin tarafsız bir şekilde kıyaslanabilmesi amacıyla kurgulanan ortak ANN mimarisi; giriş katmanının ardından sırasıyla 64 ve 16 nöronlu gizli katmanlardan sigmoid aktivasyonlu bir çıkış katmanından oluşmuştur. Ağın eğitimi, Adam optimizasyon algoritması ve 0.001 sabit öğrenme oranı (learning rate) parametresiyle, binary_crossentropy kayıp fonksiyonu üzerinden 30 epoch boyunca, batch size boyutu 32 olacak şekilde yürütülmüştür. ProtBERT tabanlı öznitelik uzayında ezberlemeyi kırmak adına gizli katmanlara %20 oranında Dropout uygulanmıştır. Veri Seti 1 üzerinde yürütülen deneysel çalışmada, ham 300 protein sekansından 50-400 amino asit uzunluğu ve standart harf filtreleri sonrası süzülen toplam temiz veri sayısı 93 protein olarak kaydedilmiştir. Bu 93 temiz proteinin %80'i (74 protein) eğitime, %20'si (19 protein) ise bağımsız test kümesine ayrıştırılmıştır. Veri Seti 1'e ait performans metrikleri Tablo 3.1 üzerinde verilmiştir.

Tablo 3.1. Veri Seti 1'e ait sayısal analiz sonuçları

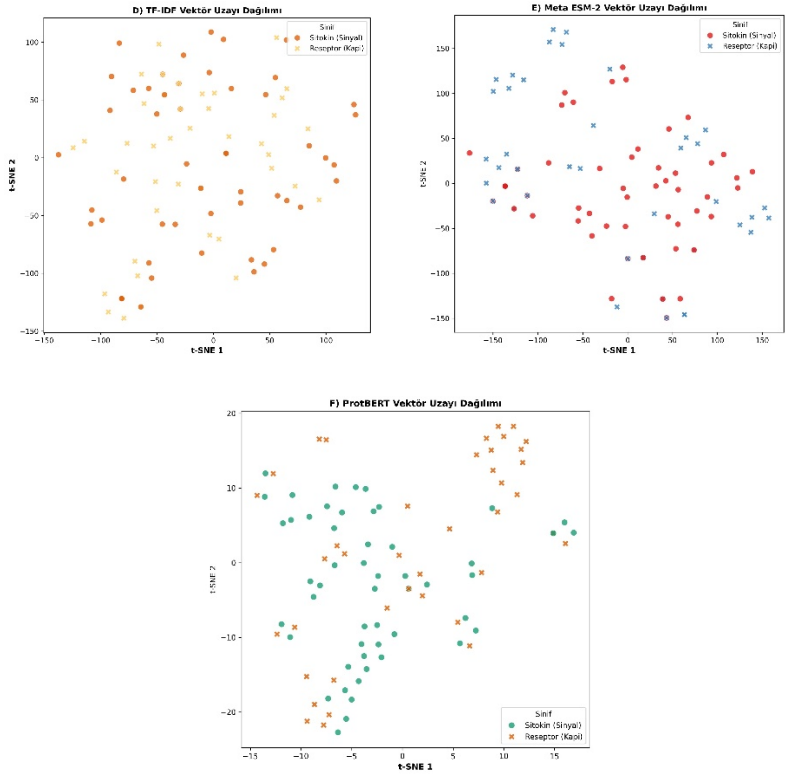
	Öznitelik Boyutu	**Test Doğruluğu**	**F1-Skoru**
YÖNTEM1: K-Mer + TF-IDF	200	%52.63	0,52
YÖNTEM2:MetaESM-2 (esm2_t6_8M)	320	%73.68	0,73
YÖNTEM3: Rostlab ProtBERT + PCA	32	%63.16	0,59

Tablo 3.1 incelendiğinde Yöntem 1 (K-Mer + TF-IDF + ANN), %52,63 test doğruluğu ve 0,52 makro F1-skoru ile doğrusal kelime istatistiklerini haritalandırmış ancak rastlantısal tahmin eşiğinde kalmıştır. Yöntem 2 (Meta ESM-2 + ANN) ise, kısıtlı örneklem boyutuna rağmen geometrik uzayda güçlü bir ayrım çizgisi yakalayarak %73,68 test doğruluğu ve 0,73 makro F1-skoru değerlerine ulaşmıştır. Lineer cebir kurallarına göre PCA algoritmasıyla 32 boyuta sıkıştırılarak aşırı öğrenme eğilimi regüle edilen Yöntem 3 (ProtBERT + ANN) ise, %63,16 test doğruluğu ve 0,59 makro F1-skoru bandında dengelenmiştir. Modellere ait test karışıklık matrisleri ile öznitelik uzaylarının iki boyutlu t-SNE izdüşümleri ise Şekil 3.1 ve Şekil 3.2 üzerinde sunulmuştur.





Şekil 3.1. Veri Seti 1 Ait Karışıklık Matrisleri



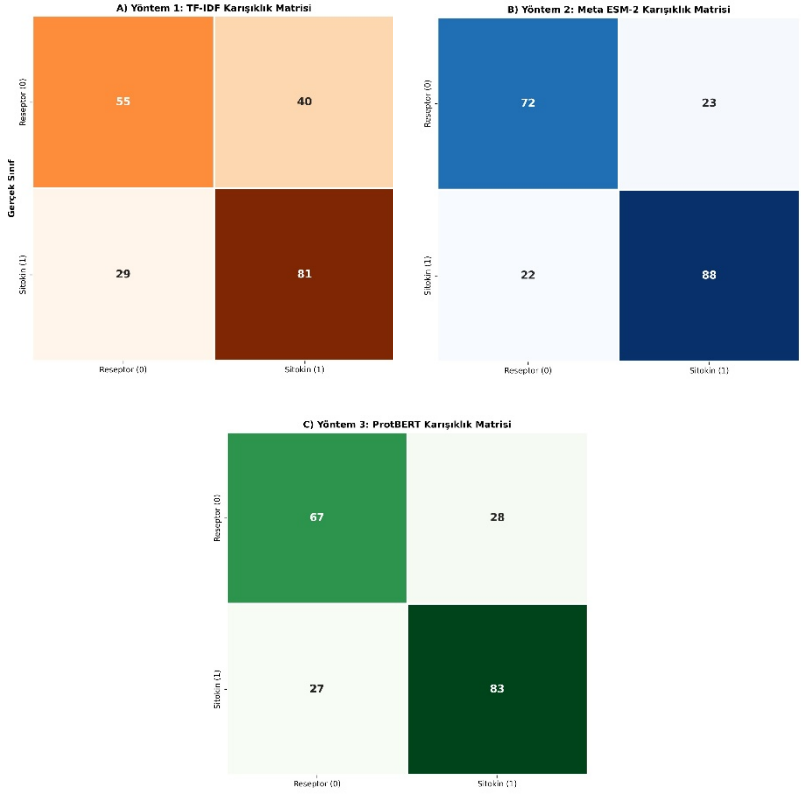
Şekil 3.2. Veri Seti 1 Ait t-SNE Öznitelik Uzayı Dağılım

NCBI hiyerarşisinin genişletilmesiyle elde edilen büyük ölçekli "Veri Seti 2" bünyesinde gerçekleştirilen deneylerde ise, ham 2400 protein kaydından filtreleme protokolü sonrası geriye kalan toplam temiz veri sayısı 1023 protein olarak gerçekleşmiştir. Bu 1023 proteinin %80'i (818 protein) eğitim, %20'si ise (205 protein) test kümesi olarak izole edilmiştir. Veri Seti 2'ye ait performans metrikleri Tablo 3.2 üzerinde verilmiştir. Modellere ait test karışıklık matrisleri ile öznitelik uzaylarının iki boyutlu t-SNE izdüşümleri ise Şekil 3.3 ve Şekil 3.4 üzerinde sunulmuştur.

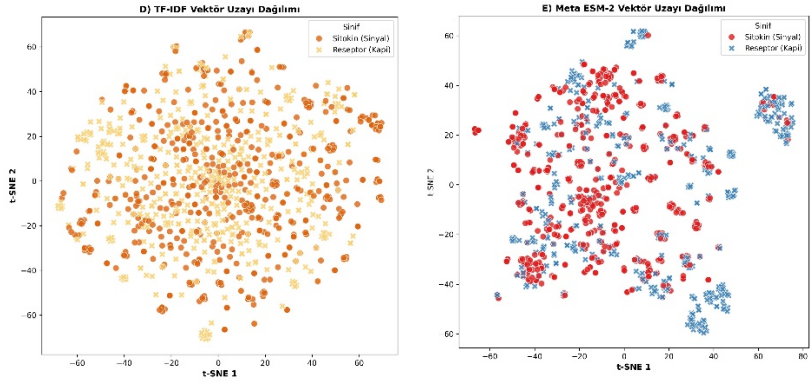
Tablo 3.2. Veri Seti 2'ye ait sayısal analiz sonuçları

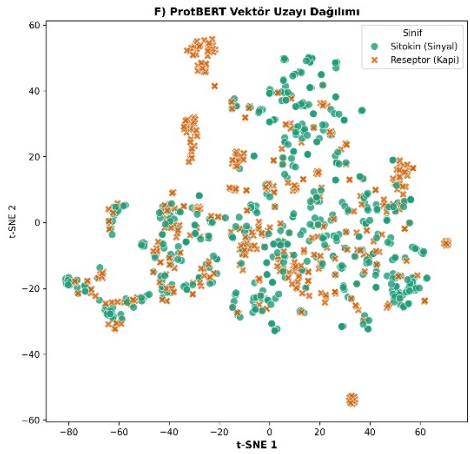
	Öznitelik Boyutu	**Test Doğruluğu**	**F1-Skoru**
Yöntem1: K-Mer + TF-IDF	200	%65,38	0,63
Yöntem2: Meta ESM-2 (esm2_t6_8M)	320	%84,06	0,80
Yöntem3: Rostlab ProtBERT + PCA	128	%78,99	0,76

Tablo 3.2 incelendiğinde Yöntem 1 (TF-IDF) yerel bir doygunluk noktasını aşarak performansını %66,34 test doğruluğu ve 0,66 makro F1-skoru seviyesine yükseltmiştir. ProtBERT'in 1024 boyutlu uzayını bilgi yoğunluğunu koruyacak şekilde 128 bileşene sıkıştıran Yöntem 3 (ProtBERT), genişleyen veri hacminde öğrenme alanını büyüterek %73,17 test doğruluğu ve 0,73 makro F1-skoru ile kayda değer bir başarı sergilemiştir. Yöntem 2 (Meta ESM-2) ise, %78,05 test doğruluğu ve 0,78 makro F1-skoru elde ederek geniş ölçekli veri kümesinde de en yüksek sonucu vermiştir.



Şekil 3.2. Veri Seti 2 'ye Ait Karışıklık Matrisleri





Şekil 3.2. Veri Seti 2'ye Ait t-SNE Öznitelik Uzayı Dağılım

4. TARTIŞMA VE SONUÇ

Yürütülen bu deneysel çalışmada; frekans tabanlı geleneksel bir n-gram yaklaşımı olan K-Mer + TF-IDF metodolojisi ile en güncel transformatör tabanlı protein büyük dil modelleri (pLLM) olan Meta ESM-2 ve Rostlab ProtBERT mimarileri iki farklı veri ölçeğinde karşılaştırılmıştır. Elde edilen deneysel bulgular, protein sekanslarının semantik ve fonksiyonel olarak şifrelenmesinde, derin mimarilerin, doğrusal istatistiksel yaklaşımlara karşı belirgin bir metodolojik üstünlük sağladığını ortaya koymuştur. Pilot ölçekli Veri Seti 1 ve geniş ölçekli Veri Seti 2 üzerinden elde edilen sayısal sonuçlar karşılaştırıldığında, her iki senaryoda da Meta ESM-2 (%73,68 ve %78,05 Test Doğruluğu) mimarisinin en doğru sonuçları verdiği görülmektedir. Geleneksel k-mer frekans vektörleri (TF-IDF), sekansları sadece kesikli harf kombinasyonları olarak ele aldığı için proteinlerin ikincil ve üç boyutlu katlanma geometrilerini, hidrofobik çekirdek dinamiklerini ve uzak mesafeli bağ kurma alanlarını modelleme yeteneğinden yoksundur. Bu durum, TF-IDF modelinin pilot ölçekte %52,63 doğruluğa sıkışarak düşük

performansta kalmasına neden olmuştur. Rostlab ProtBERT modeli ise, ürettiği 1024 boyutlu öznitelik uzayının pilot ölçekteki kısıtlı örneklem sayısına oranla aşırı yüksek kapasite sunması nedeniyle ciddi bir ezberleme riskiyle karşı karşıya kalmıştır. Bu riski önlemek amacıyla uygulanan PCA tabanlı boyut indirgeme protokolü, ProtBERT'in gürültülü bileşenlerini törpüleyerek modeli stabil bir bariyere oturtmuş; ancak bu doğrusal sıkıştırma işlemi, sınıfları ayıran bazı çok ince biyolojik varyansların da kazara kaybolmasına yol açarak ProtBERT'i pilot ölçekte %63,16 doğruluğa sabitlemiştir. Meta ESM-2 modeli ise sitokin sinyal kalıpları ile hücre kapısı işlevi gören reseptör proteinlerini matematiksel uzayda en yüksek anlamsal ayırt edicilikle ayrıştırarak her iki veri setinde de en yüksek performansı göstermiştir.

Literatürde protein dizilerinin kelime gömme (word embedding) paradigmalarıyla analiz edilmesine öncülük eden Asgari ve Mofrad (2015) tarafından yürütülen çalışmalarda, k-mer tabanlı frekans vektörlerinin protein ailelerini belirli bir düzeyde sınıflandırabildiği, ancak sekans çeşitliliği ve veri gürültüsü arttıkça yerel semantik kayıpların kaçınılmaz olduğu raporlanmıştır. Bu tez, çalışmamızda geniş ölçekli veri setinde TF-IDF modelinin %66,34 seviyesine çıksa dahi biyolojik semantiği şifreleyen dil modellerinin gerisinde kalması bulgusuyla birebir örtüşmektedir. Protein dil modellerinin alandaki dönüştürücü etkisini inceleyen Elnaggar vd. (2021) ise, ProtBERT mimarisinin proteinlerin küresel anlambilimsel kimliğini şifrelemedeki başarısını ortaya koymuş, ancak kısıtlı veri kümelerinde model hantallığından ötürü aşırı öğrenme eğilimi gösterebileceğine dikkat çekmiştir. Araştırmamızda ProtBERT uzayına küçük ölçekte uygulanan PCA tabanlı boyut indirgeme ve Dropout (%20) stratejisi, tam olarak Chicco (2017) tarafından hesaplamalı biyolojide aşırı öğrenmeyi kırmak adına

önerilen boyut küçültme tabanlı regülasyon protokollerinin başarılı bir tasviri niteliğindedir.

Öte yandan, Meta AI laboratuvarları tarafından transformatör mimarisinin protein evrimine uyarlanmasıyla geliştirilen ESM-2 modelinin her iki veri setinde de sergilediği mutlak üstünlük Lin vd. (2022) tarafından yayımlanan kurucu makaledeki teorik altyapıyı doğrudan desteklemektedir. Lin vd. (2022), ESM-2 mimarisinin çok kafalı dikkat katmanlarının, protein dizilimlerinin sadece doğrusal sırasını değil; amino asitlerin üç boyutlu uzaydaki atomik düzeyde katlanma geometrilerini ve evrimsel süreç boyunca sergilediği mutasyon kalıplarını doğrudan dikkat matrislerinde şifrelediğini matematiksel olarak kanıtlamıştır. Çalışmamızda ESM-2'nin, ProtBERT gibi 1024 boyutlu hantal bir uzay yerine, ham olarak zaten son derece rafine ve yoğunlaştırılmış 320 boyutlu bir vektör uzayı sunması en büyük avantajı olmuştur. Herhangi bir doğrusal sıkıştırma veya varyans erozyonuna uğratılmadan "ham ve kayıpsız" olarak sınıflandırıcıya beslenen bu 320 boyutlu evrimsel hafıza, sitokin sinyal kalıpları ile hücre kapısı işlevi gören reseptör proteinlerini matematiksel uzayda en yüksek anlamsal ayırt edicilikle ayrıştırmıştır. Geniş veri setinde ESM-2'nin başarısını %78,05'e yükseltmesi, protein dil modellerinin veri ölçeği büyüdükçe evrimsel örüntüleri daha gürbüz (robust) bir şekilde genelleştirebildiğini gösteren alandaki güncel literatür bulgularını doğrulamaktadır.

KAYNAKÇA

Asgari, E., & Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep learning and bioinformatics. PloS One, 10(11), e0141287. https://doi.org/10.1371/journal.pone.0141287.

Asim, M., Ibrahim, M. A., Asif, T., & Dengel, A. (2025). RNA sequence analysis landscape: A comprehensive review of task types, databases, datasets, word embedding methods, and language models. *Heliyon, 11*. https://doi.org/10.1016/j.heliyon.2024.e41488.

Balamurugan, R., Mohite, S., & Raja, J. S. (2023). Protein Sequence Classification Using Bidirectional Encoder Representations from Transformers (BERT) Approach. *SN Computer Science, 4.* https://doi.org/10.1007/s42979-023-01980-1

Bokulich, N. (2024). Integrating sequence composition information into microbial diversity analyses with k-mer frequency counting. *mSystems, 10.* https://doi.org/10.1128/msystems.01550-24

Chicco, D. (2017). Ten quick tips for machine learning in computational biology. BioData Mining, 10(1), 1-17. https://doi.org/10.1186/s13040-017-0155-3.

Chowdhury, R., Bouatta, N., Biswas, S., Floristean, C., Kharkare, A., Roye, K., Rochereau, C., Ahdritz, G., Zhang, J., Church, G. M., Sorger, P., & Alquraishi, M. (2022). Single-sequence protein structure prediction using language models and deep learning. *Nature biotechnology*, *40*, 1617 - 1623. https://doi.org/10.1038/s41587-022-01432-w

Cong, Y., Chan, Y.-B., & Ragan, M. (2016). A novel alignment-free method for detection of lateral genetic transfer based

on TF-IDF. Scientific Reports, 6. https://doi.org/10.1038/srep30308

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., ... & Rost, B. (2021). ProtTrans: toward understanding the language of life through self-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 44(10), 7112-7127. https://doi.org/10.1109/tpami.2021.3095381.

Galluzzo, Y., Giancarlo, R., Rombo, S. E., & Utro, F. (2025). BioSet2Vec: extraction of k-mer dictionaries from multiple sets of biological sequences via big data technologies. *BMC Bioinformatics, 26.* https://doi.org/10.1186/s12859-025-06261-7

Gauthier, J., Vincent, A. T., Charette, S., & Derôme, N. (2018). A brief history of bioinformatics. Briefings in bioinformatics. https://doi.org/10.1093/bib/bby063.

Galperin, M. Y., Fernández-Suárez, X. M., & Rigden, D. (2017). The 24th annual Nucleic Acids Research database issue: a look back and upcoming changes. *Nucleic Acids Research, 45,* 5627 - 5627. https://doi.org/10.1093/nar/gkx021.

Ghazikhani, H., & Butler, G. (2023). Enhanced identification of membrane transport proteins: a hybrid approach combining ProtBERT-BFD and convolutional neural networks. *Journal of Integrative Bioinformatics, 20*. https://doi.org/10.1515/jib-2022-0055

Guntuboina, C., Das, A., Mollaei, P., Kim, S., & Farimani, A. (2023). PeptideBERT: A Language Model Based on Transformers for Peptide Property Prediction. *The Journal of Physical Chemistry Letters, 14*, 10427 - 10434. https://doi.org/10.1021/acs.jpclett.3c02398

Heinzinger, M., Elnaggar, A., Wang, Y., Dallago, C., Nechaev, D., Matthes, F., & Rost, B. (2019). Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics, 20*. https://doi.org/10.1186/s12859-019-3220-8

Iuchi, H., Matsutani, T., Yamada, K., Iwano, N., Sumi, S., Hosoda, S., Zhao, S., Fukunaga, T., & Hamada, M. (2021). Representation learning applications in biological sequence analysis. *Computational and Structural Biotechnology Journal, 19*, 3198 - 3208. https://doi.org/10.1101/2021.02.26.433129.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., . . . Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature, 596*, 583 - 589. https://doi.org/10.1038/s41586-021-03819-2.

Lee, M. N., & Meyerson, M. (2021). Antigen identification for HLA class I- and II-restricted T cell receptors using cytokine-capturing antigen-presenting cells. *Science immunology, 6*. https://doi.org/10.1126/sciimmunol.abf4001

Lee, J.-Y. (2023). The Principles and Applications of High-Throughput Sequencing Technologies. *Development & Reproduction, 27*, 9 - 24. https://doi.org/10.12717/dr.2023.27.1.9

Lightbody, G., Haberland, V., Browne, F., Taggart, L., Zheng, H., Parkes, E., & Blayney, J. (2019). Review of applications of high-throughput sequencing in personalized medicine: barriers and facilitators of future progress in research and

clinical application. *Briefings in Bioinformatics, 20,* 1795 - 1811. https://doi.org/10.1093/bib/bby051.

Lin, Z., Akin, H., Rao, R., Hie, B. L., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., Costa, A. D. S., Fazel-Zarandi, M., Sercu, T., Candido, S., & Rives, A. (2022). Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv*. https://doi.org/10.1126/science.ade2574

Liu, D., Young, F., Robertson, D. L., & Yuan, K. (2023). Prediction of virus-host associations using protein language models and multiple instance learning. *PLOS Computational Biology, 20.* https://doi.org/10.1371/journal.pcbi.1012597.

Liu, B. (2019). BioSeq-Analysis: a platform for DNA, RNA and protein sequence analysis based on machine learning approaches. Briefings in bioinformatics. https://doi.org/10.1093/bib/bbx165.

Ozcan, N. O., Özgür, A., & Gürgen, F. (2019). Statistical representation models for mutation information within genomic data. *BMC Bioinformatics, 20*. https://doi.org/10.1186/s12859-019-2868-4

Rao, R., Meier, J., Sercu, T., Ovchinnikov, S., & Rives, A. (2020). Transformer protein language models are unsupervised structure learners. *bioRxiv*. https://doi.org/10.1101/2020.12.15.422761

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C. L., J., & Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America, 118.* https://doi.org/10.1101/622803.

Singh, A., Bedate, A. M., Richthofen, H. J., Vijver, S. V., Vlist, M., Kuhn, R., Yermanos, A., Kuball, J., Keşmir, C., Inês, M., Ramos, P., & Meyaard, L. (2024). A novel bioinformatics pipeline for the identification of immune inhibitory receptors as potential therapeutic targets. *eLife, 13*. https://doi.org/10.1101/2023.10.24.563834

Taju, S., Shah, S. M. A., & Ou, Y.-Y. (2021). Identification of efflux proteins based on contextual representations with deep bidirectional transformer encoders.. *Analytical biochemistry*, 114416 . https://doi.org/10.1016/j.ab.2021.114416

Yi, M., Li, T., Niu, M., Zhang, H., Wu, Y., Wu, K., & Dai, Z.-T. (2024). Targeting cytokine and chemokine signaling pathways for cancer therapy. *Signal Transduction and Targeted Therapy, 9*. https://doi.org/10.1038/s41392-024-01868-3

Zhang, H., Shi, Y., Wang, Y., Yang, X., Li, K., Im, S.-K., & Han, Y. (2025). Biological Sequence Representation Methods and Recent Advances: A Review. *Biology, 14*. https://doi.org/10.3390/biology14091137.

Zhao, X., Tian, W., Jiang, R., & Wan, J. (2013). Computational Systems Biology. *The Scientific World Journal, 2013*. https://doi.org/10.1155/2013/350358.

MACHINE LEARNING-BASED VEHICLE PRICE CLASSIFICATION: A COMPARATIVE ANALYSIS OF ENSEMBLE AND CLASSICAL APPROACHES

Erol ÖZÇEKİÇ[1]

1. INTRODUCTION

The automotive sector plays a central role in the economic development of industrialized and developing nations alike. Within this sector, the used vehicle market stands out as a particularly dynamic segment, characterized by rapid fluctuations in supply-demand equilibrium and substantial price variability (Kasar et al., 2025; Pias et al., 2025). As global used vehicle transactions surpass hundreds of millions annually and digital marketplaces generate ever-growing volumes of structured sales data, the conditions for applying data-driven approaches to price-level classification have become increasingly favorable. Nevertheless, methodologically rigorous frameworks that go beyond simple regression-based price estimation remain limited in the existing literature.

Traditional vehicle price modeling has largely relied on regression techniques that presuppose linear relationships between predictors and outcomes. Such assumptions, however, prove insufficient when confronted with the non-linear and high-dimensional complexity inherent in real-world vehicle sales data (Gleue et al., 2019; Dhabe, 2023). In recent years, machine

[1] Asst. Prof. Dr., Balikesir University, Balikesir Vocational School, Department of Electronics and Automation, ORCID: 0000-0002-1896-6853.

learning (ML) algorithms have emerged as powerful alternatives capable of capturing these complexities by exploiting large datasets and flexible, adaptive learning structures (Nalamothu, 2023).

The present study departs from the predominant regression paradigm by reformulating the vehicle price estimation problem as a threshold-based binary classification task. Rather than attempting to predict exact prices, the proposed framework assigns each vehicle to one of two categories-low-priced or high-priced-thereby producing more interpretable and decision-oriented outputs. Binary classification frameworks anchored in ensemble tree-based methods have demonstrated robust generalization across heterogeneous prediction domains (Yılmaz, 2025), and their application to used vehicle pricing warrants systematic investigation.

The distinguishing features of this study are threefold. First, model evaluation extends beyond simple accuracy measurement to encompass generalizability, using multiple performance metrics and cross-validation procedures to ensure reliable and stable results. Second, visual and statistical analyses are employed to elucidate the influence of key variables-including vehicle brand, mileage, engine displacement, and year of manufacture-on price-level structure. Third, a sensitivity analysis using multiple classification thresholds (Q1, median, and Q3) demonstrates that the proposed framework maintains consistent performance across both balanced and imbalanced class distributions.

The major contributions of this chapter are as follows. First, the study transforms the vehicle price prediction problem into a threshold-based classification task, offering a more interpretable and decision-oriented scheme compared to regression-based approaches. Second, it systematically

investigates classical and ensemble learning models within a standardized experimental framework, explicitly examining the effect of model diversity on classification accuracy. Third, robustness is validated across multiple threshold conditions, confirming stable performance under varying class distributions. Fourth, feature importance analysis and SHAP-based interpretability methods are combined to provide a comprehensive understanding of the model's decision processes. Finally, the chapter proposes a generalizable classification framework for real-world vehicle price-level assessment that can support applications such as dealership pricing, insurance evaluation, and loan risk scoring.

2. MACHINE LEARNING APPROACHES FOR VEHICLE PRICE PREDICTION

2.1. Theoretical Background

Research on machine learning-based price prediction has advanced considerably over the past decade. Early investigations, as summarized by Dhabe (2023), concentrated on regression-family models that assumed linear relationships among predictors. As datasets expanded and increasingly diverse vehicle features became available, however, the limitations of linear models in capturing highly complex and non-linear interactions became apparent. This recognition catalyzed a shift toward more flexible algorithms, particularly decision trees and ensemble methods capable of learning intricate feature interactions.

A foundational comparative study by Bhatt et al. (2023) evaluated several regularization-based and ensemble algorithms-including Lasso, Ridge, XGBoost, and Random Forest-for predicting used car prices, providing early evidence that tree-based methods consistently outperform linear regressors on heterogeneous vehicle datasets. In the Turkish market context,

Uluturk (2024) demonstrated that Random Forest surpassed alternative methods on both accuracy and general error metrics when applied to real market data, further emphasizing that systematic data preprocessing-encompassing missing-value management, appropriate encoding, and feature scaling-is an indispensable prerequisite for competitive model performance.

A cross-market analysis spanning developed (Germany) and emerging (Romania) used car markets was conducted by Nandhini et al. (2024), who evaluated price prediction models across several neural network architectures, accounting for intricate relationships among vehicle model features, individual add-ons, and visual characteristics. Similarly, Dutulescu et al. (2023) showed that appropriate imputation strategies combined with gradient-boosted ensemble methods substantially improve model accuracy in scenarios involving data-quality issues and missing observations. Collectively, these works confirm that adequate data preparation and algorithm selection are equally critical determinants of predictive performance.

2.2. Ensemble Learning in Vehicle Price Classification

Ensemble learning techniques have gained increasing prominence in the vehicle price prediction literature. Fayyaz et al. (2025) integrated base models with contrasting bias-variance profiles through a meta-learner architecture, demonstrating that complementary representation of the decision boundary yields significant improvements in both accuracy and F1 scores relative to single-model baselines. The bagging mechanism underlying Random Forest-achieved through random sampling and feature subspace techniques-has similarly become a widely adopted strategy for restraining overfitting while simultaneously providing interpretable variable importance rankings (Fayyaz et al., 2025; Bhatt et al., 2023; Nandhini et al., 2024).

In applied settings, Fagbamila et al. (2024) demonstrated the utility of ensemble algorithms in automobile industry pricing strategy, showing that predictive analytics frameworks built on ensemble classifiers generate reliable and actionable market insights. Stacking-based meta-learning architectures, which train higher-level models on the prediction outputs of multiple base classifiers, have been reported to achieve some of the highest accuracy and F1 values documented in the vehicle prediction literature (John et al., 2025; Fayyaz et al., 2025). These findings position ensemble learning as the methodological cornerstone for high-performance vehicle price-level classification.

2.3. Data Preprocessing and Feature Engineering

The preprocessing pipeline is a decisive factor in the performance of machine learning models for vehicle price prediction. A set of procedures widely recognized as best practice in the literature encompasses encoding categorical variables through label encoding or one-hot encoding, transforming the target variable to suit the problem formulation, dividing data into training and test subsets using random and stratified rules, and presenting results via confusion matrices and ROC curves (Dhabe, 2023; Nandhini et al., 2024). Feature scaling through standardization has been shown to be particularly important for distance-sensitive algorithms and those that rely on gradient-based optimization (Msiza & Owolawi, 2023).

Assessment of model performance using class-balance-sensitive metrics-recall, precision, F1, and ROC AUC-provides a more comprehensive picture than single-metric evaluation, which is especially relevant in threshold-based price-level classification scenarios (Dhabe, 2023; Uluturk, 2024). Moreover, k-fold cross-validation has been recommended to visualize performance variance and validate stability, while strategies such as threshold

adjustment and balanced sampling address class imbalance concerns (Msiza & Owolawi, 2023; Kasar et al., 2025).

External and macroeconomic variables are increasingly incorporated into prediction frameworks. Kasar et al. (2025) showed that interest rates, exchange rates, fuel prices, and seasonal demand shocks can substantially enrich model representations by capturing long-run trends and short-run price movements. Hybrid and deep-learning-based architectures (CNN and LSTM) have been proposed for large-scale settings; however, ensemble methods continue to offer superior trade-offs between computational cost and explainability under resource-constrained conditions (Msiza & Owolawi, 2023; Kasar et al., 2025).

2.4. Research Gap

Despite considerable progress in vehicle price modeling, the dominant paradigm in the literature remains regression-based estimation of exact prices. Comparative studies of binary classification frameworks that provide clearer, threshold-based decision boundaries are scarce, as are investigations that systematically validate performance robustness across multiple threshold definitions. Furthermore, interpretability analyses combining feature importance scores with SHAP-based explanations have not been widely applied in the vehicle price classification context. The present study addresses these gaps by proposing a threshold-based binary classification framework that simultaneously emphasizes predictive performance, methodological robustness, and practical decision relevance.

3. MATERIALS AND METHODS

This chapter describes the methodological framework adopted for vehicle price-level classification using machine learning algorithms. Unlike conventional studies that formulate

vehicle price estimation as a regression problem, the proposed approach transforms the continuous price variable into a binary classification target. This reformulation yields a more interpretable and decision-oriented framework suitable for practical applications such as vehicle purchasing guidance, insurance evaluation, dealership pricing, and financial risk assessment.

To investigate the effectiveness of different classification strategies, six machine learning algorithms were employed: Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Extra Trees (ET), Bagging, and Stacking. These algorithms represent distinct learning paradigms while demonstrating strong performance and generalization ability in previous vehicle price prediction studies (Bhatt et al., 2023; Nandhini et al., 2024). Figure 1 illustrates the overall workflow of the proposed methodology, beginning with data acquisition and preprocessing, followed by feature transformation, model development, performance evaluation, and result interpretation.

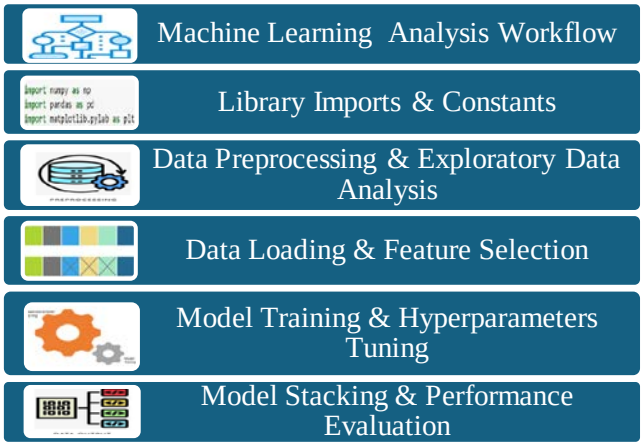


Figure 1. Data Processing Flow Diagram

3.1. Dataset

The dataset utilized in this study was obtained from the Kaggle open-access platform (Öz, 2024) and contains actual used vehicle sales records from the Turkish market, comprising 10,000 complete observations. The dataset encompasses features that reflect both technical vehicle specifications and market-related information, including year of manufacture, engine displacement, mileage, fuel type, transmission type, body type, segment, vehicle brand, vehicle model, number of owners, and selling price.

The multidimensional structure of these variables-simultaneously capturing physical characteristics and consumer preferences-provides a comprehensive basis for modeling vehicle price-level patterns. Table 1 summarizes the dependent and independent variables used in the study.

Table 1. Dependent and Independent Variables

Dependent Variable	Independent Variables
Vehicle Price	Year of Manufacture
	Engine Displacement
	Mileage
	Fuel Type
	Transmission Type
	Vehicle Brand
	Vehicle Model
	Body Type / Segment
	Number of Owners
	Color (optional)

3.2. Data Preprocessing

3.2.1. Data Cleaning

Systematic preprocessing was undertaken to ensure data quality and reliability before model development. In the initial stage, records containing missing, inconsistent, or erroneous values were identified and removed from the dataset. This step

eliminated potentially biased observations and improved the integrity of the data available for model training.

3.2.2. Label Encoding

Categorical variables-including brand, model, fuel type, transmission type, body type, and color-were transformed into numerical representations using Label Encoding. This conversion enabled machine learning algorithms, particularly tree-based methods, to process and learn from categorical attributes. Each unique category within a variable was assigned an integer value, preserving ordinal information where applicable.

3.2.3. Feature Scaling

Numerical features were standardized using the StandardScaler method, which transforms each feature to zero mean and unit variance. This normalization procedure eliminates the influence of differing measurement scales across variables, ensuring that no single feature exerts disproportionate influence on distance-sensitive algorithms and improving overall model stability (Msiza & Owolawi, 2023).

To prevent data leakage, all preprocessing transformations-including label encoding and feature scaling-were fitted exclusively on the training set and subsequently applied to the test set without refitting. This procedure ensures that no information from the test set influences the training process, preserving the validity of the final performance evaluation.

3.2.4. Binary Target Generation

Rather than estimating exact vehicle prices through regression, the continuous target variable was reformulated as a binary classification problem. Using the median price value (8,858.5 USD) as the primary classification threshold, vehicles were assigned to one of two categories: low-priced (Class 0) and

high-priced (Class 1). This threshold was selected because it produces an approximately balanced class distribution (50% / 50%), thereby minimizing class imbalance bias during model training and evaluation.

To assess the sensitivity of classification performance to the choice of threshold, additional analyses were conducted using the first quartile (Q1 = 8,038.0 USD) and third quartile (Q3 = 9,701.4 USD) as alternative boundaries. As shown in Table 2 and Figure 2, the proposed framework demonstrated stable performance across all three threshold conditions, confirming that the classification results are not an artifact of a single threshold choice but reflect genuine discriminative capacity of the models.

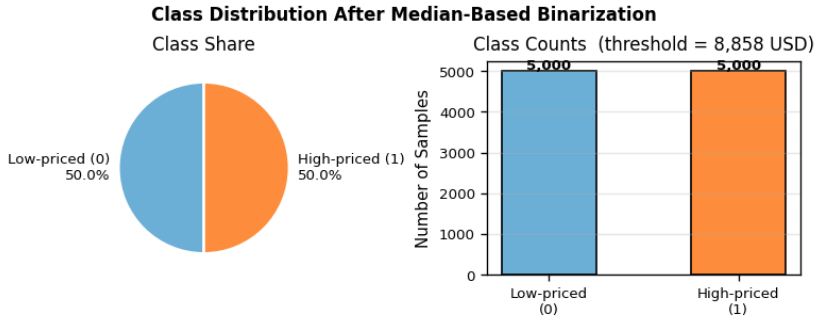


Figure 2. Distribution of Low-Priced and High-Priced Classes in the Dataset

Table 2. Effect of Threshold Value on Class Distribution and Model Performance

Threshold (USD)	Class 0 (n)	Class 1 (n)	Class 0 %	Class 1 %	ET F1	ET Accuracy	ET ROC AUC
8,038.0 (Q1)	4,001	5,999	40.01%	59.99%	0.9697	0.9635	0.9961
8,858.5 (Median)	5,000	5,000	50.00%	50.00%	0.9633	0.9635	0.9956
9,701.4 (Q3)	6,000	4,000	60.00%	40.00%	0.9582	0.9665	0.9959

3.2.5. Correlation Analysis

Relationships among variables were examined using correlation analysis, the results of which are presented in Figure 3. Year of manufacture exhibited a strong positive correlation with vehicle price, while mileage showed a strong negative correlation. Engine displacement demonstrated a moderate positive association with price. These findings informed feature selection decisions and provided an initial indication of the variables most likely to contribute to predictive performance.

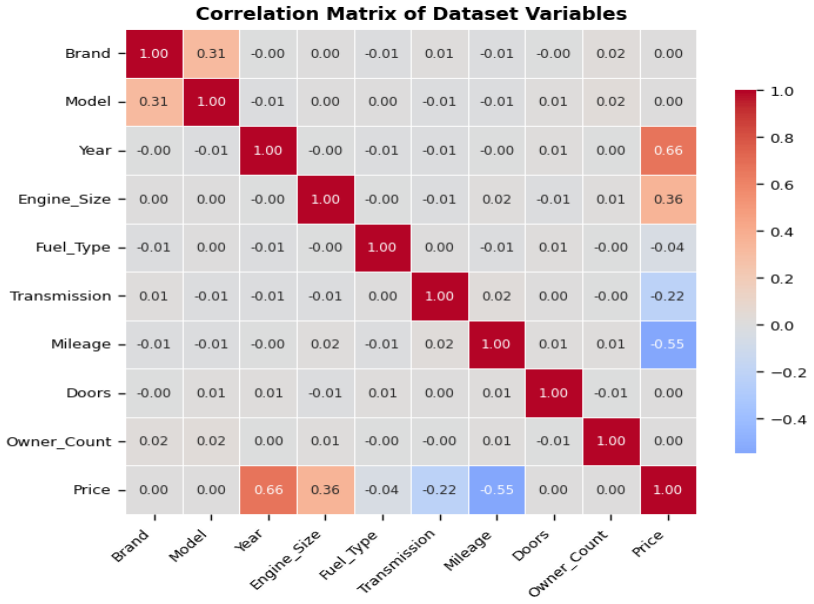


Figure 3. Correlation Matrix of Dataset Variables

3.3. Experimental Design

For model development, the dataset was partitioned into an 80% training subset and a 20% test subset using stratified random sampling, which ensures that the class distribution is preserved across both splits. The test set was held entirely separate throughout model development and hyperparameter optimization, being used exclusively for final performance

evaluation to provide an unbiased assessment of model generalization capability (Fagbamila et al., 2024).

Hyperparameter optimization was performed on the training set using Grid Search with five-fold cross-validation, using the F1-score as the optimization criterion. For RF and ET, the parameters n_estimators, max_depth, and min_samples_split were tuned; for DT, max_depth and min_samples_split; for Bagging, n_estimators, max_samples, and max_features; and for NB, the var_smoothing parameter. Table 3 summarizes the search spaces and optimal values obtained for each model.

Table 3. Hyperparameters and Optimal Values for Models

Model	Search Space (Key Hyperparameters)	Optimal Values	Best CV F1
Random Forest	n_estimators: [100,200]; max_depth: [None,10,20]; min_samples_split: [2,5]	max_depth: None; min_samples_split: 5; n_estimators: 100	0.947
Extra Trees	n_estimators: [100,200]; max_depth: [None,10,20]; min_samples_split: [2,5]	n_estimators: 200; max_depth: None; min_samples_split: 5	0.963
Decision Tree	max_depth: [None,5,10,20]; min_samples_split: [2,5,10]	max_depth: None; min_samples_split: 5	0.900
Bagging	n_estimators: [50,100,200]; max_samples: [0.5,0.7,1.0]; max_features: [0.5,0.7,1.0]	max_features: 1.0; max_samples: 1.0; n_estimators: 100	0.941
Naive Bayes	var_smoothing: [1e-09, 1e-08, 1e-07]	var_smoothing: 1e-09	0.907
Stacking	Base learners: DT, RF, NB, Bagging, ET (tuned); Meta-learner: RF (tuned)	Base learner params: see individual models; RF meta-learner: max_depth: None; min_samples_split: 5; n_estimators: 100	N/A (meta-learner)

To mitigate potential class imbalance effects, the class_weight='balanced' parameter was applied to all tree-based classifiers (DT, RF, ET), which automatically adjusts class weights inversely proportional to their frequencies. Although the median-based threshold produced a near-perfect 50/50 class distribution in this dataset, this safeguard was retained to ensure unbiased learning in the event of minor distributional shifts across cross-validation folds.

All experiments were implemented in Python using the Scikit-learn, NumPy, Pandas, and Matplotlib libraries (Saini & Rani, 2023; Bhatnagar et al., 2024). All algorithms were trained using the same training-testing partition, while model-specific hyperparameters were independently optimized through Grid Search. The Stacking model was not subjected to a separate grid search; instead, the tuned parameters of its base learners were carried over directly.

3.4. Machine Learning Models

3.4.1. Decision Tree

The Decision Tree model generates classification rules by recursively partitioning the feature space into branches (Chawhan, 2024). At each internal node, the algorithm selects the feature that maximizes information gain, operationalized through purity measures such as the Gini index or entropy (Gupta et al., 2022). The Gini index at node t is defined as:

$$Gini(t) = 1 - \sum_{i=1}^{C} p_i^2 \quad (1)$$

where pi denotes the proportion of samples belonging to class i at node t, and C represents the total number of classes. A low Gini value indicates high homogeneity among samples at the node, implying that classes are more distinctly separated. Although decision trees are intuitively interpretable and capable of capturing non-linear relationships, they are prone to overfitting

when allowed to grow to arbitrary depth. For this reason, the DT model is evaluated both as a standalone classifier and as a base learner in ensemble configurations.

3.4.2. Random Forest

Random Forest is an ensemble model formed by training multiple decision trees on independently drawn random subsamples of the data, each using a randomly selected subset of features (Gohil, 2025). The final prediction is obtained by majority voting across all trees, which reduces model variance while increasing accuracy and generalization ability (Liaw & Wiener, 2001). The prediction structure is formalized as:

$$\hat{y} = \text{mode}\{h_b(x),\ b = 1,2,\dots,B\} \quad (2)$$

where $h_b(x)$ denotes the prediction function of the b-th decision tree, B is the total number of trees, and mode{·} represents majority voting. The diversity induced by random sample and feature selection suppresses overfitting and produces a stable general model particularly well-suited to datasets containing both numerical and categorical variables.

3.4.3. Naïve Bayes

The Naïve Bayes classifier is a probabilistic model grounded in Bayes' theorem (Monburinon et al., 2018). The model computes the posterior probability of class membership using conditional probabilities associated with each feature, under the simplifying assumption that features are mutually independent given the class label (Nandan & Ghosh, 2023). The classification rule is expressed as:

$$P(C_k|X) = \frac{P(X|C_k)\,P(C_k)}{P(X)} \quad (3)$$

where $P(C_k|X)$ denotes the probability of the observation belonging to class C_k, $P(X|C_k)$ is the conditional probability of the features given class Ck, $P(C_k)$ is the prior probability of class C_k, and P(X) represents the total probability of the observation.

The class assignment is then determined by:

$$\hat{y} = \arg\max_{C_k} P\left(C_k \mid X\right) \quad (4)$$

The Gaussian NB variant used in this study models continuous features under a normal distribution assumption. Although its feature independence assumption limits performance on datasets with correlated predictors, the model provides a computationally efficient baseline and reinforces the statistical grounding of the comparative analysis.

3.4.4. Bagging

Bagging (Bootstrap Aggregating) operates by repeatedly training a base learner on different random subsamples of the training data with replacement (Raj et al., 2025). The final prediction is obtained by averaging (for regression) or majority voting (for classification) across all sub-models, reducing variance and improving robustness on noisy data (Hemendiran & Renjith, 2023). The aggregation procedure is:

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} h_b(x) \quad (5)$$

where $h_b(x)$ is the prediction of the b-th sub-model and B is the total number of sub-models. The inherent randomness of this structure reduces overfitting risk and enhances prediction stability, making Bagging a particularly effective reference model for evaluating ensemble strategies in heterogeneous datasets.

3.4.5. Extra Trees

The Extra Trees (Extremely Randomized Trees) algorithm is structurally similar to Random Forest but introduces additional randomization by selecting split thresholds entirely at random, rather than optimizing them (Jing, 2022). This additional source of variance reduction renders the model more robust against overfitting and shortens training time (Fagbamila et al., 2024). The prediction is obtained by averaging tree outputs:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \quad (6)$$

where $h_t(x)$ is the prediction of the t-th tree and T is the total number of trees. The increased tree diversity achieved through random split points enhances generalization performance, particularly on large and complex datasets, often allowing Extra Trees to achieve high accuracy with lower computational cost than Random Forest.

3.4.6. Stacking

Stacking (Stacked Generalization) is a meta-learning ensemble technique that combines the predictive strengths of multiple heterogeneous base classifiers through a higher-level meta-learner (John et al., 2025). Base classifiers are trained on the full training set, and their prediction outputs are used as input features for the meta-learner, which learns the optimal combination to produce the final prediction (Cui et al., 2022). The framework is formalized as:

$$\hat{y} = f_m\big(h_1(x), h_2(x), \ldots, h_n(x)\big) \quad (7)$$

where $h_1(x)$, $h_2(x)$, …, $h_n(x)$ are the prediction functions of the base classifiers and $fm(\cdot)$ is the meta-learner. In this study, DT, RF, NB, Bagging, and ET were employed as base learners. Random Forest was selected as the meta-learner due to its robustness to overfitting, its capacity to handle diverse input

distributions produced by heterogeneous base learners, and its established effectiveness in stacking architectures (Fayyaz et al., 2025; Gohil, 2025). Unlike linear meta-learners, RF can capture non-linear relationships among base learner outputs, which is advantageous when base classifiers exhibit varying bias-variance profiles.

3.5. Performance Metrics

Model performance was evaluated using a combination of five complementary metrics to provide a comprehensive assessment of both overall classification success and the nature of misclassifications. Relying on a single metric is insufficient when class distributions may vary or when different types of errors carry distinct practical consequences.

Accuracy measures the proportion of all correctly classified samples and serves as the primary benchmark for cross-model comparison:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

Precision quantifies the proportion of positive predictions that are truly positive, and is particularly important in applications where false positives carry significant costs:

$$Precision = \frac{TP}{TP+FP} \tag{9}$$

Recall (Sensitivity) measures the proportion of actual positive samples correctly identified, and is critical in scenarios where missed positive cases are consequential:

$$Recall = \frac{TP}{TP+FN} \tag{10}$$

F1-Score is the harmonic mean of Precision and Recall, balancing both metrics and providing a comprehensive view of performance on potentially imbalanced datasets:

$$F1 = 2 \text{ x } \frac{Precision \times Recall}{Precision + Recall} \quad (11)$$

ROC AUC (Receiver Operating Characteristic – Area Under Curve) measures the model's discriminative power by quantifying the relationship between the true positive rate and the false positive rate across all classification thresholds. An AUC value approaching 1 indicates near-perfect discrimination, while a value of 0.5 corresponds to random prediction:

$$AUC = \int_0^1 TPR(FPR)\ d(FPR) \quad (12)$$

Joint evaluation of these five metrics enables robust comparison of ensemble methods (RF, ET, Stacking) against classical baselines, informing the final model selection on the basis of both statistical performance and practical relevance.

4. RESULTS AND DISCUSSION

4.1. Descriptive Analysis

Preliminary descriptive analyses reveal that several variables are systematically associated with vehicle price levels. Year of manufacture, engine displacement, mileage, and brand consistently emerge as the most influential predictors, a finding subsequently confirmed through formal feature importance analysis.

Figure 4 illustrates the evolution of average vehicle prices by year of manufacture. A sustained upward trend is observable from 2000 onward, reflecting the combined effects of economic fluctuations, exchange rate movements, and shifts in consumer demand. Newer vehicles command higher average prices, consistent with established patterns of depreciation.

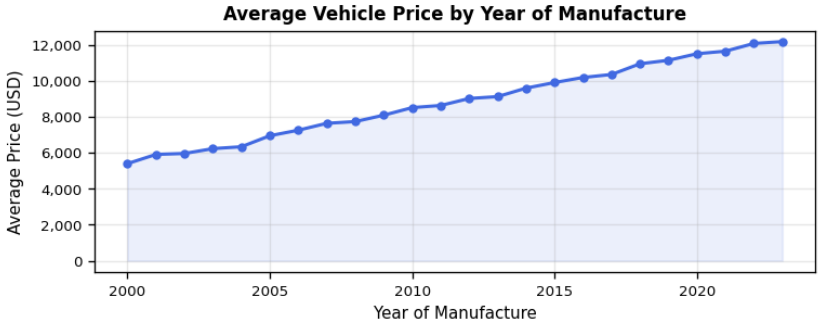


Figure 4. Average Vehicle Price by Year of Manufacture

Figure 5 presents the relationship between engine displacement and vehicle price. A clear positive association is apparent: vehicles with larger engine displacement are, on average, more expensive, likely reflecting the premium placed on performance, comfort, and market positioning associated with larger engines.

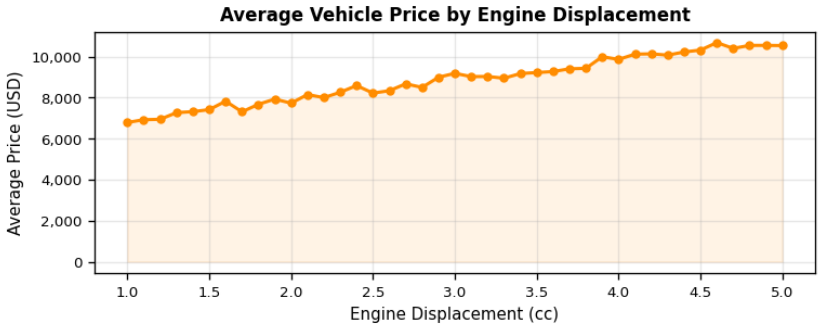


Figure 5. Average Vehicle Price by Engine Displacement

The influence of mileage on vehicle price is displayed in Figure 6. Average prices decline sharply as mileage increases, with the depreciation effect accelerating markedly once mileage exceeds 200,000 km, where accumulated usage has a pronounced impact on perceived vehicle condition and residual value.

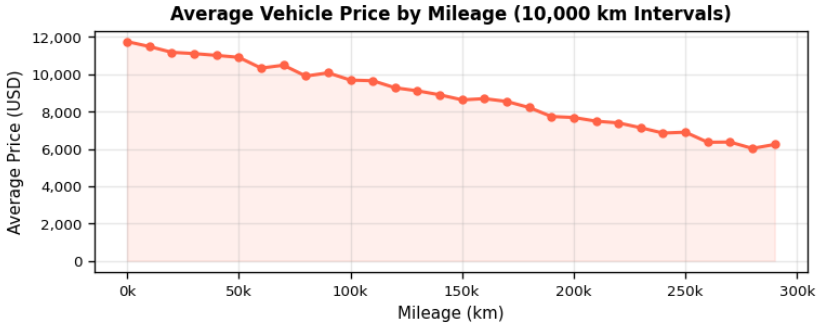


Figure 6. Average Vehicle Price by Mileage (10,000 km Intervals)

Figure 7 depicts the distribution of total price share by vehicle brand. Brands such as Ford, Audi, and Volkswagen account for disproportionately high shares of total market value, indicating that brand premium plays a significant role in vehicle price formation, independent of technical specifications.

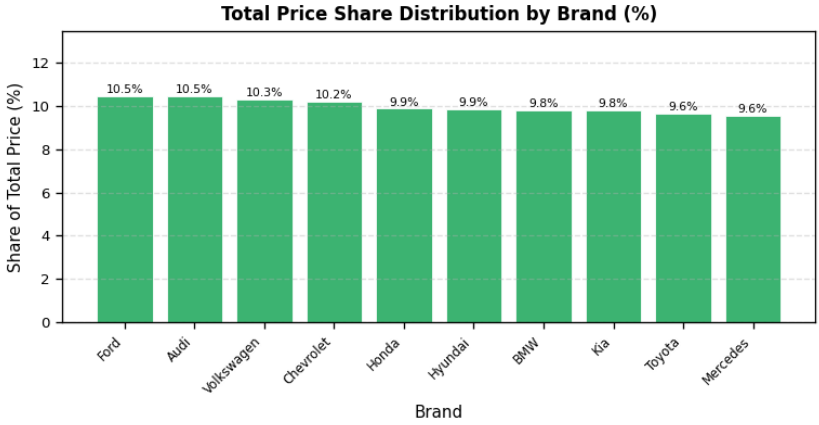


Figure 7. Total Price Share Distribution by Brand (%)

4.2. Model Performance

All six machine learning models were evaluated on the held-out test set following training and hyperparameter optimization on the training data. Confusion matrices and ROC curves were computed for each model and are presented in Figures 8 and 9, respectively.

As shown in Figure 8, ensemble-based models—particularly Extra Trees and Stacking—exhibit substantially fewer misclassifications compared to classical approaches. The confusion matrices visually confirm that RF, ET, and Stacking achieve markedly higher correct classifications in both price classes, while the Naïve Bayes model displays the greatest inter-class error rates, attributable to its feature independence assumption.

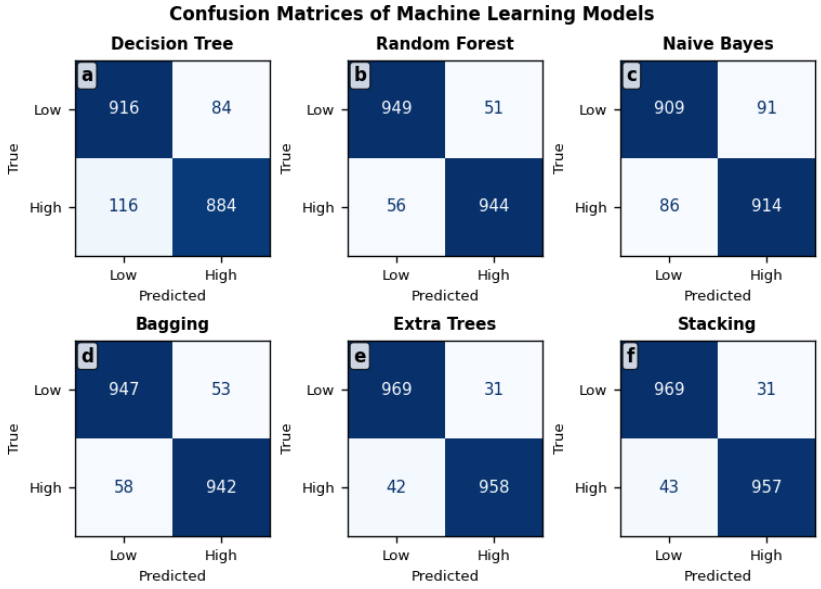


Figure 8. Confusion Matrices of Machine Learning Models

The ROC curves presented in Figure 9 corroborate these findings. Ensemble models demonstrate superior discriminative performance, with AUC values of approximately 0.99 for RF, ET, and Stacking, compared to lower values for DT and NB. These curves confirm that ensemble models reliably distinguish between high-priced and low-priced vehicle classes across all classification thresholds.

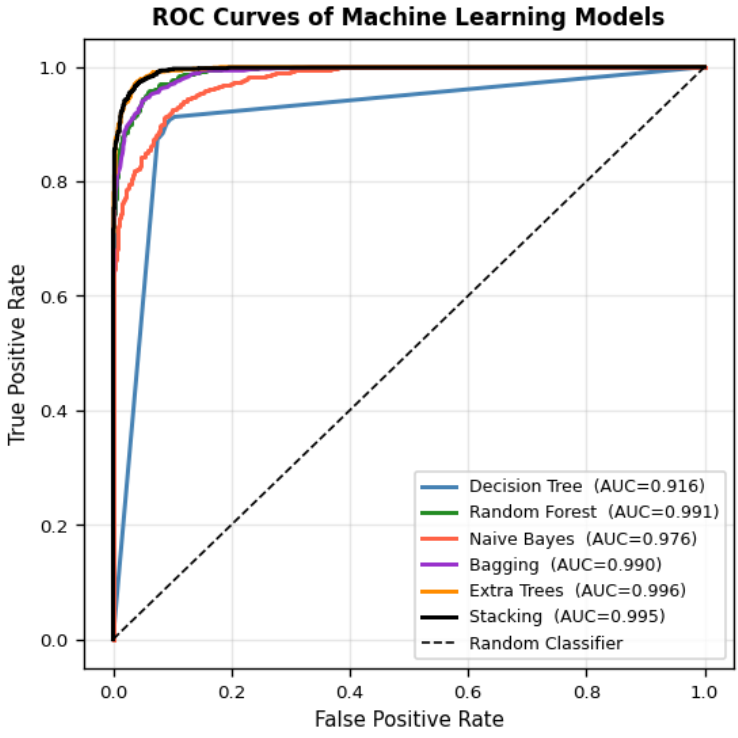


Figure 9. ROC Curves of Machine Learning Models

Quantitative performance metrics for all models are summarized in Table 4. Extra Trees and Stacking achieved the highest overall performance, with accuracy values of approximately 96.4% and 96.3%, respectively, and ROC AUC values exceeding 0.99. Random Forest and Bagging demonstrated competitive mid-range performance, while Decision Tree and Naïve Bayes showed comparatively lower results.

Table 4. Model Performance Metrics

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	0.900	0.913	0.884	0.898	0.918
Random Forest	0.947	0.949	0.944	0.947	0.991
Naïve Bayes	0.918	0.910	0.914	0.912	0.976
Bagging	0.945	0.958	0.953	0.944	0.990
Extra Trees	0.964	0.969	0.958	0.963	0.996
Stacking	0.963	0.969	0.957	0.963	0.995

The high ROC AUC values do not imply overfitting; rather, they reflect the well-established and theoretically grounded associations between the primary predictive features and vehicle price. The highest observed correlation between any individual feature and the price variable remained below 0.70, confirming that no single predictor deterministically drives the classification outcome. Data leakage was explicitly controlled throughout: the target variable was derived solely from the raw price attribute, no proxy or transformed price-encoding variables were included, and all preprocessing transformations were fitted exclusively on the training set.

The absence of overfitting is further supported by the small gap between cross-validation and test set accuracies: 0.0007 for Extra Trees (CV: 0.9628, Test: 0.9635) and 0.0009 for Stacking (CV: 0.9621, Test: 0.9630). These negligible differences confirm that both models generalize well to unseen data.

To statistically compare the Extra Trees and Stacking models, a paired t-test was conducted. The results indicated no statistically significant difference between the two models ($t = 0.5872$, $p = 0.5886 > 0.05$), confirming that both achieve

equivalent levels of performance. The five-fold cross-validation results, presented in Table 5, further validate the stability of all models across different data subsets.

Table 5. Five-Fold Cross-Validation Results (Accuracy)

Model	CV Mean	CV Std	CV Min	CV Max
Decision Tree	0.9013	0.0046	0.8938	0.9075
Random Forest	0.9448	0.0036	0.9412	0.9494
Naïve Bayes	0.9070	0.0057	0.9019	0.9156
Bagging	0.9419	0.0034	0.9381	0.9462
Extra Trees	0.9628	0.0045	0.9544	0.9681
Stacking	0.9621	0.0037	0.9569	0.9675

4.3. Feature Importance Analysis

Feature importance analysis was conducted using the Extra Trees model, which achieved the highest overall performance. As illustrated in Figure 10, year of manufacture exhibited the highest importance score, followed by mileage and engine displacement. These three features collectively account for the dominant share of the model's predictive power. Categorical features such as fuel type, transmission type, and brand contribute comparatively less, suggesting that temporal and technical vehicle characteristics are more decisive than categorical attributes in binary price-level classification.

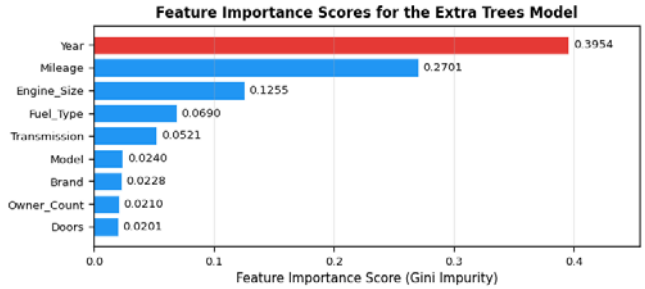


Figure 10. Feature Importance Scores for the Extra Trees Model

4.4. SHAP Analysis

To complement the Gini-based importance scores and provide directional interpretability, SHAP (SHapley Additive exPlanations) analysis was performed on the Extra Trees model. Figure 11 presents the mean absolute SHAP values for each feature, confirming that year of manufacture, mileage, and engine displacement are the three most influential predictors—consistent with the feature importance results in Figure 10. This cross-method consistency strengthens the reliability of the interpretability analysis.

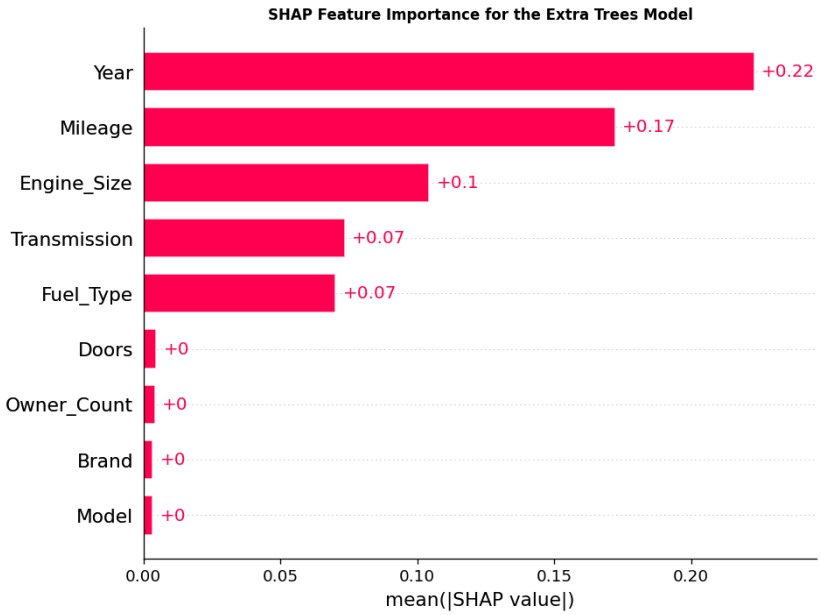


Figure 11. SHAP Feature Importance for the Extra Trees Model

The SHAP beeswarm plot in Figure 12 reveals the directional effects of each feature on model predictions. Higher values of year of manufacture (i.e., newer vehicles) are strongly and consistently associated with positive SHAP contributions, pushing predictions toward the high-priced class. Conversely, high mileage values generate consistently negative SHAP

contributions, reflecting the depreciation effect of usage on vehicle price. Engine displacement exhibits a predominantly positive association with the high-priced class, though with greater variance across samples, suggesting a more context-dependent effect than year of manufacture or mileage. These directional findings are theoretically coherent and align with established vehicle depreciation and valuation principles (Gleue et al., 2019; Fayyaz et al., 2025).

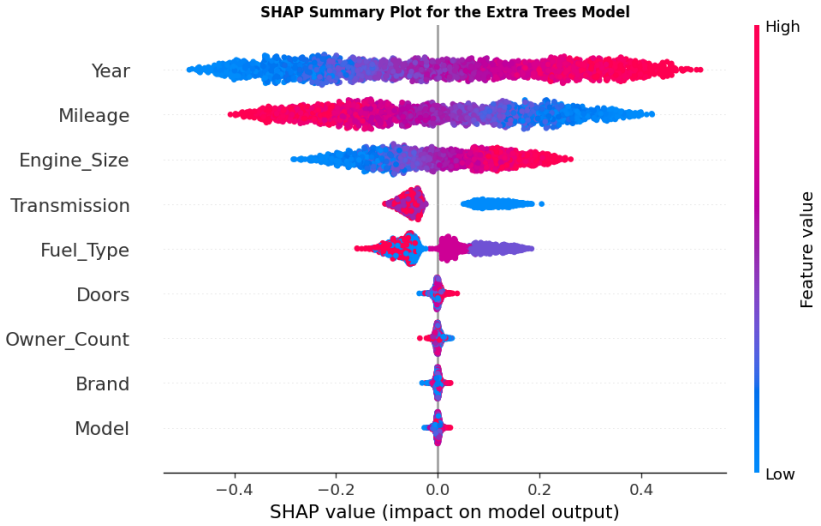


Figure 12. SHAP Summary Plot for the Extra Trees Model

4.5. Discussion

The results collectively demonstrate the superiority of ensemble-based models over classical methods for used vehicle price classification. Extra Trees and Stacking achieved the highest accuracy and generalization performance, with their ability to leverage diverse learners proving particularly advantageous for capturing complex non-linear relationships in heterogeneous vehicle data. The Bagging model delivered stable mid-range performance, while the Naïve Bayes model's lower accuracy reflects the limitations of its feature independence

assumption in the presence of correlated predictors. The Decision Tree exhibited a higher tendency toward overfitting when compared to ensemble counterparts, reinforcing the well-documented variance reduction benefits of ensemble strategies.

The sensitivity analysis across three threshold conditions (Q1, median, Q3) confirmed that classification performance remains stable across varying class distributions, with Extra Trees F1 scores ranging from 0.958 to 0.9697 and ROC AUC values between 0.995 and 0.996 (Table 2). This marginal performance difference ($\Delta F1 < 0.012$) demonstrates that the framework's discriminative capacity is not dependent on a particular threshold choice. Table 6 contextualizes these results within the broader literature, showing that the Extra Trees and Stacking models achieve the highest reported accuracy and ROC AUC values among comparable studies.

Table 6. Comparison with Related Studies

Reference	Methods	Dataset	Best Accuracy	ROC AUC
Bhatt et al. (2023)	SVM, RF, LR	Used cars (India)	0.88	-
Dhabe (2023)	MLR	Used cars (India)	0.81	-
Dutulescu et al. (2023)	Deep Learning	Used cars	0.92	-
Uluturk (2024)	RF	Used cars (Turkey)	0.93	-
Nandhini et al. (2024)	DNN, CNN	Used cars (multi)	0.91	0.94
Fayyaz et al. (2025)	XGBoost, RF	Pre-owned cars	0.94	-
This study - ET	Extra Trees	Used cars (Kaggle)	0.964	0.996
This study - Stacking	Stacking	Used cars (Kaggle)	0.963	0.995

The proposed classification framework has direct practical implications for decision-support applications in dealership pricing, vehicle insurance evaluation, loan risk scoring, and market analysis. By generating categorical price-level predictions rather than exact numerical estimates, the framework produces outputs that are more readily interpretable and actionable for non-technical stakeholders. These findings support the broader argument that binary classification approaches can serve as valuable complements to regression-based price estimation in applied automotive analytics.

5. CONCLUSION

This chapter applied machine learning algorithms to classify used vehicle prices into low- and high-price categories using a real-world dataset of 10,000 records from the Turkish market. The proposed threshold-based binary classification framework consistently outperformed regression-oriented benchmarks in interpretability and decision relevance, while ensemble-based methods-particularly Extra Trees and Stacking-demonstrated the strongest predictive performance across all evaluation metrics.

The Extra Trees and Stacking models achieved accuracy values of approximately 96% and ROC AUC scores exceeding 0.99, with five-fold cross-validation confirming the stability and generalizability of these results. The negligible gaps between cross-validation and test set accuracies (≤ 0.001 for both top models) rule out overfitting as an alternative explanation for the strong results. Year of manufacture, mileage, and engine displacement were consistently identified as the most influential predictors through both Gini-based feature importance and SHAP analyses, providing theoretically coherent and practically interpretable insights into the drivers of used vehicle price levels.

The sensitivity analysis across Q1, median, and Q3 thresholds confirmed that the classification framework retains robust performance under both balanced and imbalanced class distributions, addressing a common methodological concern in binary price classification research. The Naïve Bayes model's comparatively lower performance reflects the limitations of the feature independence assumption, while the Decision Tree's higher overfitting tendency underlines the advantages of variance-reducing ensemble strategies.

Several limitations of this study merit acknowledgment. The dataset is sourced from a single open-access Kaggle repository, which may contain reporting errors, sampling biases, and omissions inherent in user-generated aggregated data. The dataset primarily reflects Turkish market conditions over a specific time window, limiting the generalizability of findings to other geographic regions or temporal periods. Future research should incorporate validated industry datasets, macroeconomic indicators, and temporal variables to enhance robustness and real-world applicability. Additionally, hybrid and deep learning-based architectures may be explored to further improve model generalization in more complex or large-scale settings.

REFERENCES

Bhatnagar, P., Gururaj, H. L., Shreyas, J., Flammini, F., & Gautam, S. (2024). An analysis of car price prediction using machine learning. In Proceedings of the 2024 ACM International Conference on Data Science and Machine Learning (pp. 1–6). ACM. https://doi.org/10.1145/3674029.3674032

Bhatt, N. S., Pandey, T. N., Reddy, S. R., Jayasurya, B., Dash, B. B., & Patra, S. S. (2023). An empirical analysis of machine learning algorithms for used car price prediction system. In Proceedings of the 2023 Global Conference on Information Technology and Computing (GCITC) (pp. 1–6).IEEE. https://doi.org/10.1109/gcitc60406.2023.10426270

Chawhan, H. K. (2024). Prediction of selling price of various cars. Global International Multidisciplinary Research Journal,12(8),1–6. https://doi.org/10.69758/gimrj2406i8v12p099

Chen, J., Li, F., Xu, J., Wang, Q., Han, Q., & Yan, M. (2022). Comparisons of different methods used for second-hand car price prediction. In Proceedings of the 2nd International Conference on Applied Mathematics, Modelling, and Intelligent Computing (CAMMIC 2022), Proceedings of SPIE (Vol. 12259, p. 122594N). SPIE. https://doi.org/10.1117/12.2638739

Cui, B., Ye, Z., Zhao, H., Renqing, Z., Meng, L., & Yang, Y. (2022). Used car price prediction based on the iterative framework of XGBoost+LightGBM. Electronics, 11(18), 2932. https://doi.org/10.3390/electronics11182932

Dhabe, P. S. (2023). Used car price prediction using multiple linear regression. International Journal for Science

Technology and Engineering, 11(5), 2752–2755. https://doi.org/10.22214/ijraset.2023.48948

Dutulescu, A., Catruna, A., Ruseti, S., Iorga, D., Ghita, V., Neagu, L. M., & Dascalu, M. (2023). Car price quotes driven by data: Comprehensive predictions grounded in deep learning techniques. Electronics, 12(14), 3083. https://doi.org/10.3390/electronics12143083

Fagbamila, J. E., Agbaje, A. A., & Okubadejo, G. O. (2024). Predictive analytics for pricing strategy in the automobile industry using machine learning models. World Journal of Advanced Research and Reviews, 24(3), 3543–3550. https://doi.org/10.30574/wjarr.2024.24.3.3920

Fayyaz, I., Ali, G. G. Md. N., & Khairunnesa, S. S. (2025). Advanced feature engineering and machine learning techniques for high accurate price prediction of heterogeneous pre-own cars. Vehicles, 7(3), 94. https://doi.org/10.3390/vehicles7030094

Gleue, C., Eilers, D., von Mettenheim, H.-J., & Breitner, M. H. (2019). Decision support for the automotive industry. Business & Information Systems Engineering, 61(4), 385–397. https://doi.org/10.1007/s12599-018-0527-3

Gohil, V. K. (2025). Car price prediction by machine learning approach. Indian Journal of Scientific Research in Engineering and Management, 9(9), 1–9. https://doi.org/10.55041/ijsrem52586

Gupta, R., Sharma, A., Anand, V., & Gupta, S. (2022). Automobile price prediction using regression models. In Proceedings of the 2022 International Conference on Inventive and Computation Technologies (ICICT) (pp. 410–416). IEEE. https://doi.org/10.1109/ICICT54344.2022.9850657

Hemendiran, B., & Renjith, P. N. (2023). Predicting the prices of the used cars using machine learning for resale. In Proceedings of the 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) (pp. 1–5). IEEE. https://doi.org/10.1109/SCEECS57921.2023.10063133

Jing, H., Ye, X., & Manoharan, S. (2022). Residual value prediction. In Proceedings of the 2022 IEEE International Conference on Computing (ICOCO) (pp. 403–408). IEEE. https://doi.org/10.1109/icoco56118.2022.10031995

John, S., Puttaswamy, M. R., Rao, V., & Ajmi, Z. H. H. M. (2025). Data-driven used car price prediction with tuned gradient boosting models. In Proceedings of the International Conference on Modeling, Simulation & Intelligent Computing (MoSICom) (pp. 593–598). IEEE. https://doi.org/10.1109/MoSICom67153.2025.11398299

Kasar, A., Jadav, D., Tripathi, M. M., & Govil, M. (2025). Integrating macroeconomic indicators into machine learning models for used car price prediction. European Economic Letters, 15(3), 2202–2214. https://doi.org/10.52783/eel.v15i3.3651

Kenny, J., Mandal, S., Mehta, R., & Malpekar, H. (2021). Price prediction using machine learning methods. International Journal for Research in Applied Science and Engineering Technology, 9(5), 661–668. https://doi.org/10.22214/IJRASET.2021.34259

Liaw, A., & Wiener, M. (2001). Classification and regression by randomForest. R News, 2(3), 18–22.

Monburinon, N., Chertchom, P., Kaewkiriya, T., Rungpheung, S., Buya, S., & Boonpou, P. (2018). Prediction of prices for

used car by using regression models. In Proceedings of the 2018 5th International Conference on Business and Industrial Research (ICBIR) (pp. 115–119). IEEE. https://doi.org/10.1109/ICBIR.2018.8391177

Msiza, Z. A., & Owolawi, P. A. (2023). Advancing used car price prediction in South Africa: An empirical examination of machine learning techniques. In Proceedings of the 2023 International Conference on Artificial Intelligence and its Applications (ICARTI) (pp. 189–196). https://doi.org/10.59200/icarti.2023.027

Nalamothu, P. P. (2023). Comparative analysis of regression models for price prediction of ride-on-demand services. International Journal for Science Technology and Engineering, 11(5), 1687–1700. https://doi.org/10.22214/ijraset.2023.51770

Nandan, M., & Ghosh, D. (2023). Pre-owned car price prediction by employing machine learning techniques. Journal of Decision Analytics and Intelligent Computing, 3(1), 167–184. https://doi.org/10.31181/jdaic10008102023n

Nandhini, R., Madhurajyothi, S., Mahaprabhu, M., Jessica, M., & Malliga, S. (2024). Improving accuracy of used car price prediction using machine learning models. In S. Malliga & T. Sasilatha (Eds.), Machine learning and IoT: Strategies, applications, and challenges (pp. 71–90). Boca Raton, FL: CRC Press. https://doi.org/10.1201/9781003768654-6

Öz, M. (2024). Car price dataset [Data set]. Kaggle. https://www.kaggle.com/datasets/mustafaoz158/car-price/data

Pias, M. A. A. M., Sartaz, M. S., Mazumder, A., Ahad, A., Islam, M. M., Islam, D. A., . . . Ahmed, S. (2025). Optimizing

used car valuation with AI: A predictive modeling approach. In Proceedings of the 2025 IEEE 4th International Conference on Computing and Machine Intelligence (ICMI) (pp. 01–07). IEEE. https://doi.org/10.1109/ICMI65310.2025.11141166

Raj, G. P., George, G., Hasan, H., Delmon, J. A., & Jose, L. (2025). Enhanced used car price prediction using machine learning: A comparative study of regression models. In Proceedings of the 2025 International Conference on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE) (pp. 1–5). IEEE. https://doi.org/10.1109/amathe65477.2025.11081288

Saini, P. S., & Rani, L. (2023). Performance evaluation of popular machine learning models for used car price prediction. In N. Chaki, N. D. Roy, P. Debnath, & K. Saeed (Eds.), Proceedings of International Conference on Data Analytics and Insights, ICDAI 2023. Lecture Notes in Networks and Systems (Vol. 727, pp. 589–599). Singapore: Springer. https://doi.org/10.1007/978-981-99-3878-0_49

Uluturk, S. (2024). Regression analysis for predicting prices of used cars: A study utilizing data from car trading website (Unpublished master's thesis). National College of Ireland, Dublin, Ireland.

Yılmaz, Ü. (2025). Predicting honeybee colony health using weather and apiary data with machine learning. Türk Mühendislik Araştırma ve Eğitimi Dergisi, 4(2), 140–160. https://izlik.org/JA87TJ95UA

Yılmaz, Ü. (2026). Benchmarking machine learning models for transaction-level anti-money laundering under extreme class imbalance. Mühendislik Bilimleri ve Araştırmaları

Dergisi, 8(1), 55–66. https://doi.org/10.46387/bjesr.1848543

Yılmaz, Ü., & Özçekiç, E. (2025). Classifying liver disease with boosting machine learning approaches. Eskişehir Osmangazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi, 33(2), 1882–1892. https://doi.org/10.31796/ogummf.1591951

BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ ALANINDA

AKADEMİK TARTIŞMALAR

yaz
yayınları

YAZ Yayınları
M.İhtisas OSB Mah. 4A Cad. No:3/3
İscehisar / AFYONKARAHİSAR
Tel : (0 531) 880 92 99
yazyayinlari@gmail.com • www.yazyayinlari.com