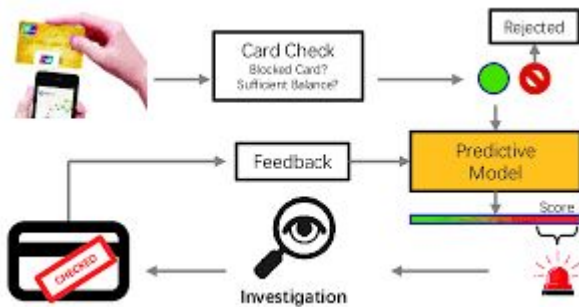


Credit Card Fraud Detection using Machine Learning

Aaron Zhong

River Hill High School

July 24, 2025



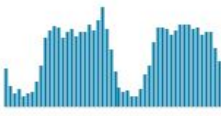




Problem Statement

- I am trying to detect credit card fraud using machine learning.
- It is important because credit card fraud can lead to massive problems in the real world like stolen money.
- Credit card fraud is a major global issue and costs billions of dollars every year. To address this, companies rely on machine learning models, which can analyze a lot of transaction data in real time, to learn complex patterns and detect anomalies. This project shows the use of machine learning to accurately identify fraudulent credit card transactions.

Data Description

- Kaggle Credit Card Fraud Detection Dataset
 - <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
- 284,807 transactions (rows), 31 (columns)
- Time, Amount, V1-V28, Class (Binary Label)
- No missing values, some outliers, severe majority class dominance

Data Description

About this file Suggest Edits					
BigQuery Table bigquery-public-data.fraud_detection.comments					
# Time	# V1	# V2	# V3	# V4	# V5
Number of seconds elapsed between this transaction and the first transaction in the dataset	may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)				
					
0 173k	-56.4 2.45	-72.7 22.1	-48.3 9.38	-5.68 16.9	-114
0	-1.3598071336738	-0.0727811733098497	2.53634673796914	1.37815522427443	-0.33832
0	1.19185711131486	0.26615071205963	0.16648011335321	0.448154078460911	0.060017
1	-1.35835406159823	-1.34016307473609	1.77320934263119	0.379779593034328	-0.50319
1	-0.966271711572087	-0.185226008082898	1.79299333957872	-0.863291275036453	-0.01030
2	-1.15823309349523	0.877736754848451	1.548717846511	0.403033933955121	-0.40719
2	-0.425965884412454	0.960523044882985	1.14110934232219	-0.168252079760302	0.420986
4	1.22965763450793	0.141003507049326	0.0453707735899449	1.20261273673594	0.191880
7	-0.644269442348146	1.41796354547385	1.0743803763556	-0.492199018495015	0.948934
7	-0.89428608220282	0.286157196276544	-0.113192212729871	-0.271526130088604	2.669598

Data Analysis

- Summary statistics
 - Shape: 284,807 transactions x 31 columns (28 PCA features, Time, Amount, Class)
 - NO missing values
 - Target Balance: 492 fraud, 284,315 legit
 - Amount: Skewed right with high variance
 - Time: Uniform across 48 hour window with few peaks
 - PCA Features: Centered distribution
- Visualizations:
 - Time Distribution: Some transaction time spikes coincide with fraud clusters
 - Correlation Matrix: PCA's features show low pairwise correlation

Data Analysis

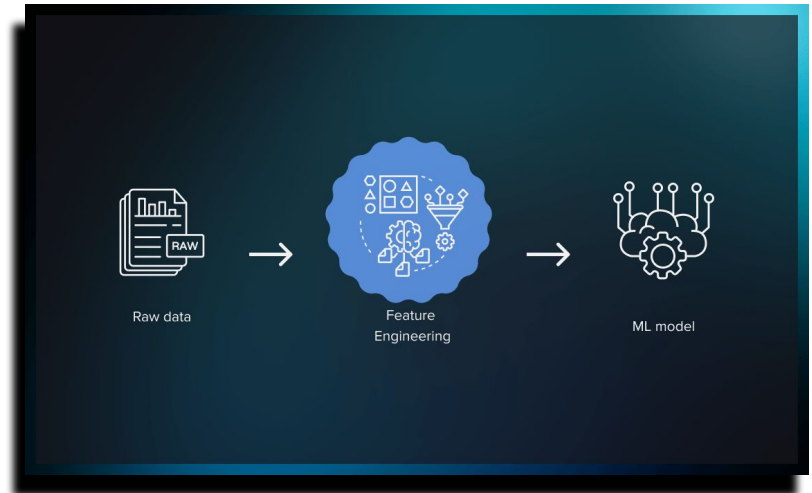
- Severe class imbalance
 - Led to use of oversampling
 - Guided selection of performance metrics like Precision Recall AUC
- High skew and outliers in Amount:
 - Motivated log transformation or robust scaling
 - Confirmed via histograms
- Time feature is not as informative:
 - Low correlation with Class
- PCA-transformed features optimal for modeling:
 - Low multicollinearity, normalized distributions, ideal for algorithms

Feature Engineering

- Feature transformation and encoding
 - PCA Applied (V1-V28)
 - The 28 anonymized features are already PCA-transformed
 - Standardized Amount and Time features using StandardScaler
- No new features created
- Feature Scaling:
 - Amount and Time were scaled
 - $\text{mean} = 0$ and $\text{std} = 1$

Feature Engineering

- No feature reduction
- Final Feature Set:
 - Time
 - Scaled Amount
 - PCA Features
 - Eg: V4, V11, V12
 - Class
 - 0 for legit transactions
 - 1 for fraud transactions



Model Selection

- What machine learning models did you use?
 - Logistic Regression
 - Fundamental linear model that estimates the probability of a binary outcome
 - Random Forest Classifier
 - An ensemble method that reduces overfitting and handles imbalance datasets well
 - K-Nearest Neighbors
 - A simple, instance based learning algorithm that can capture complex decision boundaries

Model Selection

- Hyperparameters:
 - Logistic Regression
 - Max_iter: max iterations for solver to converge
 - solver: optimization algorithm
 - Random Forest Classifier
 - n_estimators: numbers of trees in forest
 - max_depth: max depth of each tree
 - min_samples_split: min samples required to split an internal node
 - K-Nearest Neighbors
 - n_neighbors: num of neighbors to consider
 - weights: how neighbors contribute to prediction
 - algorithm: algorithm for nearest neighbors search

Model Evaluation

- Metrics:
 - Classification Report
 - includes precision, recall, f1-score, and support for both classes
 - Confusion Matrix
 - Shows true positives, true negatives, false positives, and false negatives
 - ROC Curve and AUC
 - Plots true positive rate vs false positive rate at diff thresholds
 - Precision-Recall Curve and Average Precision
 - Useful for imbalance data
 - Precision and Recall vs Threshold Plot
 - Shows how precision and recall change

Model Evaluation

Fold 1:

```
Fold 1
Classification Report:
              precision    recall  f1-score   support

     0       1.00      0.98      0.99     94772
     1       0.07      0.91      0.12        164

 accuracy          0.98     94936
 macro avg          0.53     94936
 weighted avg       1.00     94936

Confusion Matrix:
[[92662  2110]
 [   15  149]]
```

Fold 3:

```
Fold 3
Classification Report:
              precision    recall  f1-score   support

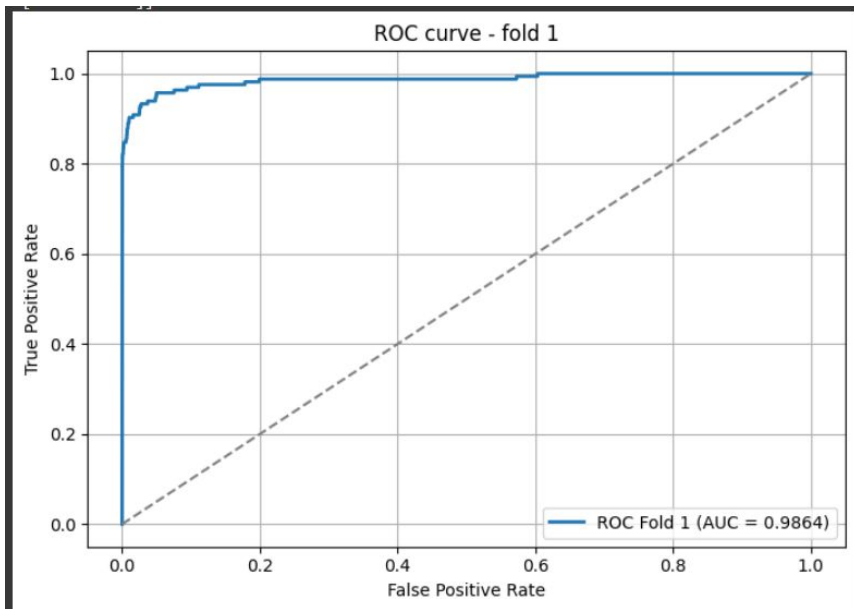
     0       1.00      0.98      0.99     94771
     1       0.07      0.89      0.13        164

 accuracy          0.98     94935
 macro avg          0.53     94935
 weighted avg       1.00     94935

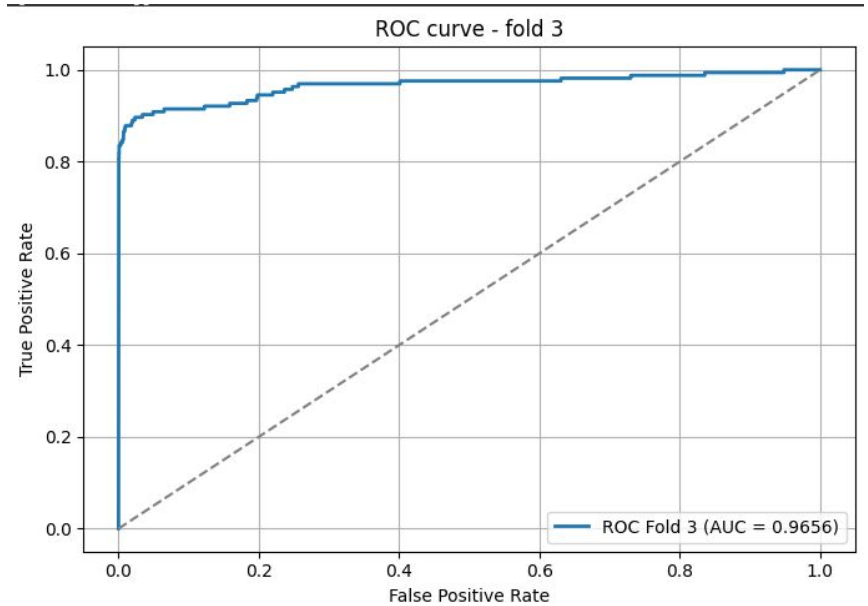
Confusion Matrix:
[[92810  1961]
 [   18  146]]
```

Model Evaluation

Fold 1:

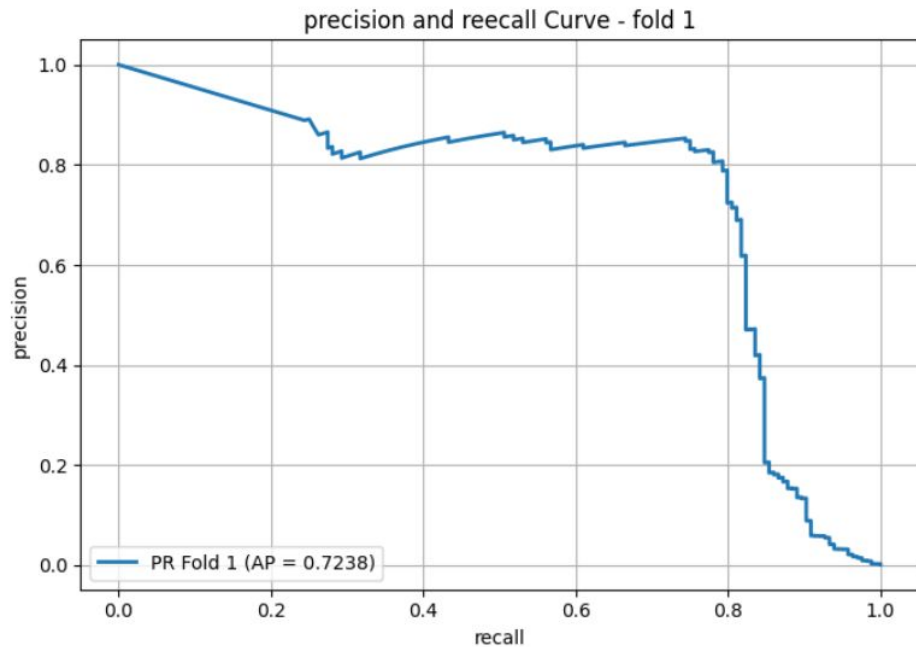


Fold 3:

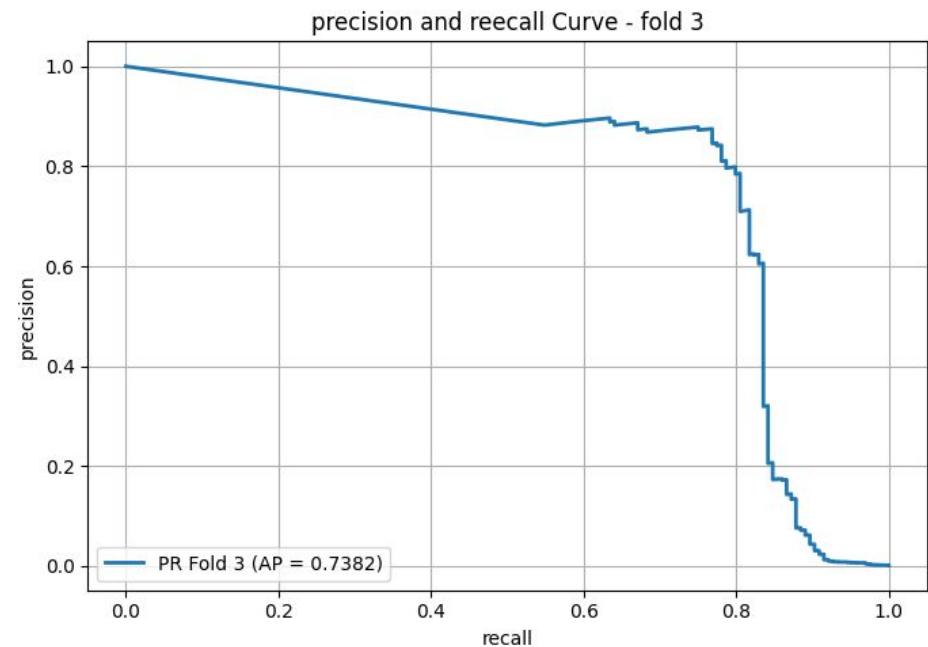


Model Evaluation

Fold 1:

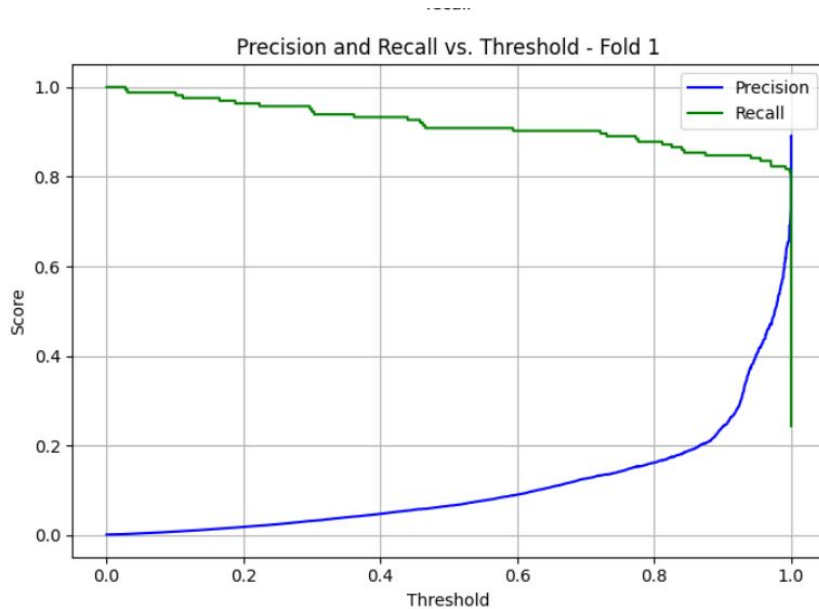


Fold 3:

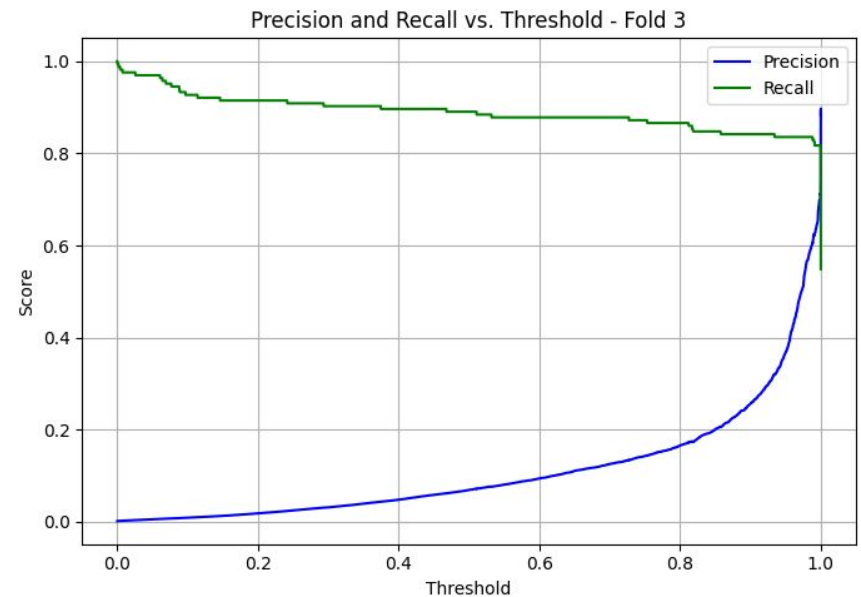


Model Evaluation

Fold 1:

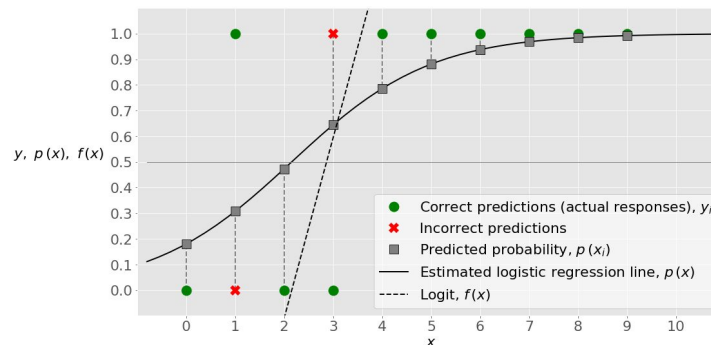


Fold 3:



Model Evaluation

- I believe Logistic Regression wins because:
 - It balances interpretability and prediction
 - Uses data resampling to address class imbalance
 - Achieves high ROC AUC (0.98) and high Average Precision (0.85)
 - Shows good discrimination between fraud and non fraud cases



Conclusion

- Key Takeaways:
 - Upsampling the minority class significantly improved model performance
 - Logistic Regression performed reliably with good interpretability and high precision
 - Evaluation using ROC and precision recall curves helped confirm model effectiveness
 - Data preprocessing is very important for consistent results

Conclusion

- I have learned:
 - How to handle imbalanced classification problems using techniques like upsampling
 - How to implement and compare machine learning models like Logistic Regression and Random Forest Classifier
 - The importance of evaluation metrics beyond accuracy
 - How to use tools like cross-validation, feature scaling, and visualization in real-world datasets

Conclusion

- Challenges:
 - Dataset was imbalance
 - Solved using upsampling to balance the training data
 - Choosing the right evaluation metrics
 - Learned to use ROC AUC and Precision recall curves instead of just accuracy
 - Avoiding overfitting
 - Used cross-validation and regularization to build generalizable models

The End

