Artificial Intelligence in Software Automation

A touch of AI reasoning, neural networks problem solving, and ML techniques aiming to efficiently perform software test automation. Optimize outputs, and move on from yesterday's laborious methods for development and testing

How Bytelabs developed TIK, a machine learning algorithm for quality automation process in software testing – Brian Grieve.

INTRO

Automation framework nowadays

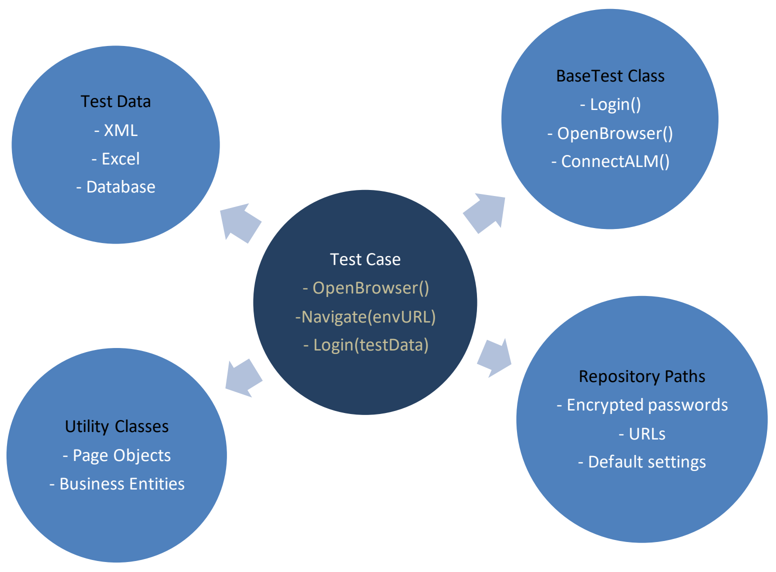
Bytelabs has built automation frameworks using industry best practices for a number of clients. Our strategy uses a combination of the Page Object Model design pattern with higher-level, object-oriented concepts of computer programming. We use strategies such as Pattern Recognition for clustering objects of similar properties, and then abstract different classes to make use of concepts such as inheritance and polymorphism.

The first step in building our automation framework is to build an Object Repository using the Page Object Model design pattern. Instead of opting for the simpler browser recording of objects provided by various easy-to-use automation tools, we analyse the presentation layer of the application (usually displayed as HTML) and build unique identifiers for each object using Xpath. Through this approach we construct an Object Repository that is more reliable on the long term, as the repositioning of objects during the development process does not require any modification to the Object Repository.

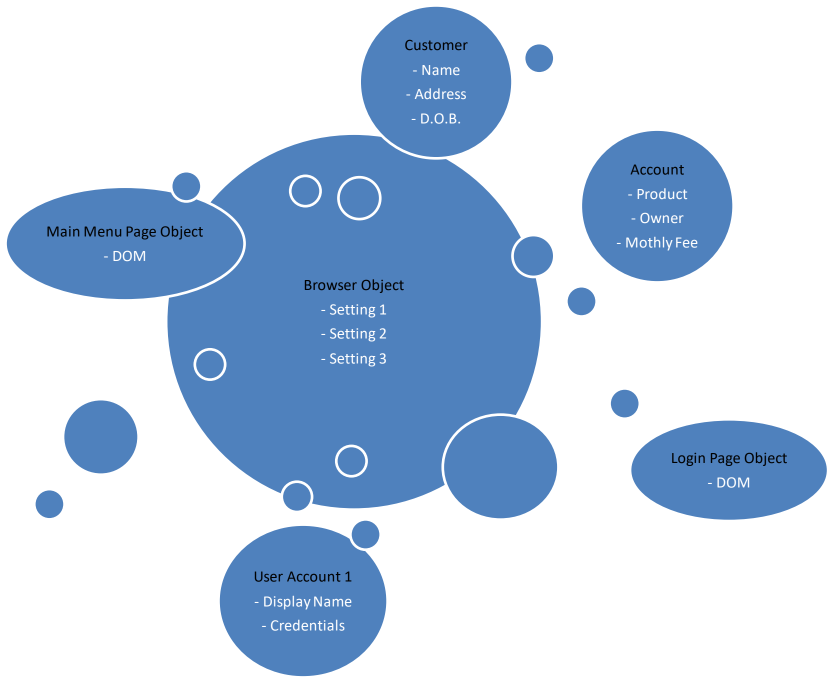
As we build the repository, we identify opportunities for optimization when we recognise different patterns in the way objects are organised in the presentation layer. We then use this pattern recognition to cluster objects into classes, using properties to differentiate between each instance we find. This allows us to easily interact with future objects that may be added, as they will likely fit within one of the class patterns we identified.

Once the Object Repository is ready, we are ready to write test scripts that create instances of these objects, so that they interact with them on the presentation layer, e.g. clicking of buttons, filling of forms, and selection of items.

Below is a graphic representation of the structure of a sample framework:



At runtime, a number of instances of these objects get created. They hold information about the entities they relate to, and get destroyed once execution is complete:



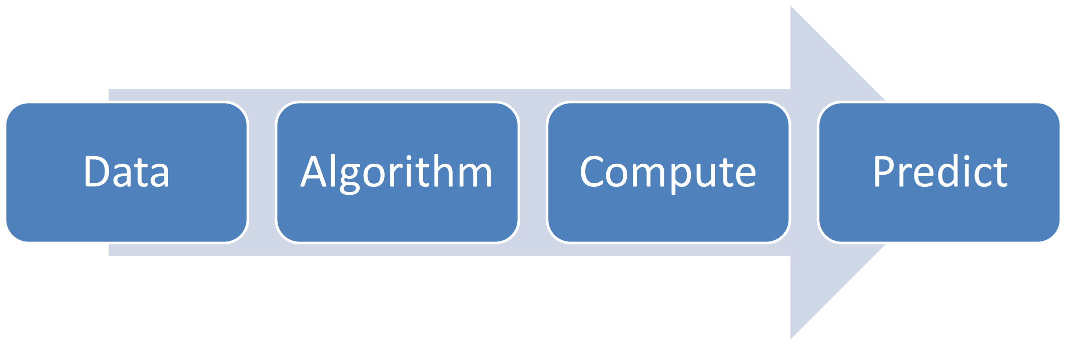
Machine Learning

What is Artificial Intelligence and Machine Learning (ML)?

One of the definitions for *Artificial Intelligence* is exemplified through any device perceiving its environment and taking actions to maximize its chances of successfully achieving goals.

Basically, the machine mimics 'cognitive' human functions such as learning and problem solving, that's why we call this process **Machine Learning**.

AI is achieved with the help of data to understand and replicate patterns.



Machine learning algorithms maximize human processes by understanding and running multiple testing scenarios in order to prevent system crashes or, in some cases, unauthorized sensitive information access.

We normally start by dividing the dataset into two (train and test) or three parts (train, test and validation). In *training data* we run all types of training algorithms, after we clean and pre-process data creating features using statistics and mathematics calculations, sometimes even borrowing some techniques from other sciences such as physics, genetics, biology, etc. to understand patterns. After running multiple algorithms, we check reports for accuracy, precision, recall and F1-score, cross-validation and loss functions when required to make sure our model is performing as expected.

However, if the model is not successful the first time; It is possible to get a bigger slice of the dataset, apply hyper-parameter tuning techniques, use activation functions, removing bias and assigning weight, control hidden layers in order to control under or overfitting when using neural networks, etc. all to get better report results. That is why ML is so powerful.

With Machine Learning, Data Scientists get the work done!

What has been done in the market thanks to ML?

1. Airports, Facebook - face recognition;
2. Stock market – regression predictive models;
3. Nintendo wii, Kinect sports and gaming - random forest classifier;
4. Virtual reality headset;
5. Alexa, Siri, Echo - Voice recognition, speech to text and vice-versa (translations apps);
6. Robots dogs – reinforcement learning from Neural Network techniques;
7. Facebook ads - learning behaviour , recommendation engine (SEO, Ads);
8. Amazon, Netflix, Audible, Spotify … - Recommender engine;
9. Medicine, cure of diseases, vaccines - predictive techniques on supervised models;
10. Fraud detection, credit analysis – customer behaviour methods on variables (age, gender …) and regression techniques;
11. Google maps … - image identification by pixels, neural networks;
12. Uber … - Self-drive cars, ensemble methods;
13. Etc.

The list goes on and on …

ML techniques

Supervised Learning:

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output.

$$y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (X) you predict the output variable (y) for that data.

It is called supervised learning because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. As we know the correct answers, the algorithm iteratively makes predictions on the training data and it is corrected by the 'teacher'. Learning step stops when the algorithm achieves an acceptable level of performance.

Supervised learning is further grouped into **Regression** and **Classification**:

Regression or quantity prediction. It is a ML technique that is trained to predict real numbered outputs like temperature, stock price, etc. It is based on a hypothesis (linear, quadratic, polynomial, non-linear, etc.). The hypothesis is a function based on some hidden parameters and input values. The process that does the optimization is the gradient descent algorithm. If we are using neural networks, then we might also opt to use backpropagation fluxes to compute gradient at each layer. Once the hypothesis parameters are trained (when they gave least error during the training), then the same hypothesis with the trained parameters are used with new input values to predict outcomes that will be again real values.

Classification or label prediction. It is the process of predicting the class of given data points. Classes are sometimes called as targets, labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

Some popular examples of supervised machine learning algorithms are:

- Linear regression for regression problems;
- Random forest for classification and regression problems;
- Support vector machines (SVM), k-Nearest Neighbour (KNN) for classification problems.

Unsupervised Learning

Unsupervised learning is a type of ML algorithm used to draw inferences from datasets consisting of input data without a labeled response. In other words, unsupervised learning is where you only have input data (x) and no corresponding output variables. The goal is to model the underlying structure or distribution in the data in order to learn more about the data by grouping elements with similar characteristics.

These are called unsupervised learning because, unlike supervised learning above, there are no correct answers and there is no 'teacher'. Algorithms are left to their own devises to discover and present the interesting structure in the data. Unsupervised learning problems are further grouped into **Clustering** and **Association** problems:

Clustering – A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour. The most common unsupervised learning method is called cluster analysis, which is used for exploratory data analysis to find hidden patterns. A cluster is modeled using measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance.

Some popular examples of unsupervised learning algorithms are:

- Hierarchical clustering to build a multilevel hierarchy of clusters by creating a cluster tree;
- K-means for clustering problems;
- Gaussian mixture models to model clusters as a mixture of multivariate normal density components;
- Self-organizing maps that use neural networks to learn the topology and distribution of the data;
- Hidden Markov models that uses observed data to recover the sequence of states
- Apriori algorithm for association rule learning problems.

Unsupervised learning methods are used in bioinformatics for sequence analysis and genetic clustering, in data mining for sequence and pattern mining, in medical imaging for image segmentation, and in computer vision for object recognition.

Semi-supervised Learning

Association – Association rule mining was originally classified as unsupervised learning but modern studies show many applications on supervised learning as well, so that generated an intersection class called semi-supervised learning. It mostly works with discrete (categorical) data; however, you discretize any continuous attributes during the pre-processing stage. An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

It is called Semi-Supervised because we mix Clustering and Association solutions from Supervised and Unsupervised Machine Learning techniques. We want to discover inherent groupings in the data through cluster analysis, which is used to find hidden patterns of groups. The clusters are modeled using measures of similarity which is defined upon metrics such as Euclidean or Probabilistic distance altogether with Association Rule Mining to generate an intersection class that works with discrete categorical data; however, we encoded categorical attributes during the pre-processing stage and used as a mixture of multivariate normal density components.

Sample Scenarios

This document will illustrate applicable cases on 3 different industries we used in our project to test different environments and website configurations.

Toronto Machine Learning Summit

TMLS is a collaborative event for the Canadian Machine Learning community comprised of over 6,000 ML researchers, professionals and entrepreneurs across several disciplines.

The TMLS initiative is dedicated to helping promote the development of AI/ML effectively, and responsibly across all Canadian Industries. As well, to help data practitioners, researchers and students fast-track their learning process and develop rewarding careers in the field of ML and AI.

We Cloud Data

Founded in Toronto, WeCloudData assembled top talent from famous data-driven companies in North America to design and teach the most comprehensive and practical data science and data engineering courses that set up students for success in the real world.

Orange HRM

OrangeHRM is the most popular open source human resource management (HRM) software in the world. Offering choice, flexibility, and affordability, Orange HRM gives three products: Open source, professional and enterprise.

Bytelabs Solution

Organizations face time crunches and almost every client we have worked with wants to increase test automation continuously. Doing that requires not only a balanced approach, but also a smarter starting plan. What we offer is a robust automation framework that outlives its implementation engagement by years with minimal maintenance required.

TIK is a pioneering delivered solution that will have the ability to self-learn and test applications without much human intervention beyond the implementation stage, making it truly autonomous and intelligent. TIK will help our clients improve agility and predictability, while optimizing efforts in testing by integrating AI techniques to test automation. Additionally, TIK as a solution is a Machine learning project for designing and executing a runtime model inspired by KNN methodology. We feed Document Object Model (DOM) instances of the application under test as input, and samples recognized patterns within it as outputs to TIK. The library is continuously evolving to pre-recognise patterns by building class structures.

We used a Semi-Supervised ML technique for pattern recognition where we teach the algorithm to label the target – element class in this case - every time it fulfills the requirements by following a pattern through a mapping function from the input to the output.

As we are dealing with a multiclass case, the model we choose is a multivariate Bernoulli Naive Bayes classifier with the assumption of conditional independence between every pair of features given the value of the class variable. Naïve Bayes learners and classifiers are extremely fast in comparison to sophisticated methods by the mean that it estimates each distribution independently as one-dimensional distribution.

Many common pattern recognition algorithms are probabilistic as they use statistical inference to find the best label for a given instance. They also output a probability of the instance being described by the given label. Besides, probabilistic algorithms output a list of N-best labels with associated probabilities, for some value of N, instead of a simple single best label. This pattern recognition solution aims to provide a reasonable answer for all possible inputs performing the ‘most likely’ match of them, considering their statistical variation.

They have worked well in multiple real-world problems as they do not require huge amounts of training data to estimate the necessary parameters. On the other hand, it is needed to double-check manually the precision of our outputs by inspecting class elements on the website (as the goal of this project is focused on product management used for software test automation cases) to certify that our predictions are correctly classified accordingly.

STRATEGY

The project strategy has a simple approach: Combine the best advantages of AI within QA test automation, as described in three areas as follows:

1. Eliminate test coverage overlaps

- Mapping the whole application under test and storing the data (.js and .csv);
- Running TIK algorithms, we can compare and detect changes;
- Developing an action script updating test automation scripts.

2. Optimize efforts with predictable testing

- TIK's algorithms group elements into classes by identifying similarities among them;
- It is also able to predict in which "class grouping" a new element is going to be clustered.

3. Move to defect prevention instead of defect detection

- A part of Total Quality Management (TQM) strategy works on the prevention that comes reasoning with TIK detecting test automation issues that need to be updated after root cause defects identification, directing us to the exact location where they must be corrected.

Organizations are struggling to develop automation scripts with pattern analysis for processing huge volumes of data. TIK traverses the code during a software upgrade, to detect key changes in functionality and link them to requirements identifying test cases. It simultaneously optimizes tests and foresees the decisions on hot spots that would possibly lead to failure.

In our near future, we are going to present solutions that will use deep learning fundamentals to build a true autonomous approach to testing. It uses the same concept of self-driving cars; autonomous technologies will help generate their own scripts for testing by learning through the system.

ML will be applied on our existing framework on data steps such as collecting, cleansing, interpreting and modifying. The same principle is intended for abstracted entity objects such as User Accounts and Customers, for example, and in many other steps to optimize time, storage and energy.

The Project Goals

Most testing projects demand skilled testing teams for full-cycle functional, automation, performance and usability testing. While in a traditional web application, testers were just dealing with the POM, or the actual elements rendered on the page. Nevertheless, when you want to test “widget X” that goes on a specific page, it will become a challenge. “Widget X” might have other ‘layers’ and could be buried in the presentation layer of the application stack lacking identification, details, and scope. This is where the automation starts to break, and were AI starts becomes handy.

The Project goals are as follows:

- AI automation algorithm to aid software testing facilitating human interactions on development of automation suit;
- Development of an AI algorithm that captures the DOM to fasten root cause analysis;
- Annotation of AI algorithm that would extend OPEN source framework via a predictive PASS / FAIL status of each functional area of the application under test;
- Design of an algorithm that learns through each automation run, given an iteration or sprint;
- Development of an “Annotation” AI algorithm that can log software failures within a test automation platform (commercial or open-source).

We merge the data to a final decision that will guide humans to act based on the machine’s output. Human brains could get to exhaustion due to multiple scenarios and calculations, which is the reason why ML has the ability to process billions of data with different combinations of algorithms.

Machine Learning has, at it seems, a very complex process underneath where we teach a machine through artificial intelligence how to behave like a human, in other words, we create software that creates software. And for this, we study and predict all possible outputs to prevent software to crash.

We aim to achieve goals through an autonomous AI automation algorithm that can aid automated software testing.

The ideal team configuration is: an Executive Sponsor, a Lead Data Scientist, a Lead Senior Developer, a Feature Engineer, and a Machine Learning Engineer.

Phase 0 – Faster Pattern Recognition

The objective is the interpolation of Machine Learning into our framework, aiding the recognition of patterns when building Object Repositories. We build our library based in structures instead of individual elements, and an AI model helps to identify such elements prior to the design of the framework.

Inspired by KNN methodology, we feed Document Object Model (DOM) instances of the application under test as input, and samples of recognised patterns within it as output to TIK. The library is continuously evolving to cover more DOMs and TIK helps to pre-recognise patterns by building class structures.

An example of pattern recognition is classification, which attempts to assign each input value to one of given sets. However, pattern recognition is a more a general problem that encompasses other types of output as well. It aims to provide a reasonable answer for all possible inputs and to perform 'most likely' matching of them, taking into account their statistical variation. This is opposed to pattern matching algorithms, which look for exact matches in the input with pre-existing patterns.

Many common pattern recognition algorithms are probabilistic in nature, in that they use statistical inference to find the best label for a given instance. Unlike other algorithms, which simply output a 'best' label, often probabilistic algorithms also output a probability of the instance being described by the given label. In addition, many probabilistic algorithms output a list of *N*-best labels with associated probabilities, for some value of *N*, instead of a simply single best label.

In this project we use a Semi-Supervised ML technique for pattern recognition where we teach the algorithm to label the target – element in this case - every time it fulfills the requirements by following a pattern through a mapping function from the input to the output. We know the correct answers, and just like a teacher (Supervisor) we correct the algorithm iteratively after it makes a prediction based on the training data. The learning process stops when the algorithm achieves an acceptable level of performance.

Data Structure

Web Harvesting

Web harvesting is a technique that allows us to go to a client website and, by checking his DOM, collect data (html codes) in order to store and compare elements and properties.

In fact it maps the whole website and creates comparative frames, based on the DOM, so before we run a test next time for the same client, these web data extraction algorithms will tell us any sort of modifications that the DOM has suffered and then we update our test scripts exactly at the identified breaking point so when we run it again we do not expect a fail report with data on these elements and properties pointed out by the algorithm. Moreover, if a page fail it is most likely due to an unmapped new functionality on the website.

In this step, our main objective is the data collection. We gather and measure information in established systems on targeted elements to answer questions and evaluate outcomes. This stage is vital to all further steps, because with inaccurate data we cannot assure the integrity of results. The intention is to reduce the likelihood of errors, what is achieved with appropriate data collection instruments and delineated instructions.

Data Cleansing

Data cleansing is one of the most challenging steps as inaccuracy on our data compromises the whole project and time will be wasted. Refinement and understanding are fundamental in each and every step of this process.

From the previous steps we collected all desired data and now we are ready to go to the next one to make the data understandable to human processing before we convert the data frame to matrix ready for machine learning manipulation:

1) Data modelling – it is when we get rid of extra spaces, treat blank cells with missing data, smooth noisy data, convert numbers stored as object strings into integers or floats, remove duplicates and outliers, format text to upper/lower case, spell check, format collections and remove inconsistencies.

2) Data dimensionality reduction - An essential piece of data mining is efficient summarization and aggregation, this can be done by grouping the data and applying encoding schemes to obtain a compressed representation of it. We also use numerosity reduction where the data are replaced by smaller representation using parametric models such as log-linear or nonparametric models such as sampling. Using PCA (Principal Component Analysis) we search for the k n-dimensional orthogonal vectors that best

represent the dataset, where k<= n. It reveals relationships that were not previously suspected and thereby allows new interpretations.

3) Data Preprocessing – one of the data transformation strategies is the feature scaling or normalization method used to standardize the range of features of data. We scale it so it all fits in a scale range [0, 1]. In supported vector machine (SVM) and K-Nearest Neighbours (KNN), scaling is extremely important where the distance between data points matters. In scaling and normalization we are changing the range and shape respectively.

In this project, we use Bernoulli distribution - a specific statistical distribution where observations are closer and fall above and below the mean.

Feature Engineering

Feature engineering, as defined by Wikipedia, is the process of using domain knowledge of the data to create features that make machine learning algorithms work. It is fundamental to the application of ML, and it is both difficult and expensive.

Although DATA is the fuel of Machine Learning, Feature Engineering is the engine of it, which means it has to be refined into features - predictor variables - before being used to train a model. The process of extracting features from a raw dataset is called Feature Engineering.

In this ML pipeline, we build features for each label while filtering the data. The best approach is to hard-code for specific labels what would correspond to one customized problem solution. There is also another approach, such as the use of APIs library that is based on Deep Feature Synthesis which stacks primitive blocks to build features. This latter approach is called automated feature engineering and we must be careful in using the same formula as answers to different set of problems.

The process of modeling, training and validation is guided by the business needs. Dividing this process, we use reasoning to decide concepts and techniques that are useful for multiple scenarios in our dataset:

a. Generate features to represent the data when there are no features;
b. Generate effective features when existing features are not competitive enough;
c. Select features when there are too many features;
d. Generate and select effective features for specific types of applications;
e. Understand the challenges associated with various data types.

Designing Machine Learning algorithms

Splitting the data

In this stage, the data is already cleaned, normalized and ready to apply machine learning algorithms in the training part of it. We split the data into train and test (sometimes there is also a validation portion), so we train our model in approximately 3/5 of the total dataset before running it against the test data to do the so called *prediction* based on regression, classification or any other method we decided to use.

1) Splitting the data - This first step is quite simple, we apply a *train, test, split* function and associate columns (called features in Data Science) to X and y, as axis of a chart, to start building the algorithm;

2) Plotting data – Using a python library we are able to visualize, with publication quality, figures and interactive environments across platforms. Hence, we look for similarities in our grouped data to best define attributes we are going to cross. It is easily manipulated and we use diverse configurations to better understand correlation between our features and its values. Understanding patterns and natural clusters by plotting the data shows us natural cluster tendencies in the dataset, so patterns are identified and validated or de-correlated, indicating where our work tends to.

3) As we care about performance, we also build confusion matrix to identify where true positive and negatives are and if the data is fitting accordingly to created models:

    a) K-means is a partitioning algorithm mistakenly used as a clustering algorithm. As we attempt to minimize intra-partition distances we specify how many clusters are expected, another approach is to try different number of clusters, run K-means a few times and score each clustering result.

    b) Affinity propagation – it is a technique similar to K-means but it uses a graph based approach to let points ‘vote’ on their preferred ‘exemplar’. Furthermore, it does not need to specify the number of clusters, it allows non-metric dissimilarities (asymmetric) and it has also better stability over runs.

4) Defining statistical modeling – Statistical modeling is a simplified mathematical-formalized way to approximate reality and optionally make predictions from this approximation. Dependent and exploratory variables are represented by y and X in the axis modeling chart. Y is in the height plant and involves only one explanatory variable, which is quantitative. X is the independent variable used to explain, describe or predict the dependent variable y. In this case, parameters are the intercept and the slope is the link between dependent and explanatory variables; it also involves a third parameter, the variance of residual (errors) – distance between data points and the model, which is

represented by the straight line in the plant height linear regression – in a relation that the lower the residual the higher is the R^2 statistic.

height (Y) = intercept + slope * X

Building ML algorithms to apply on the train data

We call training a data sample, the bigger part of a split dataset which we use to train algorithms and prove their efficiency before applying to the test data. It is this chunk that leads us to obscure ways of deciphering what is lying underneath each one of the multiple layers that the machine uses to predict output (target in Data Science) after extensively calculation metrics.

The outputs of prediction are a set of label times, historical examples of what we want to predict, and features - predictor variables used to train a model is the core of it as it carries correlations and similarities within it.

The output from modeling is a trained model that we use for inference, making predictions on new data points. When dealing with numeric values such as stock market, credit score, fraud detection, etc. One of the choices is to apply regression algorithms (linear, ridge …) against a sample of the clean data, plot and observe how it behaves. Then we cross validate and expose with the expected output. In case of de-correlational bias, noise, weight or any interference error rates are going to increase, so we need to act and tune the model before training it again.

Checking results

Precision is a good measure to determine, when costs (loss function) of false positive is high.

Recall calculates how many of the actual positives our model captures through labelling it as a True Positive and, by this definition, we know that recall shall be the model metric we use to select our model when there is a high cost (loss) associated with False Negative.

F1 score is a function based on precision and recall; it is a balance between them both especially if there is an uneven class distribution (large number of True Negatives).

Confusion Matrix describes performance of classification models on a set of train data for which the values are known; we use these numbers to analyse results instead of arbitrary variables explained in terms of formula and sentences. In most cases, it is used in binary classifications (YES, NO) defined by true positives, true negatives and false positives, false negatives (TP, TN, FP, FN). It is also explained in Bayesian statistics from applied predictive modeling as the sensitivity and specificity as being the conditional probabilities, the prevalence is the prior, and the positive-negative predicted values are the posterior probabilities.

Plotting results the decision boundary, this is easily identified on a scatter plot where every point depicts a data point of the data set and axes are depicted from the features. It separates the data points into regions, which are actually classes in which they belong.

Activation functions

We use calculations based on multiple techniques to refine our methods for regression and classification or any other we want to work with and start remodeling, refitting, and retuning our chosen data sample if needed and until reports show better results.

We also might consider the need of getting a different portion of the sample; I would say the next options would be 75% (3/4) or even 67% (2/3) of the total dataset for a new run. One important thing to remember is that optimizing parameters on the aggregated data does not mean overfitting, cross validation is the only trustful method to check overfitting.

There are a lot of options between multiple combinations of methods, functions, formulas, bias, weights, models, variables, parameters, features and there is not only one possible way to solve a problem. We keep optimizing, brainstorming and running training against predictions to find out what works better for each specific case. In many empirical cases, local truth fits better than global truth.

Many activation functions are used to measure linearity / non-linearity in a model that were introduced to our network. Their main purpose is to convert an input signal of a node to an output signal.

Without an activation function neural network would not be able to learn and model other complicated kinds of data such as image, video, audio, speech, etc. They are useful to report any arbitrary complex functions which maps inputs to outputs, that is the reason why they are considered universal function approximators.

Some important activation functions such as Sigmoid or Logistic; tanH – hyperbolic tangent; ReLu – rectified linear units are useful options available.

In this project we did not need to activate it because we used hyperparameters tuning technique to optimize the target.

Testing the model

This is one of the last stages, where we finally attest model efficiency and validate previous reports against our test data results. In this phase, it is not expect many errors as models were trained to exhaustion due to early steps in the project pipeline. It is mostly running the algorithm against our data, observing how the algorithm is going to behave in a real world and comparing results of the output. Everything is documented, deeply checked and validated before being put in production.

Our conclusions are stored for posterior analysis in case of updates and is a base when we need to analyse the concept and steps we created to derive conclusions.

We believe that there is room for an AI-powered Software Development Lifecycle (SDLC) platform. As advances take place in computing power we could foresee, AI has the ability to recognize an entire project's infrastructure and progress more efficiently than a human.

While we can say that the project was a definitive marker of success in the evolutionary test automation framework landscape, we see that in a long-term run the objective ability of AI testing will further elevate AI-based automation in the DevOps age where iteration happens on the go.

AI will also help to generate codeless test automation. AI-based testing could identify mistaken requirements from the Business Requirement Document (BRD). This is the current direction and hope we hold for a new project development and on completion will ultimately be a pathbreaker for the QA software industry.

Results

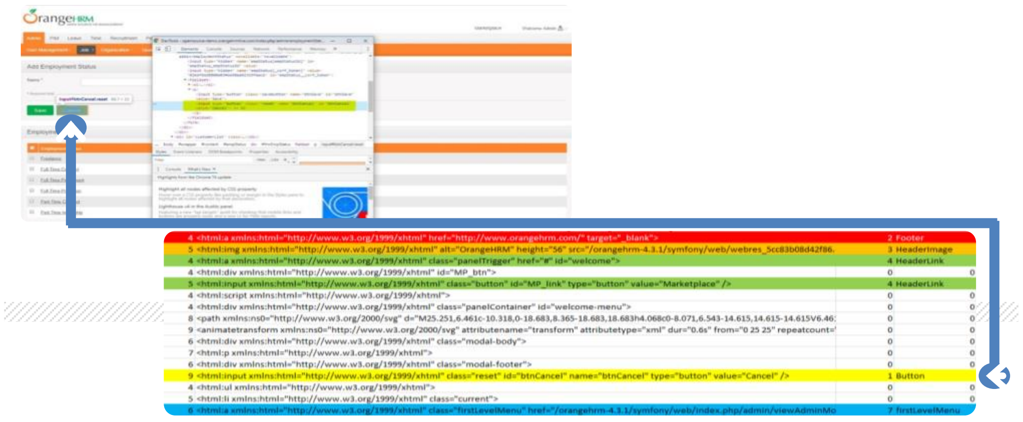
The primary focus of the project was to develop efficiencies into test automation by machine learning techniques. Additionally, bolstering up interpolation of artificial intelligence through machine-learning algorithms into open-source automation frameworks, when building object model-based repositories (POM & DOM).

The result goes beyond optimizing time and effort, especially on root cause triage reporting level which is the most time-consuming activity in Test Automation industry. At the latter part of our project, we discovered that we could effectively derive element type prediction(s) through cluster groups of similar objects, identifying hidden patterns that were deemed a black box for human identification. Additionally, we reduce root-cause analysis time by a factor of 72 % effectively saving the bank 10's of thousands of dollars monthly. Factoring the project duration of 2.5 year to date and still ongoing, we estimate an overall savings of 300,000.00 (CAD) within the QA budget.

As outlaid in our original aims and goals we were able to develop, complete and measure the following automation algorithm that can:

- Aid software testing without human interaction;
- Captures the DOM to Reduce root cause analysis:

Note the following screen captures are running examples of our TIK AI running with its successful screen output(s)



4	<html:a xmlns:html="http://www.w3.org/1999/xhtml" href="http://www.orangehrm.com/" target="_blank">	2	Footer
5	<html:img xmlns:html="http://www.w3.org/1999/xhtml" alt="OrangeHRM" height="56" src="/orangehrm-4.3.1/symfony/web/webres_5cc83b08d42f86.	3	HeaderImage
4	<html:a xmlns:html="http://www.w3.org/1999/xhtml" class="panelTrigger" href="#" id="welcome">	4	HeaderLink
4	<html:div xmlns:html="http://www.w3.org/1999/xhtml" id="MP_btn">	0	0
5	<html:input xmlns:html="http://www.w3.org/1999/xhtml" class="button" id="MP_link" type="button" value="Marketplace" />	4	HeaderLink
4	<html:script xmlns:html="http://www.w3.org/1999/xhtml">	0	0
4	<html:div xmlns:html="http://www.w3.org/1999/xhtml" class="panelContainer" id="welcome-menu">	0	0
8	<path xmlns:ns0="http://www.w3.org/2000/svg" d="M25.251,6.461c-10.318,0-18.683,8.365-18.683,18.683h4.068c0-8.071,6.543-14.615,14.615-14.615V6.46:	0	0
9	<animatetransform xmlns:ns0="http://www.w3.org/2000/svg" attributename="transform" attributetype="xml" dur="0.6s" from="0 25 25" repeatcount='	0	0
6	<html:div xmlns:html="http://www.w3.org/1999/xhtml" class="modal-body">	0	0
7	<html:p xmlns:html="http://www.w3.org/1999/xhtml">	0	0
6	<html:div xmlns:html="http://www.w3.org/1999/xhtml" class="modal-footer">	0	0
9	<html:input xmlns:html="http://www.w3.org/1999/xhtml" class="reset" id="btnCancel" name="btnCancel" type="button" value="Cancel" />	1	Button
4	<html:ul xmlns:html="http://www.w3.org/1999/xhtml">	0	0
5	<html:li xmlns:html="http://www.w3.org/1999/xhtml" class="current">	0	0
6	<html:a xmlns:html="http://www.w3.org/1999/xhtml" class="firstLevelMenu" href="/orangehrm-4.3.1/symfony/web/index.php/admin/viewAdminMo	7	firstLevelMenu

Figure 3 – Sample result (pass fail) Element type identification, cluster and prediction (ỹ)

- Root cause showing attribute change and new mandatory field:

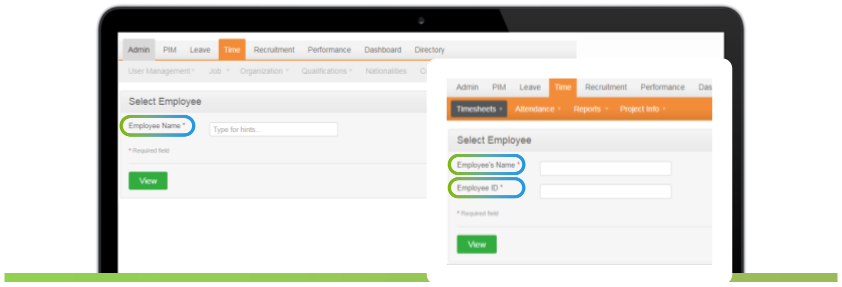
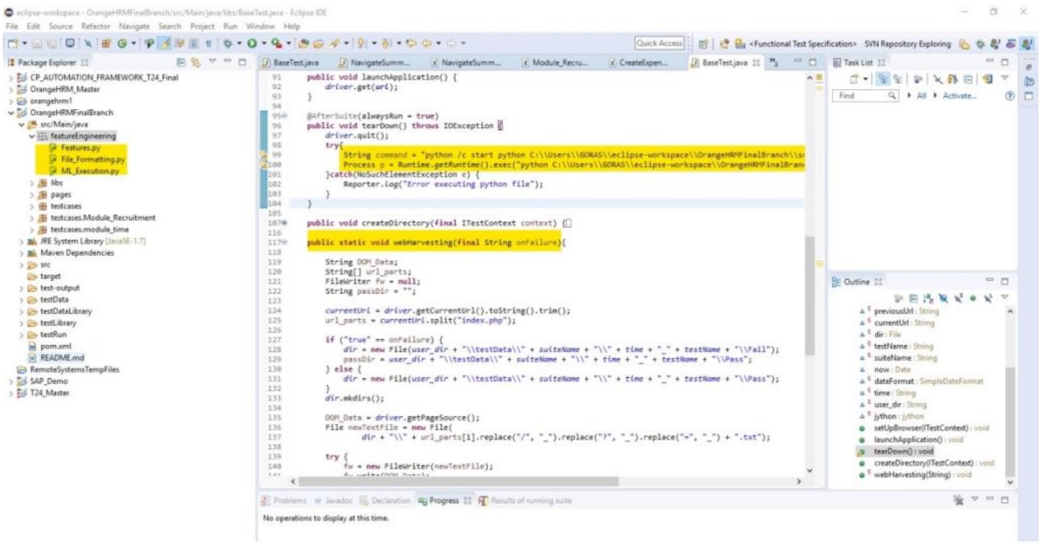


Figure 4 Root cause triage AI prediction

- Designing of AI algorithm via a predictive PASS / FAIL status;

- Log software failures within a test automation platform:

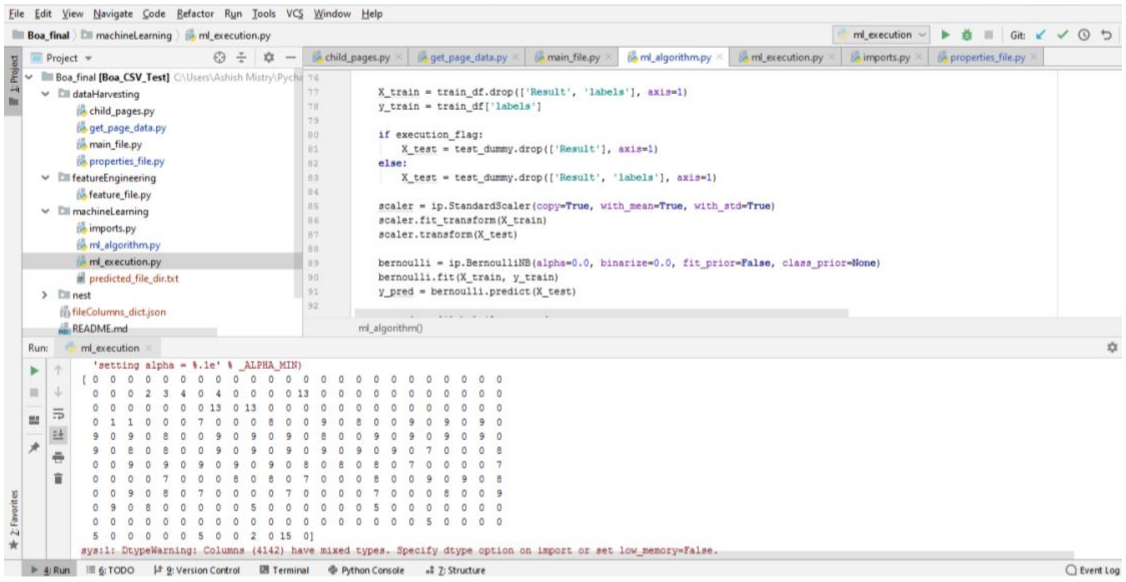


```java
    public void launchApplication() {
        driver.get(url);
    }

    @AfterSuite(alwaysRun = true)
    public void tearDown() throws IOException {
        driver.quit();
        try{
            String command = "python /c start python C:\\Users\\GORAS\\eclipse-workspace\\OrangeHRMFinalBranch\\s
            Process p = Runtime.getRuntime().exec("python C:\\Users\\GORAS\\eclipse-workspace\\OrangeHRMFinalBran
        }catch(NoSuchElementException e) {
            Reporter.log("Error executing python file");
        }
    }

    public void createDirectory(final ITestContext context) {

    public static void webHarvesting(final String onFailure){

        String DOM_Data;
        String[] url_parts;
        FileWriter fw = null;
        String passDir = "";

        currentUrl = driver.getCurrentUrl().toString().trim();
        url_parts = currentUrl.split("index.php");

        if ("true" == onFailure) {
            dir = new File(user_dir + "\\testData\\" + suiteName + "\\" + time + "_" + testName + "\\Fail");
            passDir = user_dir + "\\testData\\" + suiteName + "\\" + time + "_" + testName + "\\Pass";
        } else {
            dir = new File(user_dir + "\\testData\\" + suiteName + "\\" + time + "_" + testName + "\\Pass");
        }
        dir.mkdirs();

        DOM_Data = driver.getPageSource();
        File newTextFile = new File(
                dir + "\\" + url_parts[1].replace("/", "_").replace("?", "_").replace("=", "_") + ".txt");

        try {
            fw = new FileWriter(newTextFile);
```

```
No operations to display at this time.
```

Figure 5 - The TIK project being called within Selenium as a library (highlighted in yellow)

- o An algorithm that learns through automation run given a sprint.



```python
    X_train = train_df.drop(['Result', 'labels'], axis=1)
    y_train = train_df['labels']

    if execution_flag:
        X_test = test_dummy.drop(['Result'], axis=1)
    else:
        X_test = test_dummy.drop(['Result', 'labels'], axis=1)

    scaler = ip.StandardScaler(copy=True, with_mean=True, with_std=True)
    scaler.fit_transform(X_train)
    scaler.transform(X_test)

    bernoulli = ip.BernoulliNB(alpha=0.0, binarize=0.0, fit_prior=False, class_prior=None)
    bernoulli.fit(X_train, y_train)
    y_pred = bernoulli.predict(X_test)
```

```
  'setting alpha = %.1e' % _ALPHA_MIN)
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  2  3  4  0  4  0  0  0  0 13  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0 13  0 13  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  1  1  0  0  0  7  0  0  0  8  0  0  9  0  8  0  0  9  0  9  0  9  0
  9  0  9  0  8  0  0  9  0  9  0  9  0  8  0  0  9  0  9  0  9  0  9  0
  9  0  8  0  8  0  0  9  0  9  0  9  0  9  0  9  0  9  0  7  0  0  0  8
  0  0  9  0  9  0  9  0  9  0  9  0  8  0  8  0  8  0  7  0  0  0  0  7
  0  0  0  0  7  0  0  0  8  0  8  0  7  0  0  0  8  0  0  9  0  9  0  8
  0  0  9  0  8  0  7  0  0  0  0  7  0  0  0  0  7  0  0  0  8  0  0  9
  0  9  0  8  0  0  0  0  0  5  0  0  0  0  0  0  5  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0
  5  0  0  0  0  0  5  0  0  2  0 15  0]
sys:1: DtypeWarning: Columns (4142) have mixed types. Specify dtype option on import or set low_memory=False.
```

Figure 6 - TIK running through an iteration that converts Object to Numerical values consuming data to learn

Phase I – Root cause diagnosis

One of the most frequently overlooked challenges of predictive modeling is acquiring the right data to use when developing algorithms. By some estimates, data scientists spend about 80% of their time on this step. While predictive modeling is often considered to be primarily a mathematical problem, users must plan for the technical and organizational barriers that might prevent them from getting the data they need.

As a comparison, during regression testing, there is a lot of useful information at runtime that we collect and store for building a predictive model. Prior to destroying each object instance created at runtime, we document its properties as a .csv history file, so we refer to that information in future runs. With every new run of the framework, this historical data grows, allowing predictive models to become more accurate.

False positive detection is handled here once TIK points out if an episode was triggered due to test cases failure or any external cases such as environment issues or business requirements (with the help of a human eye).

Automated exploratory testing is another challenge we overcame. It is due to the efficiency of TIK reports that we identify and extract test cases that do not need to run regression test on because no modification was pointed out by our ML algorithm. Consequently, regression test is faster.

Automated defect diagnosis as TIK demonstrates savings of 60% average of the time needed to run regression tests in all test cases by determining the primary cause of a fault when a test crash. What is also reflected in money savings as our goal is to reduce human labour besides optimizing outputs.

TIK then loops through previous and current data to capture changes that happen which may cause a test case to fail based on test page steps pointing out modifications in environment, classes, attributes, properties, etc.

Historical data:

- DOM of instances of concrete objects at runtime;
- Properties of instances of abstract objects at runtime;
- DIFF between *historical* and *follow-up* runs;

Follow-up data:

- Test results;
- Recommended clusters

Challenges overcome

The biggest challenge we will face is the impact AI will have on staff. Many of the current automation roles will be transformed to hybrid positions, as AI replaces thinking processes and analytics. The management team will most likely be reengineered as staffing requirements will be reduced across the entire automation landscape.

During the project we fought some battles, a few of them we already expected due to casualty of facts we faced when developing automation frameworks through the project such as:

- Staff requiring a background in AI and conceptually understanding how AI works with its supported software languages.

Internally this is by far the biggest challenge. From the onset of the project, we did not have any data scientist on staff. Our team had to acquire several key individuals from programmers to a data scientist to enable the completion of this project. We did have the opportunity to train 4 of our staff members (development) in Python and statistics for machine learning.

Another challenge was throughout early months of the project where we did not have any consultation partners or research materials to refer back to. In despite of a lot of time spent in academic and market research, we could not find publications or evidence anywhere. We talked to many professionals, researchers and teachers from the field but as the core of Data Science starts by good quality data and, in this specific case, we built the dataset from the DOM and POM originally (html coding), which made the project even more challenging. It was truly a bleeding edge project, as most of the time designs were based on trial and error.

Lastly, convincing and maintaining the client's confidence to move the project forward was more of a challenge than we initially planned for. During weekly stand-ups (Scrum of Scrums) we kept detailed progress and relayed back to the PM team. Within 90 days we delivered a working model and it was able to demonstrate the direction of ***TIK*** to Laurentian Bank.

Future phases

Phase II – Predictive Models for Testing Results

Once we have a system in place that gradually creates input and output data, we will create a Machine Learning algorithm to predict test results before the framework is executed. This will bring a number of advantages, such as effective assessment of risk for test selection and prioritisation, as well as efficient signaling of potential catastrophic changes that should not be propagated to higher environments.

The execution of automated regression tests can be quite time consuming, even with many resources being used in parallel. By providing a forecast of test results prior to execution, TIK offers great help assessing critical, time-constrained decisions on code changes that could not be fully tested prior to deployment. The tests that are likely to fail and the defects that are likely to be reintroduced will be prioritized, so the immediate first results confirm the initial assumptions, allowing confidence in the code change to grow or shrink.

In bigger development projects, when multiple teams are working together, it is very challenging to spread the lessons learned from one individual to the entire team and the entire project. With its predictive model, TIK will help effectively flag the reoccurrence of problems based on historical data, minimizing root cause investigations and maximizing the efficiency of smoke tests.

Phase III - Exception handling strategies

Most test automation frameworks are written specifically for happy-ending, successful scenarios in which everything works. If, during runtime, something unexpected happens, the code stops running and the specific test are marked as Failed. When designing frameworks for our clients, we often go beyond that and write common alternative scenarios by manually handling exceptions that may be fired at runtime. This provides a great advantage in classifying the nature of the incident, so that a root cause is identified sooner.

With the use of Machine Learning, future versions of TIK might be able to dynamically create more alternative scenarios and different captured runtime iterations. This allows the framework to not only pinpoint a deviation from the happy-path, but also dynamically handle the exceptions so that the code continue to be executed, allowing for much greater test coverage even when blocker defects are on the way.

Phase VI - Automatic creation of abstract classes

Similar to how we plan on using TIK for pre-recognising and building structures of Page Object Model classes, we eventually have it smart enough to aid the creation of abstracted classes, which usually represent business concepts that are not obvious on the DOM.

Further research of ML techniques and their practical applications will help us develop this strategy, so that our framework automatically builds even higher-level additions to itself, increasing its adaptability even more.

APPENDIX

APPENDIX I. Bernoulli Naïve Bayes Machine Learning Method

Support:

Binary vectors of length K

x ϵ {0, 1}^{k}

Hypothesis:

Given the prior $\mathrm{P(y = k)}$

Assumes all features are conditional independent given the class

Likelihood $\mathrm{P(x_j | y = k)}$

Generative story:

$$\mathrm{Y \sim Bernoulli\ (\phi)}$$

$$\mathrm{X_k \sim Bernouilli\ (\theta_k, y)\ \ \forall k \in \{1, \ldots K\}}$$

The decision rule:

$$\hat{y} = arg\ max_k \underbrace{\boxed{P\left(x_j \middle| y = k\right)}}_{\text{Posterior distribution of classes}}$$

$$= arg\ max_k P(x|y = k)P(y = k)$$

$$= arg\ max_k \underbrace{\boxed{P(y = k)}}_{\text{Prior distribution of classes}} \underbrace{\boxed{\prod_{j=1}^{p^k} P\left(x_j \middle| y = k\right)}}_{\text{Decomposition using conditional independence}}$$

APPENDIX II. The Algorithm

```
from sklearn.naive_bayes import BernoulliNB

bernoulli = BernoulliNB(alpha=0.0, binarize=0.0, fit_prior=False, class_prior=None)

bernoulli.fit(X_train, y_train)

y_pred = bernoulli.predict(X_test)
```

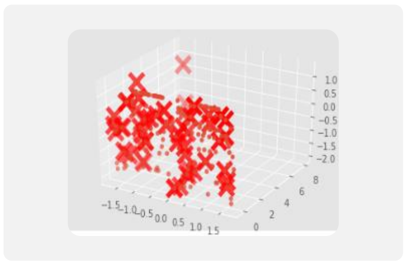


Figure 1 - Element Classification

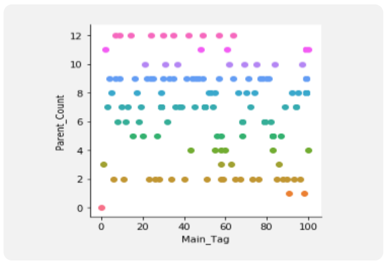


Figure 2 - Element Cluster