Object Oriented Software Engineering Project

Group 6

5th December 2022

Team Members:

1. Suraj Raj - 2836416
2. Pradeep Nandagiri - 2835198
3. John wesely Karavadi - 2830415
4. Pavan Sai Pamujula - 2836714

An Experimental Study with Imbalanced Classification Approaches for Credit Card Fraud Detection

INTRODUCTION

Now a day the usage of credit cards has dramatically increased. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising. In this paper, we model the sequence of operations in credit card transaction processing using a Decision tree and Deep Neural Network show how it can be used for the detection of frauds. An both algorithms is initially trained with the normal behaviour of a cardholder. If an incoming credit card transaction is not accepted by the trained with sufficiently high probability, it is considered to be fraudulent. At the same time, we try to ensure that genuine transactions. We present detailed experimental results to show the effectiveness of our approach and compare it with other techniques available in the literature.

Motivation

- The prediction Model will describe you whether to invest in the proposal or not. Here, we choose to minimize the risk for investing, i.e. we aim to minimize investing in proposals for which the loan will not be paid back.

Issues

Credit card fraud is a criminal offense. It causes severe damage to financial institutions and individuals. Therefore, the detection and prevention fraudulent activities are critically important to financial institutions. Fraud detection and prevention are costly, time-consuming and labor-intensive tasks. A number of significant research works have been dedicated to developing innovative solutions to detect different types of fraud. However, these solutions have been proved ineffective. According to Cifa, 33,305 cases of credit card identity fraud were reported between January and June in 2018.

Scope of The Project

- In this proposed project we designed a protocol or a model to detect the fraud activity in credit card transactions.
- This system is capable of providing most of the essential features required to detect fraudulent and legitimate transactions.
- As technology changes, it becomes difficult to track the behaviour and pattern of fraudulent transactions.
- With the rise of machine learning, artificial intelligence and other relevant fields of information technology, it becomes feasible to automate this process and to save some of the intensive amount of labour that is put into detecting credit card fraud.

Abstract

In our project, mainly focussed on credit card fraud detection for in real world. Initially I will collect the credit card datasets for trained dataset. Then will provide the user credit card queries for testing data set. After classification process of random forest algorithm using to the already analysing data set and user provide current dataset. Finally optimizing the accuracy of the result data. Then will apply the processing of some of the attributes provided can find affected fraud detection in viewing the graphical model visualization. The performance of the techniques is evaluated based on accuracy, sensitivity, and specificity, precision. The results indicate about the optimal accuracy for Decision tree are 98.6% respectively.

Existing System

In existing System, a research about a case study involving credit card fraud detection, where data normalization is applied before Cluster Analysis and with results obtained from the use of Cluster Analysis and Artificial Neural Networks on fraud detection has shown that by clustering attributes neuronal inputs can be minimized. And promising results can be obtained by using normalized data and data should be MLP trained. This research was based on unsupervised learning. Significance of this paper was to find new methods for fraud detection and to increase the accuracy of results. The data set for this paper is based on real life transactional data by a large European company and personal details in data is kept confidential. Accuracy of an algorithm is around 50%. Significance of this paper was to find an algorithm and to reduce the cost measure. The result obtained was by 23% and the algorithm they find was Bayes minimum risk.

Disadvantage:

- In this paper a new collative comparison measure that reasonably represents the gains and losses due to fraud detection is proposed.
- A cost sensitive method which is based on Bayes minimum risk is presented using the proposed cost measure.

Proposed System

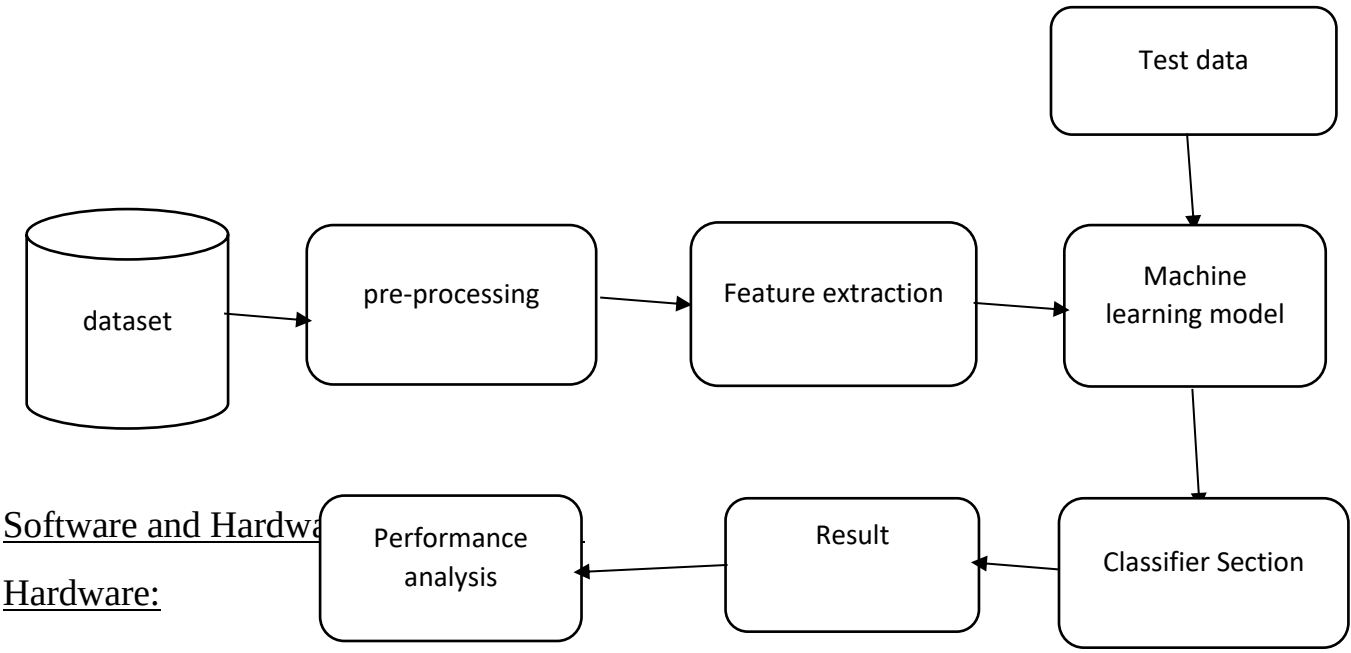
In proposed System, we are applying random forest algorithm for classify the credit card dataset. Decision tree is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Decision tree has advantage over decision tree as it corrects the habit of over fitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others.

The Decision tree algorithm has been found to provide a good estimate of the generalization error and to be resistant to overfitting.

Advantage:

- Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Decision tree.
- The 'amount' feature is the transaction amount. Feature 'class' is the target class for the binary classification and it takes value 1 for positive case (fraud) and 0 for negative case (non fraud).

System Architecture



Software and Hardwa

Hardware:

- OS – Windows 7, 8 and 10 (32 and 64 bit)
- RAM – 4GB

Software:

- Python

- Anaconda

PROBLEM STATEMENT

Billions of dollars of loss are caused every year by the fraudulent credit card transactions. Fraud is old as humanity itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there is definitely a need to solve the problem of credit card fraud detection. Moreover, the development of new technologies provides additional ways in which criminals may commit fraud. The use of credit cards is prevalent in modern day society and credit card fraud has been kept on growing in recent years. Hugh Financial losses has been fraudulent affects not only merchants and banks, but also individual person who are using the credits. Fraud may also affect the reputation and image of a merchant causing non-financial losses that, though difficult to quantify in the short term, may become visible in the long period. For example, if a cardholder is victim of fraud with a certain company, he may no longer trust their business and choose a competitor.

METHODOLOGY

There are various fraudulent activities detection techniques has implemented in credit card transactions have been kept in researcher minds to methods to develop models based on artificial intelligence , data mining, fuzzy logic and machine learning. Credit card fraud detection is an extremely difficult, but also popular problem to solve. In our proposed system we built the credit card fraud detection using Machine learning. With the advancement of machine

learning techniques. Machine learning has been recognized as a successful measure for fraud detection. A great deal of data is transferred during online transaction processes, resulting in a binary result: genuine or fraudulent. Online businesses are able to identify fraudulent transactions accurately because they receive chargebacks on them. Within the sample fraudulent datasets, features are constructed. These are data points such as the age and value of the customer account, as well as the origin of the credit card. There can be hundreds of features and each contributes, to varying extents, towards the fraud probability. Note, the degree in which each feature contributes to the fraud score is not determined by a fraud analyst, but is generated by the artificial intelligence of the machine which is driven by the training set. So, in regards to the card fraud, if the use of cards to commit fraud is proven to be high, the fraud weighting of a transaction that uses a credit card will be equally so. However, if this were to diminish, the contribution level would parallel. Simply put, these models self-learn without explicit programming such as with manual review. Credit card fraud detection using Machine learning is done by deploying the classification and regression algorithms. We use supervised learning algorithm such as Decision tree algorithm to classify the fraud card transaction in online or by offline. Random forest has better efficiency and accuracy than the other machine learning algorithms. Random forest aims to reduce the previously mentioned correlation issue by choosing only a subsample of the feature space at each split. Essentially, it aims to make the trees de-correlated and prune the trees by setting a stopping criteria for node splits, which I will cover in more detail later.

LITERATURE SURVEY

The Use of Predictive Analytics Technology to Detect Credit Card Fraud in Canada

(Kosemani Temitayo Hafiz, Dr. Shaun Aghili, Dr. Pavol Zavarsky)

This research paper focuses on the creation of a scorecard from relevant evaluation criteria, features, and capabilities of predictive analytics vendor solutions currently being used to detect credit card fraud. The scorecard provides a side-by-side comparison of five credit card predictive analytics vendor solutions adopted in Canada. From the ensuing research findings, a list of credit card fraud PAT vendor solution challenges, risks, and limitations was outlined.

1) **BLAST-SSAHA Hybridization for Credit Card Fraud Detection (Amlan Kundu, Suvasini Panigrahi, Shamik Sural, Senior Member, IEEE, and Arun K. Majumdar)**

In this paper, we propose to use two-stage sequence alignment in which a profile Analyser (PA) first determines the similarity of an incoming sequence of transactions on a given credit card with the genuine cardholder's past spending sequences. The unusual transactions traced by the profile analyser are next passed on to a deviation analyser (DA) for possible alignment with past fraudulent behaviour. The final decision about the nature of a transaction is taken on the basis of the observations by these two analysers. In order to achieve online response time for both PA and DA, we suggest a new approach for combining two sequence alignment algorithms BLAST and SSAHA.

2) **Research on Credit Card Fraud Detection Model Based on Distance Sum (Wen-Fang YU, Na Wang)**

Along with increasing credit cards and growing trade volume in China, credit card fraud rises sharply. How to enhance the detection and prevention of credit card fraud becomes the focus of risk control of banks. This paper proposes a

credit card fraud detection model using outlier detection based on distance sum according to the infrequency and unconventionality of fraud in credit card transaction data, applying outlier mining into credit card fraud detection. Experiments show that this model is feasible and accurate in detecting credit card fraud.

3) **Fraudulent Detection in Credit Card System Using SVM & Decision Tree**

   (Vijayshree B. Nipane, Poonam S. Kalinge, Dipali Vidhate, Kunal War, Bhagyashree P. Deshpande)

With growing advancement in the electronic commerce field, fraud is spreading all over the world, causing major financial losses. In current scenario, Major cause of financial losses is credit card fraud; it not only affects trades person but also individual clients. Decision tree, Genetic algorithm, Meta learning strategy, neural network, HMM are the presented methods used to detect credit card frauds. In contemplate system for fraudulent detection, artificial intelligence concept of Support Vector Machine (SVM) & decision tree is being used to solve the problem. Thus by implementation of this hybrid approach, financial losses can be reduced to greater extend.

4) **Supervised Machine (SVM) Learning for Credit Card Fraud Detection**

   (Sitaram patel, Sunita Gond)

In this thesis we are proposing the SVM (Support Vector Machine) based method with multiple kernel involvement which also includes several fields

of user profile instead of only spending profile. The simulation result shows improvement in TP (true positive), TN (true negative) rate, & also decreases the FP (false positive) & FN (false negative) rate.

5) **Detecting Credit Card Fraud by Decision Trees and Support Vector Machines**
   (Y. Sahin and E. Duman)

In this study, classification models based on decision trees and support vector machines (SVM) are developed and applied on credit card fraud detection problem. This study is one of the firsts to compare the performance of SVM and decision tree methods in credit card fraud detection with a real data set.

6) **Machine Learning based Approach to Financial Fraud Detection Process in Mobile Payment System**
   (Dahee Choi and Kyungho Lee)

Mobile payment fraud is the unauthorized use of mobile transaction through identity theft or credit card stealing to fraudulently obtain money. Mobile payment fraud is the fast growing issue through the emergence of smart phone and online transition services. In the real world, highly accurate process in mobile payment fraud detection is needed since financial fraud causes financial loss. Therefore, our approach proposed the overall process of detecting mobile payment fraud based on machine learning, supervised and unsupervised method to detect fraud and process large amounts of financial data. Moreover, our approach performed sampling process and feature selection process for fast processing with large volumes of transaction data

and to achieve high accuracy in mobile payment detection. F-measure and ROC curve are used to validate our proposed model.

7) **Credit Card Fraud Detection Using Decision Tree Induction Algorithm**

(Snehal Patil, Harshada Somavanshi, Jyoti Gaikwad, Amruta Deshmane, Rinku Badgujar)

A new cost-sensitive decision tree approach which reduces the sum of misclassification costs while selecting the splitting attribute at each nonterminal node is advanced and the act of this approach is compared with the well-known traditional classification models on a real world credit card data set. This research is totally concerned with credit card application fraud detection by performing the process of asking security queries to the persons intricate with the transactions and as well as by eliminating real time data faults.

8) **Data Mining Techniques for Credit Card Fraud Detection: Empirical Study**

(Marwan Fahmi, Abeer Hamdy, Khaled Nagati)

Fraud detection is a crucial problem that has been facing the e-commerce industry for decades. Financial institutions throughout the world lose billions due to credit card fraud, which necessitates the use of credit card fraud prevention. Several models have been proposed in the literature, however, the accuracy of the model is crucial. In this paper four fraud detection models based on data mining techniques (Support vector machine, K-nearest neighbours, Decision Trees, Naïve Bayes) were developed and their performances were compared when applied on a real life anonymised data set

of transactions ("UCSD-FICO Data Mining Contest 2009"). Four relevant metrics were used in evaluating the performance of the classifiers which are True positive rate (TPR), False Positive Rate (FPR), Balanced Classification Rate (BCR) and Matthews Correlation Coefficient (MCC).

9) Card Fraud Detection Using Learning Machines (Gheorghe Asachi" din Iaşi)

Searching for Card Fraud via the Internet will return approximately 180 million results. The total level of fraud reached 1.26 billion euro in 2010 in Europe according with BCE. The ingenuity of thieves reached highly sophisticated forms. To model mathematically this behaviour requires a classification method derived from supervised learning algorithm which must be able to separate the class of fraudulent with a high degree of accuracy. Following his definition, the technique of Support Vector Machines is characterized by two strong hypotheses: margin optimization and kernel representation. So, I chose the techniques of SVM with non-linear kernels. We propose the Gaussian kernel function for measuring the similarities between features into new linear space as the best approach to detect the fraud patterns.

PURPOSE OF THE PROJECT

We propose a Machine learning model to detect the fraudulent credit card activities in online financial transactions. Analysing fraudulent transactions manually is unfeasible due to huge amounts of data and its

complexity. However, given sufficiently informative features, one could expect it is possible to do using Machine Learning. This hypothesis will be explored in the project.

To classify fraudulent and legitimate credit card transaction by supervised learning Algorithm such as Random forest.

To helps us to get awareness about the fraudulent and without loss of any financially.

MODULES

1. **DATA COLLECTION**
2. **DATA PRE-PROCESSING**
3. **FEATURE EXTRATION**
4. **EVALUATION MODEL**
5. **FLASK**

DATA COLLECTION

Data used in this paper is a set of product reviews collected from credit card transactions records. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called *labelled data*.

DATA PRE-PROCESSING

Organize your selected data by formatting, cleaning and sampling from it.

Three common data pre-processing steps are:

- Formatting: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.
- Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.
- Sampling: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

FEATURE EXTRATION

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random forest. These algorithms are very popular in text classification tasks.

EVALUATION MODEL

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data

used for training is not acceptable in data science because it can easily generate overoptimistic and over fitted models. There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid over fitting, both methods use a test set (not seen by the model) to evaluate model performance. Performance of each classification model is estimated base on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs. **Accuracy** is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

FLASK

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Features:

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Complete documentation
- Google App Engine compatibility
- Extensions available to extend functionality

Example:

The following code shows a simple web application that displays "Hello World!" when visited:

```
from flask import Flaskapp = Flask(__name__)
@app.route("/")def hello() -> str:
```

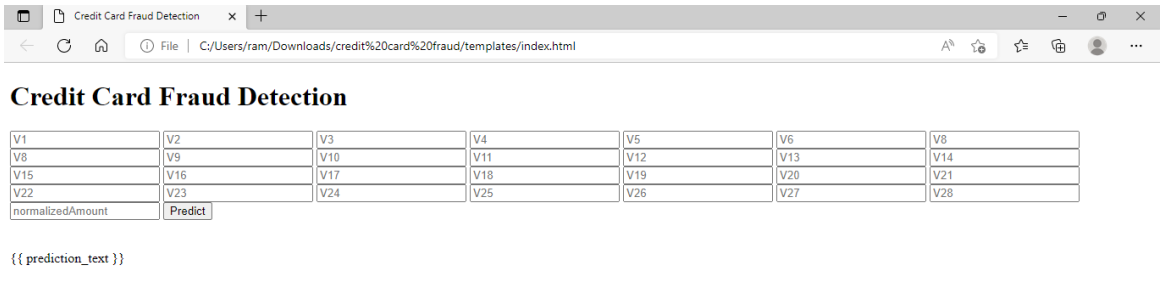
```
    return "Hello World"

if __name__ == "__main__":
    app.run(debug=False)
```

Templates are files that contain static data as well as placeholders for dynamic data. A template is rendered with specific data to produce a final document. Flask uses the Jinja template library to render templates.
In your application, you will use templates to render HTML which will display in the user's browser. In Flask, Jinja is configured to autoescape any data that is rendered in HTML templates. This means that it's safe to render user input; any characters they've entered that could mess with the HTML, such as `<` and `>` will be escaped with safe values that look the same in the browser but don't cause unwanted effects.
Jinja looks and behaves mostly like Python. Special delimiters are used to distinguish Jinja syntax from the static data in the template. Anything between `{{` and `}}` is an expression that will be output to the final document. `{%` and `%}` denotes a control flow statement like `if` and `for`. Unlike Python, blocks are denoted by start and end tags rather than indentation since static text within a block could change indentation.
Each page in the application will have the same basic layout around a different body. Instead of writing the entire HTML structure in each template, each template will extend a base template and override specific sections.

```
<!doctype html>
<title>{% block title %}{% endblock %} - Flaskr</title>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
<nav>
  <h1>Flaskr</h1>
  <ul>
    {% if g.user %}
      <li><span>{{ g.user['username'] }}</span>
      <li><a href="{{ url_for('auth.logout') }}">Log Out</a>
    {% else %}
      <li><a href="{{ url_for('auth.register') }}">Register</a>
      <li><a href="{{ url_for('auth.login') }}">Log In</a>
    {% endif %}
  </ul>
</nav>
<section class="content">
  <header>
    {% block header %}{% endblock %}
  </header>
  {% for message in get_flashed_messages() %}
    <div class="flash">{{ message }}</div>
  {% endfor %}
  {% block content %}{% endblock %}
</section>
```

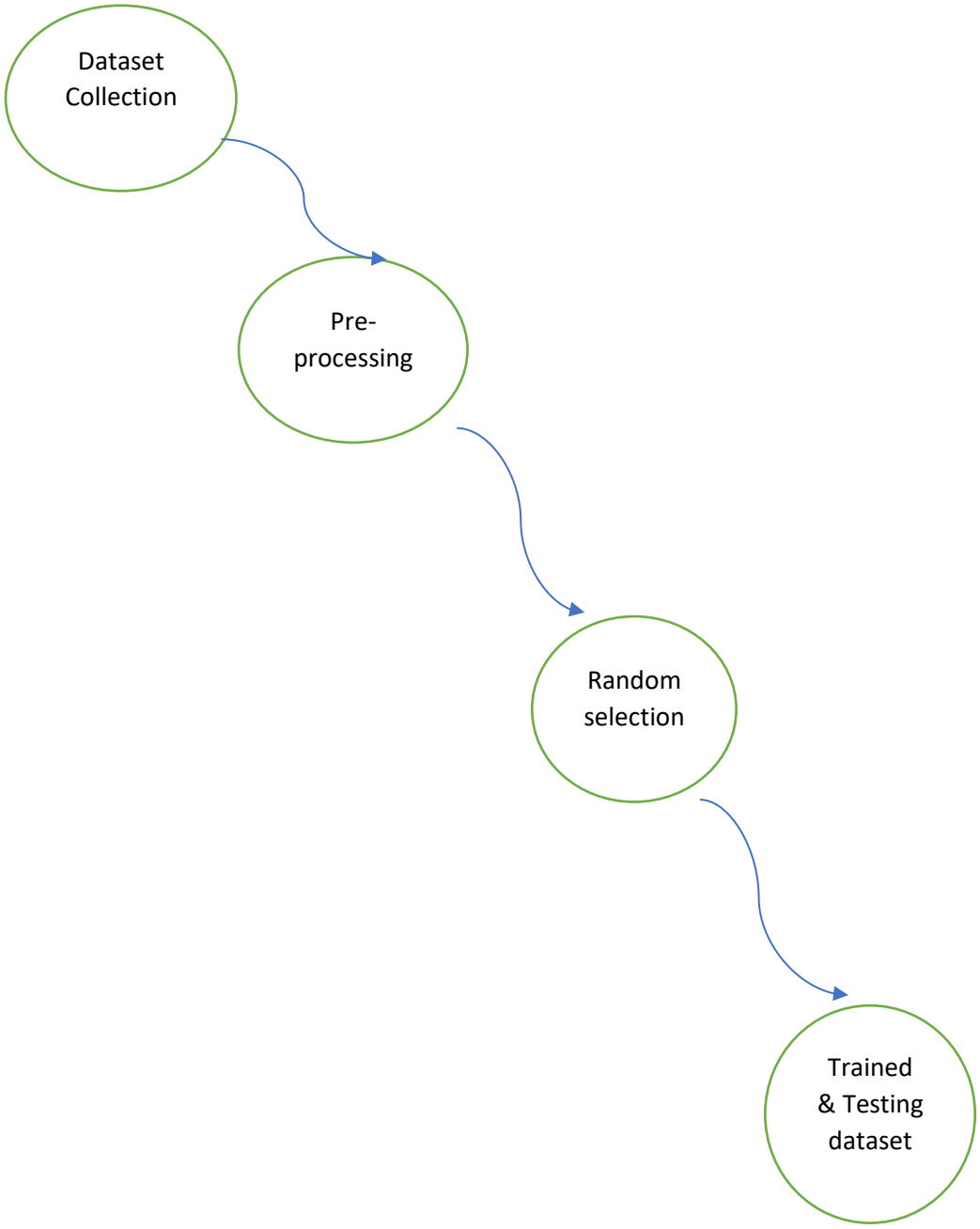
g is automatically available in templates. Based on if g.user is set (from load_logged_in_user), either the username and a log out link are displayed, or links to register and log in are displayed. **url_for()** is also automatically available, and is used to generate URLs to views instead of writing them out manually.

OUTPUT FOR FRONTEND:

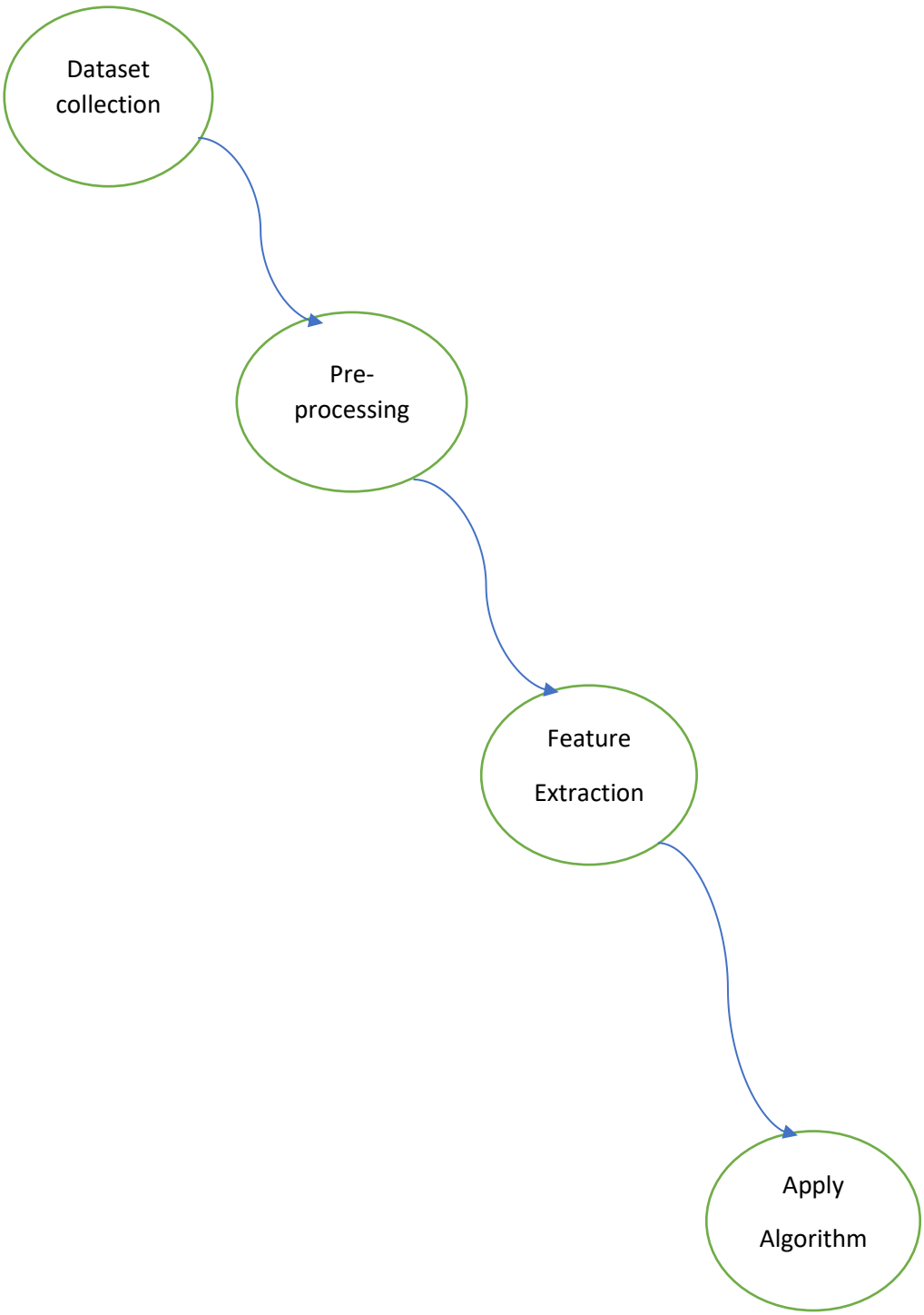


DATA FLOW DIAGRAM

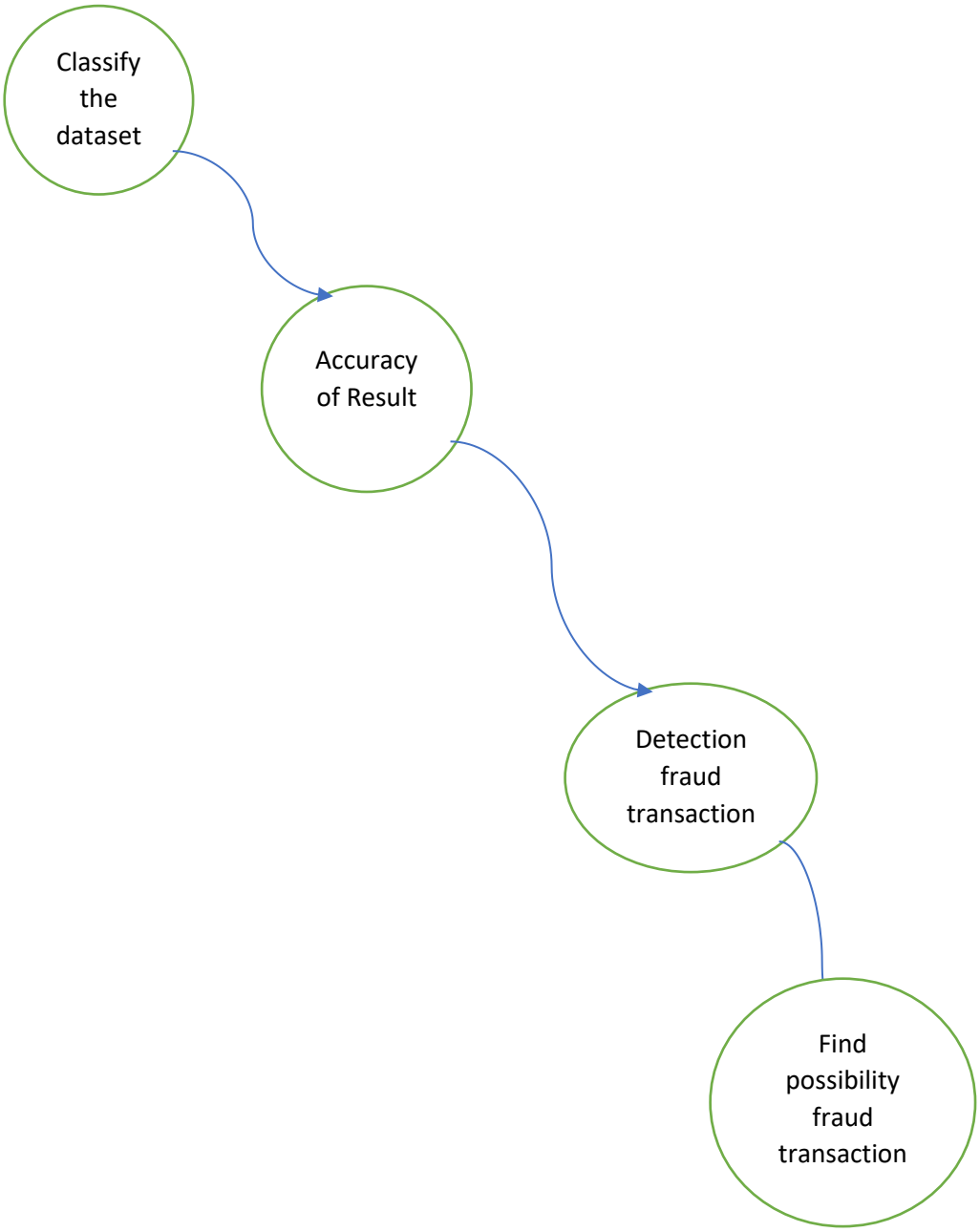
LEVEL 0



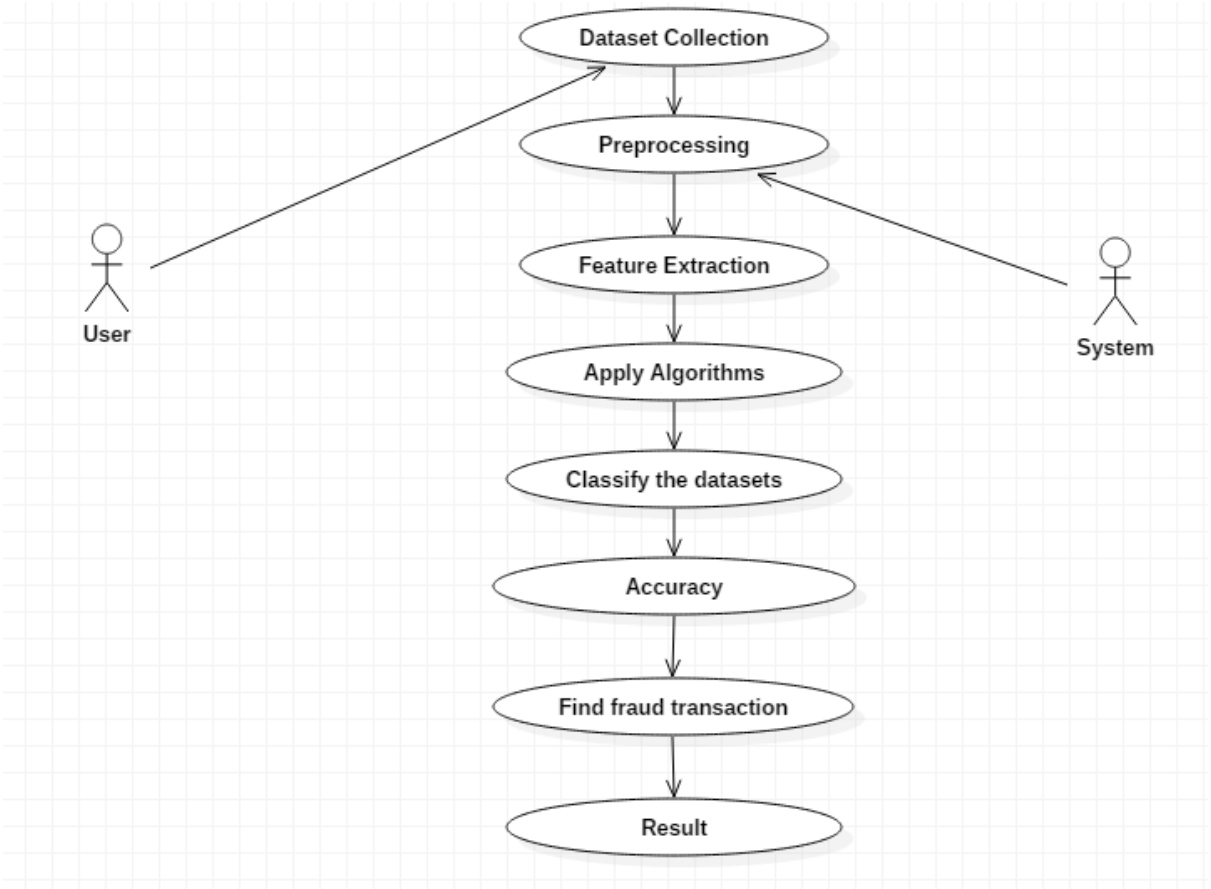
LEVEL 1


Dataset collection
Pre-processing
Feature Extraction
Apply Algorithm

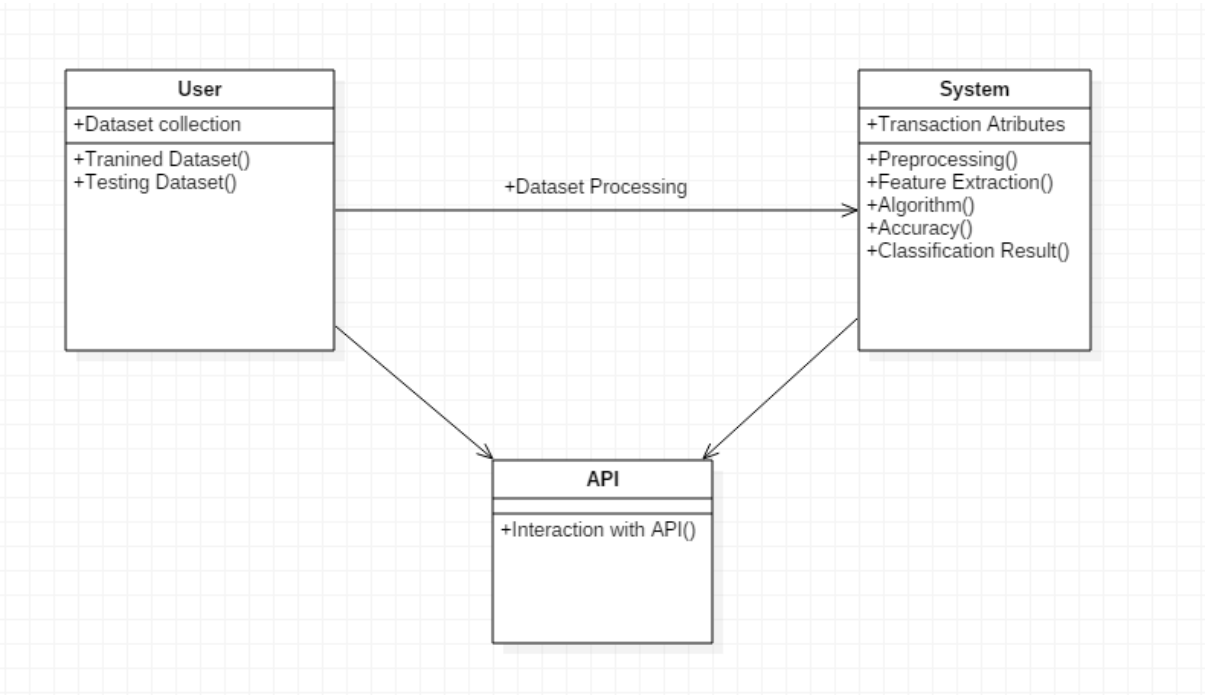
LEVEL 2


Classify the dataset
Accuracy of Result
Detection fraud transaction
Find possibility fraud transaction

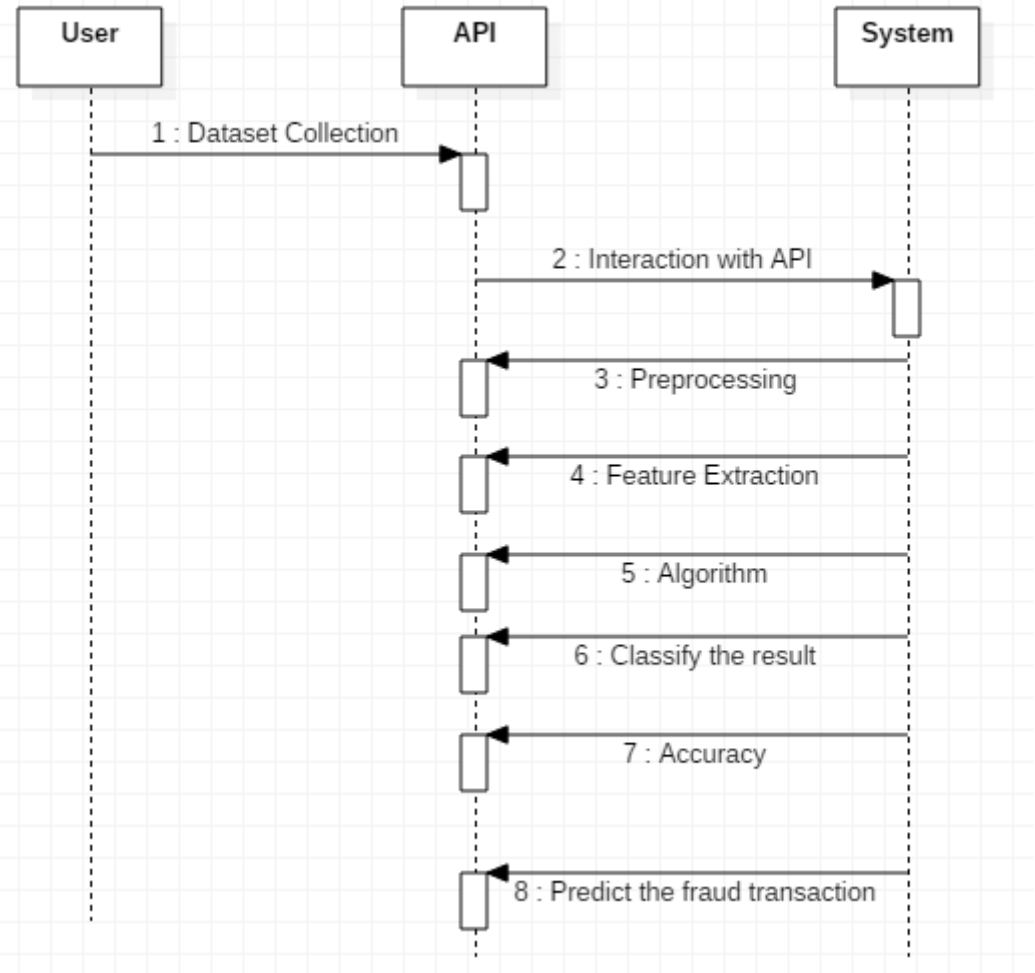
UML DIAGRAMS



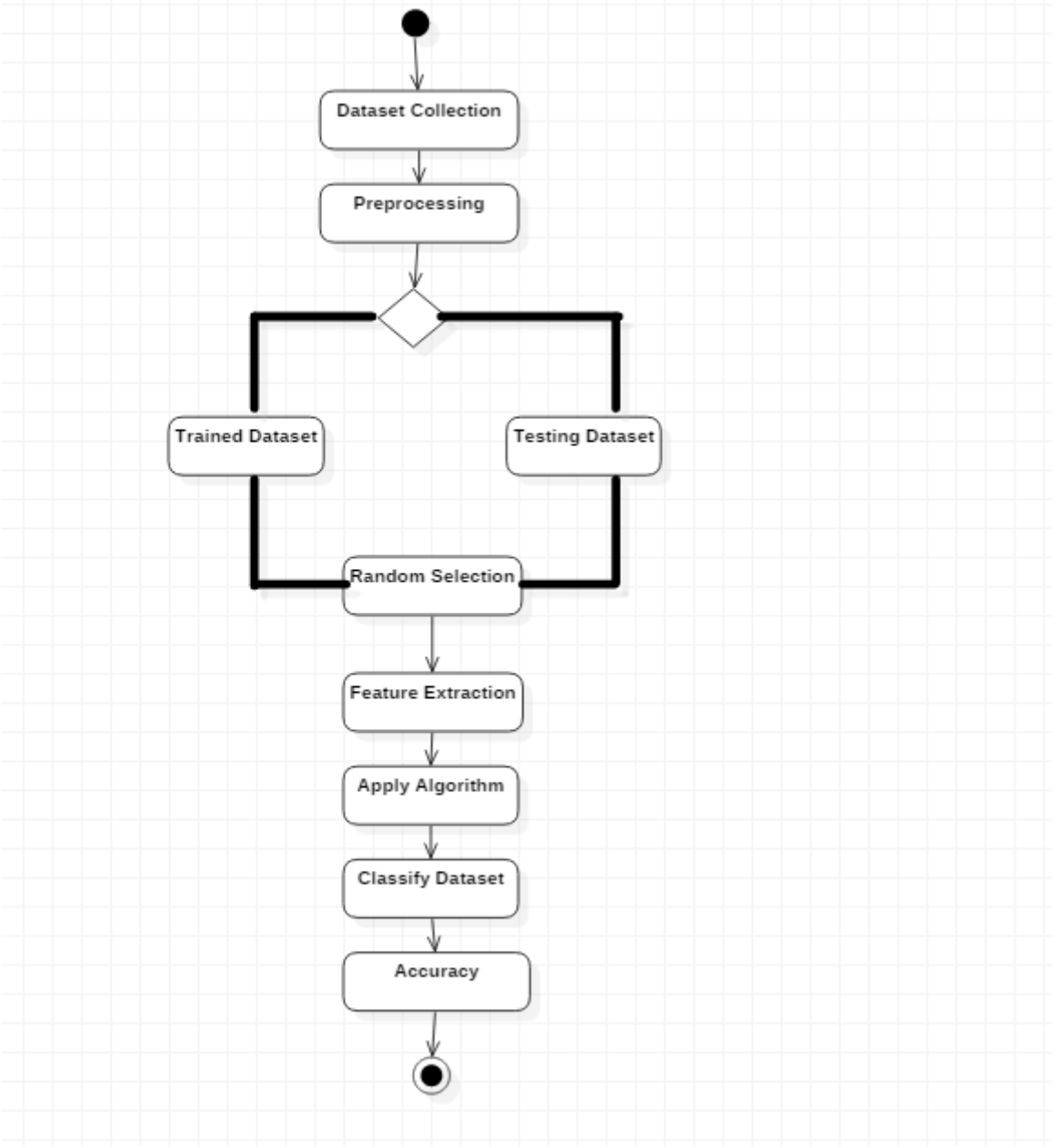
CLASS DIAGRAM



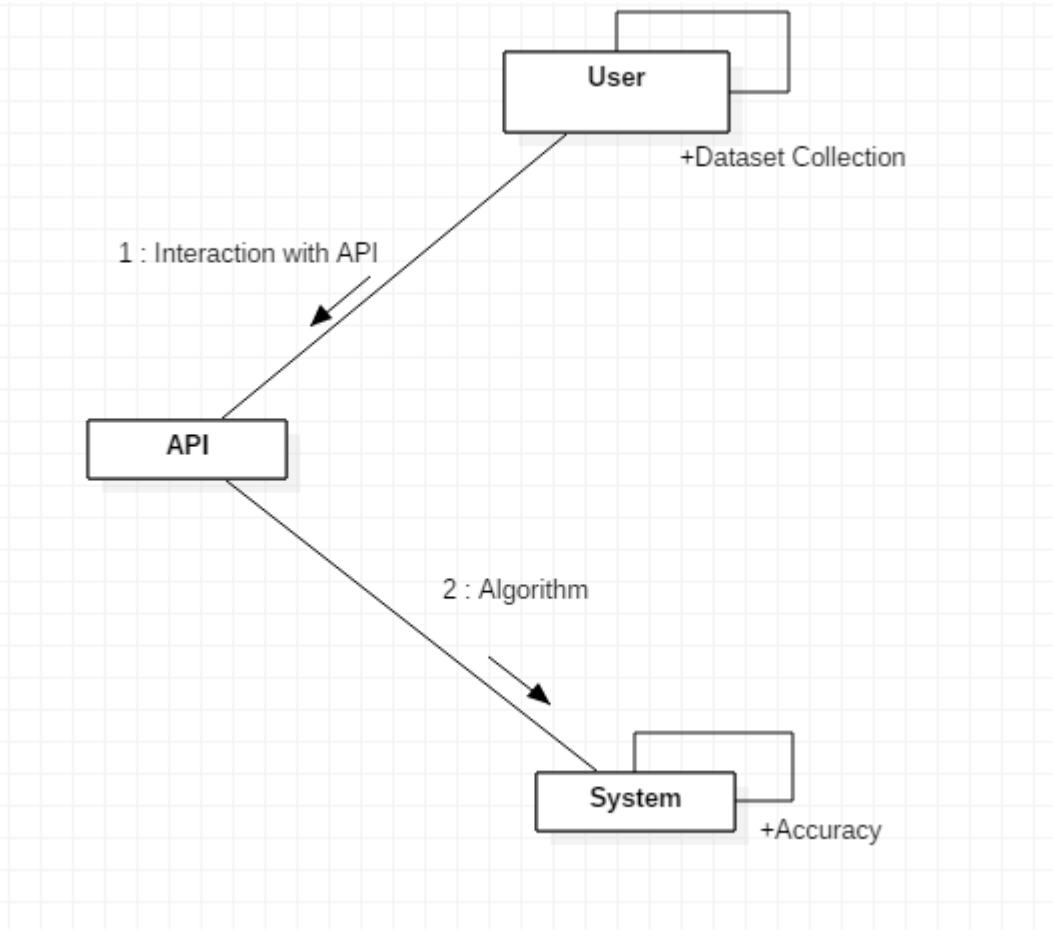
SEQUENCE DIAGRAM



ACTIVITY DIAGRAM



COLLABRATION DIAGRAM



ALGORITHM

Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

Advantages and Disadvantages

Following are the advantages of decision trees: - Easy to use and understand. - Can handle both categorical and numerical data. - Resistant to outliers, hence require little data preprocessing. - New features can be easily added. - Can be used to build larger classifiers by using ensemble methods.

Following are the disadvantages of decision trees: - Prone to overfitting. - Require some kind of measurement as to how well they are doing. - Need to be careful with parameter tuning. - Can create biased learned trees if some classes dominate.

Domain Specification

MACHINE LEARNING

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

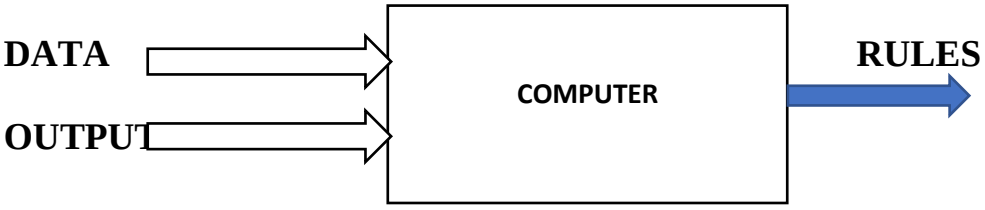
Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.



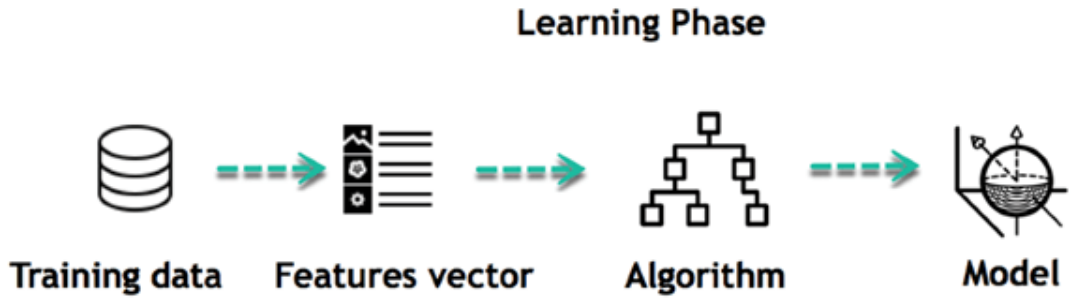
Machine Learning

How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector**.** You can think of a feature vector as a subset of data that is used to tackle a problem.

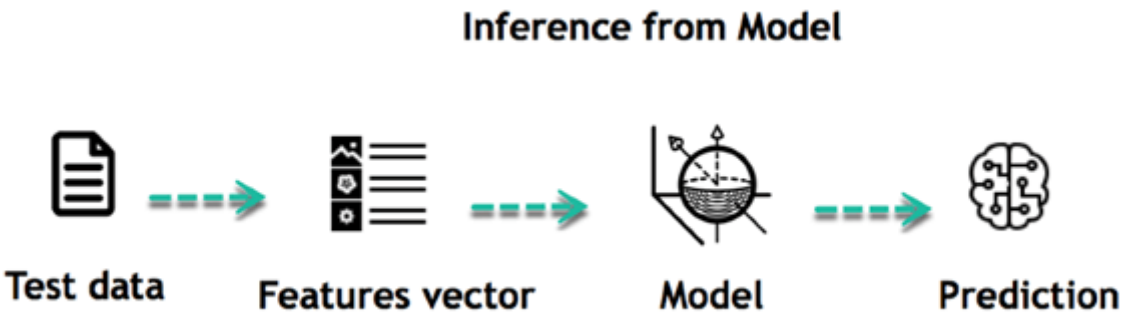
The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

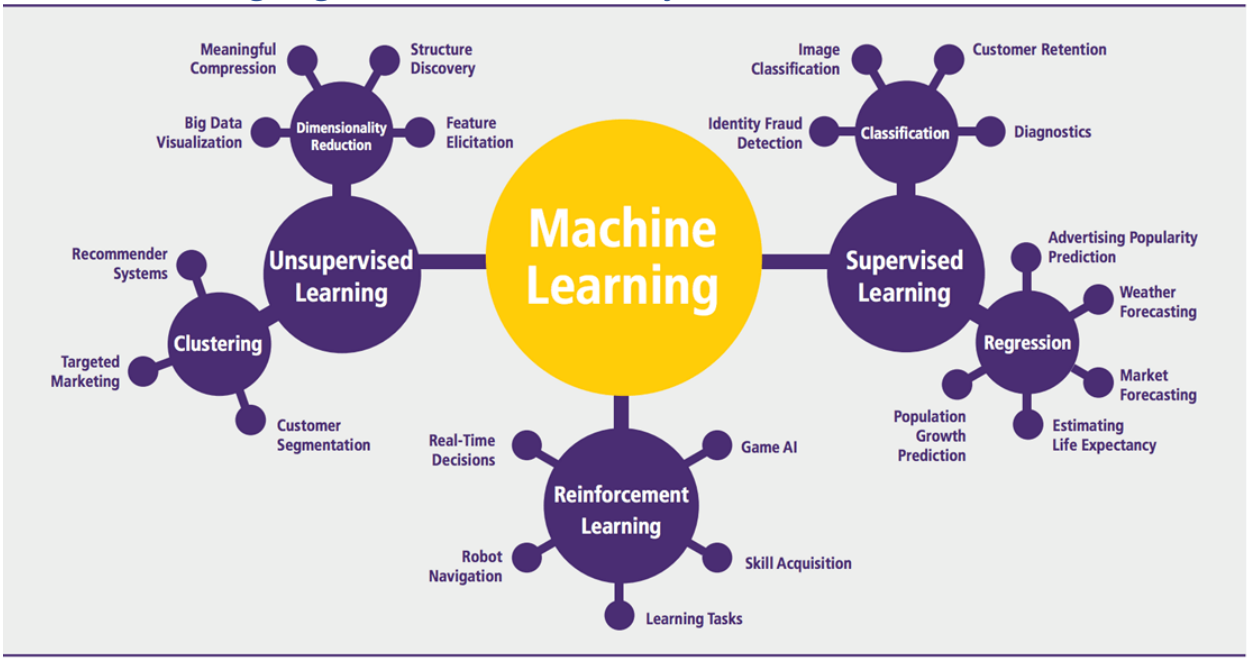


The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

Machine learning Algorithms and where they are used?



Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

Supervised learning
An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

Algorithm Name	Description	Type
Linear regression	Finds a way to correlate each feature to the output to help predict future values.	Regressior
Logistic regression	Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors)	Classificat
Decision tree	Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made	Regressior Classificat
Naive Bayes	The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event.	Regressior Classificat
Support vector machine	Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver.	Regressior very co Classificat

Random forest	The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction.	Regressior Classificat
AdaBoost	Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome	Regressior Classificat
Gradient-boosting trees	Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it.	Regressior Classificat

- Classification task
- Regression task

Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index.

The system will be trained to estimate the price of the stocks with the lowest possible error.

Unsupervised learning
In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

Algorithm	Description	Type
K-means clustering	Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans)	Clustering
Gaussian mixture model	A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters	Clustering
Hierarchical clustering	Splits clusters along a hierarchical tree to form a classification system. Can be used for Cluster loyalty-card customer	Clustering
Recommender system	Help to define the relevant data for making a recommendation.	Clustering
PCA/T-SNE	Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances.	Dimension Reduction

Application of Machine learning

Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual

Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

- Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

Government organization

- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

Example of Machine Learning Google Car

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in term of AI. In deep learning, the learning phase is done through a neural network.

Reinforcement Learning

Reinforcement learning is a subfield of machine learning in which systems are trained by receiving virtual "rewards" or "punishments," essentially learning by trial and error. Google's DeepMind has used reinforcement learning to beat a human champion in the Go games. Reinforcement learning is also used in video games to improve the gaming experience by providing smarter bot.

One of the most famous algorithms are:

- Q-learning
- Deep Q network
- State-Action-Reward-State-Action (SARSA)
- Deep Deterministic Policy Gradient (DDPG)

Applications/ Examples of deep learning applications

AI in Finance: The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

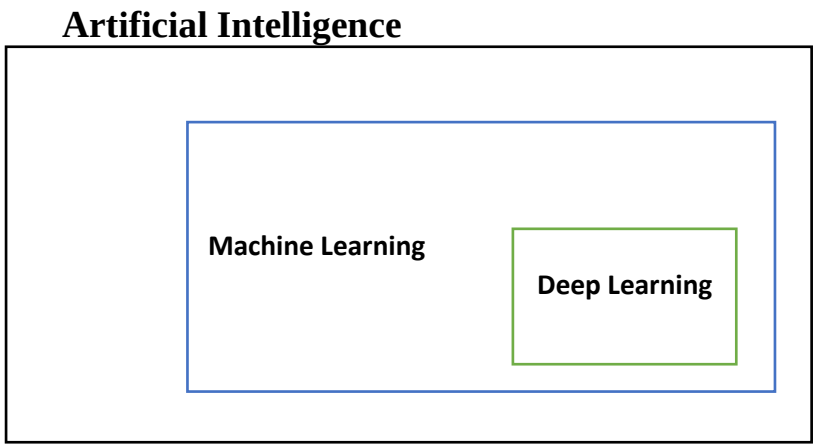
AI in HR: Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose **HireVue**, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing; they managed to decrease by 35% the time to fill. In return, the hired higher quality staffs.

AI in Marketing: AI is a valuable tool for customer service management and personalization challenges. Improved speech recognition in call-center management and call routing as a result of the application of AI techniques allows a more seamless experience for customers.

For example, deep-learning analysis of audio allows systems to assess a customer's emotional tone. If the customer is responding poorly to the AI chatbot, the system can be rerouted the conversation to real, human operators that take over the issue.

Apart from the three examples above, AI is widely used in other sectors/industries.



Difference between Machine Learning and Deep Learning

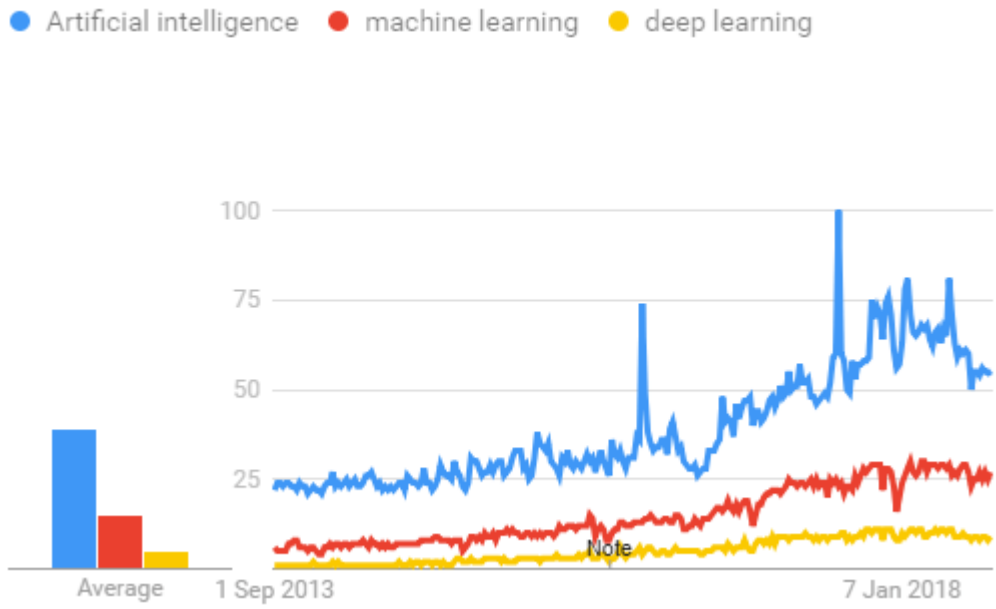
	Machine Learning	Deep Learning
Data Dependencies	Excellent performances on a small/medium dataset	Excellent performance on dataset
Hardware dependencies	Work on a low-end machine.	Requires powerful m preferably with GPU: DL per significant amount of multiplication
Feature engineering	Need to understand the features that represent the data	No need to understand th feature that represents the dat
Execution time	From few minutes to hours	Up to weeks. Neural Networ to compute a significant nu weights
Interpretability	Some algorithms are easy to interpret (logistic, decision tree), some are almost impossible (SVM, XGBoost)	Difficult to impossible

When to use ML or DL?

In the table below, we summarize the difference between machine learning and deep learning.

	Machine learning	Deep learning
Training dataset	Small	Large
Choose features	Yes	No
Number of algorithms	Many	Few
Training time	Short	Long

With machine learning, you need fewer data to train the algorithm than deep learning. Deep learning requires an extensive and diverse set of data to identify the underlying structure. Besides, machine learning provides a faster-trained model. Most advanced deep learning architecture can take days to a week to train. The advantage of deep learning over machine learning is it is highly accurate. You do not need to understand what features are the best representation of the data; the neural network learned how to select critical features. In machine learning, you need to choose for yourself what features to include in the model.



TensorFlow

the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

- Researchers
- Data scientists
- Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

TensorFlow Architecture

Tensor flow architecture works in three parts:

- Pre processing the data
- Build the model
- Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at

one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensor flow run?

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on

- Desktop running Windows, macOS or Linux
- Cloud as a web service
- Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what TensorFlow is doing.

List of Prominent Algorithms supported by TensorFlow

- Linear regression: tf. estimator .Linear Regressor
- Classification :tf. Estimator .Linear Classifier
- Deep learning classification: tf. estimator. DNN Classifier
- Booster tree regression: tf.estimator.BoostedTreesRegressor
- Boosted tree classification: tf.estimator.BoostedTreesClassifier

PYTHON OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- IT supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

REQUIREMENTS ANAYLSIS

SOFTWARE REQUIREMENTS

- Python
- Anaconda Navigator
- Python built-in modules

  - Numpy
  - Pandas
  - Matplotlib
  - Sklearn
  - Seaborm

ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

- Jupyter Lab
- Jupyter Notebook
- QT Console
- Spyder
- VS Code
- Glue viz
- Orange 3 App
- Rodeo
- RStudio

Advanced conda users can also build your own Navigator applications

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What’s new in 1.9?

- Add support for **Offline Mode** for all environment related actions.
- Add support for custom configuration of main windows links.
- Numerous bug fixes and performance enhancements.

PYTHON

Python

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to soitware development and scientific applications.

It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

Features of Python

A simple language which is easier to learn, Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

- **Free and open source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes

to the Python's source code. Python has a large community constantly improving it in each iteration.

- **Portability**

You can move Python programs from one platform to another, and run it without any changes.

It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

- **Extensible and Embeddable**

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

- **Large standard libraries to solve common tasks**

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server You can use MySQLdb library using import MySQL db Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

- **Object-oriented**

- Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively. With OOP, you are able to divide these complex problems into smaller sets by creating object

Python

History and Versions:

Python is predominantly a dynamic typed programming language which was initiated by Guido van Rossum in the year 1989. The major design philosophy that was given more importance was the readability of the code and expressing an idea in fewer lines of code rather than the verbose way of expressing things as in C++ and Java [K-8][K-9]. The other design philosophy that was worth mentioning was that, there should be always a single way and a single obvious way to express a given task which is contradictory to other languages such as C++, Perl etc. [K-10]. Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The initial versions of Python were heavily inspired from lisp (for functional programming constructs). Python had heavily borrowed the module system, exception model and also keyword arguments from Modula-3 language [K-10]. Pythons' developers strive not to entertain premature optimization, even though it might increase the performance by a few basis points [K-9]. During its design, the creators had conceptualized the language as being a very extensible language, and hence they had designed the language to have a small core library which was extended by a huge standard library [K-7]. Thus as a result, python is used as a scripting language as it can be easily embedded into any application, though it can be used to develop a full-fledged application. The reference implementation of python is CPython. There are also other implementations like Jython, Iron Python which can use python syntax as well as can use any class of Java (Jython) or .Net class (Iron Python). Versions: Python has two versions 2.x version and 3.x version. The 3.x version is a backward incompatible release was released to fix many design issues which plagued the 2.x series. The latest in the 2.x series is 2.7.6 and the latest in 3.x series is 3.4.0. 1.5.2 Paradigms: Python supports multi-

paradigms such as: Object-Oriented, Imperative, Functional, Procedural, and Reflective. In Object-Oriented Paradigm, Python supports most of the OOPs concepts such as Inheritance (It also has support for Multiple Inheritance), Polymorphism but its lack of support for encapsulation is a blatant omission as Python doesn't have private, protected members: all class members are public [K-11]. Earlier Python 2.6 versions didn't support some OOP's concepts such as Abstraction through Interfaces and Abstract Classes [K-19]. It also supports Concurrent paradigm, but with Python we will not be able to make truly multitasking applications as the inbuilt threading API is limited by GIL (Global Interpreter Lock) and hence applications that use the threading API cannot run on multi-core parallelly [K-12].The only remedy is that, the user has to either use the multi-processing module which would fork processes or use Interpreters that haven't implemented GIL such as Jython or Iron Python [K-12]. 1.5.3 Compilation, Execution and Memory Management: Compilation, Execution and Memory Management: 21 A Comparative Studies of Programming Languages (Comparative Studies of Six Programming Language) Just like the other Managed Languages, Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The reference implementation (i.e. CPython) doesn't come with a JIT compiler because of which the execution speed is slow compared to native programming languages [K-17]. We can use PyPy interpreter as it includes a JIT compiler rather than using the Python interpreter that comes by default with the python language, if speed of execution is one of the important factors [K-18]. The Python Runtime Environment also takes care of all the allocation and deallocation of memory through the Garbage Collector. When a new object is created, the GC allocates the necessary memory, and once the object goes out of its scope, the GC doesn't release memory immediately but instead it becomes eligible for Garbage Collection, which would eventually release the memory. Typing

Strategies: Python is a strongly dynamic typed language. Python 3 also supports optional static typing [K-20]. There are a few advantages in using a dynamic typed language, the most prominent one would be that the code is more readable as there is less code (in other words has less boiler-plate code). But the main disadvantage in having python as a dynamic programming language is that there would be no way to guarantee that a particular piece of code would run successfully for all the different data-types scenarios simply because it had run successfully with one type. Basically, we don't have any means to find out an error in the code, till the code has started running. 1.5.4 Strengths and Weaknesses and Application Areas: Python is predominantly used as a scripting language used in developing standalone applications that are being developed with Static-Typed languages, because of the flexibility it provides due to its dynamic typed nature. Python favours rapid application development, which qualifies it to be used for prototyping. To a certain extent, Python is also used in developing websites. Due to its dynamic typing and of the presence of a Virtual Machine, there is a considerable overhead which translates to way less performance when we compare with native programming languages [K-13]. And hence it is not suited

NUMPY

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences: • NumPy arrays have a fixed size at creation, unlike Python lists (which can

grow dynamically). Changing the size of an ndarray will create a new array and delete the original. • The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements. • NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences. • A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays. The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element:

The Numeric Python extensions (NumPy henceforth) is a set of extensions to the Python programming language which allows Python programmers to efficiently manipulate large sets of objects organized in grid-like fashion. These sets of objects are called arrays, and they can have any number of dimensions: one dimensional arrays are similar to standard Python sequences, two-dimensional arrays are similar to matrices from linear algebra. Note that one-dimensional arrays are also different from any other Python sequence, and that two-dimensional matrices are also different from the matrices of linear algebra, in ways which we will mention later in this text. Why are these extensions needed? The core reason is a very prosaic one, and that is that manipulating a set of a million numbers in Python with the standard data structures such as lists, tuples or classes is much too slow and uses too much space. Anything which we can do in NumPy we can do in standard Python – we just may not be alive to see the program finish. A more subtle reason for these extensions however is that the kinds of operations that programmers typically want to do on arrays, while sometimes very complex, can often be decomposed into a set of fairly standard operations. This decomposition has

been developed similarly in many array languages. In some ways, NumPy is simply the application of this experience to the Python language – thus many of the operations described in NumPy work the way they do because experience has shown that way to be a good one, in a variety of contexts. The languages which were used to guide the development of NumPy include the infamous APL family of languages, Basis, MATLAB, FORTRAN, S and S+, and others. This heritage will be obvious to users of NumPy who already have experience with these other languages. This tutorial, however, does not assume any such background, and all that is expected of the reader is a reasonable working knowledge of the standard Python language. This document is the "official" documentation for NumPy. It is both a tutorial and the most authoritative source of information about NumPy with the exception of the source code. The tutorial material will walk you through a set of manipulations of simple, small, arrays of numbers, as well as image files. This choice was made because:

• A concrete data set makes explaining the behavior of some functions much easier to motivate than simply talking about abstract operations on abstract data sets;

• Every reader will at least an intuition as to the meaning of the data and organization of image files, and

• The result of various manipulations can be displayed simply since the data set has a natural graphical representation. All users of NumPy, whether interested in image processing or not, are encouraged to follow the tutorial with a working NumPy installation at their side, testing the examples, and, more importantly, transferring the understanding gained by working on images to their specific domain. The best way to learn is by doing – the aim of this tutorial is to guide you along this "doing."

TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software Testing can also be stated as the process of validating and verifying that a software program/application/product:

- Meets the business and technical requirements that guided its design and Development.
- Works as expected and can be implemented with the same characteristics.

TESTING METHODS

- **Functional Testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:

- Functions: Identified functions must be exercised.
- Output: Identified classes of software outputs must be exercised.
- Systems/Procedures: system should work properly

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Test Case for Excel Sheet Verification:

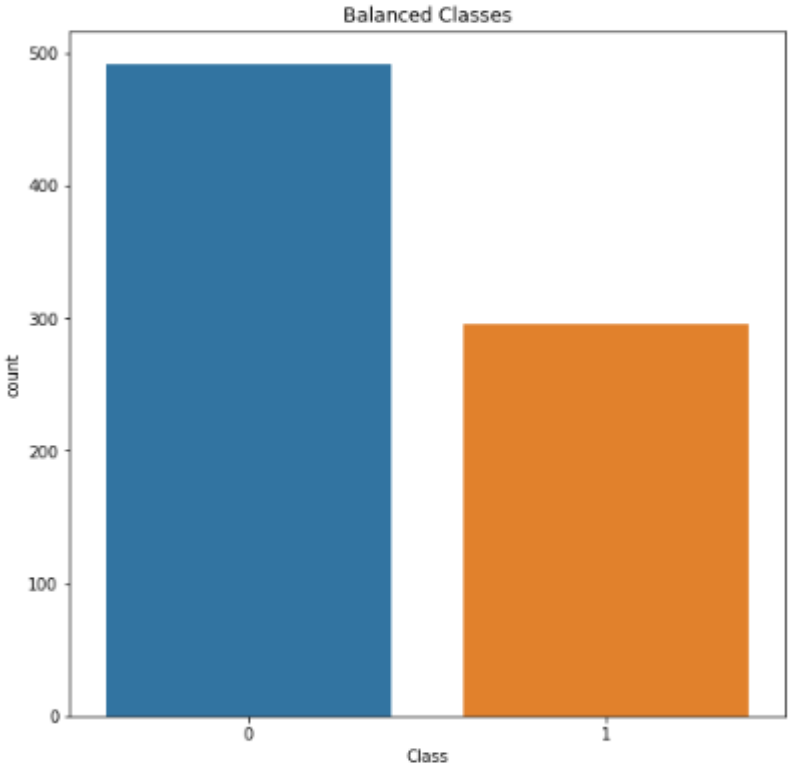
Here in machine learning we are dealing with dataset which is in excel sheet format so if any test case we need means we need to check excel file. Later on classification will work on the respective columns of dataset .

Test Case 1 :

SL #	TEST CASE NAME	DESCRIPTION	STEP NO	ACTION TO BE TAKEN (DESIGN STEPS)	EXPECTED (DESIGN STEP)	Test Execution Result (PASS/FAIL)
1	Excel Sheet verification	**Objective:** There should be an excel sheet. Any number of rows can be added to the sheet.	Step 1	Excel sheet should be available	Excel sheet is available	Pass
			Step 2	Excel sheet is created based on the template	The excel sheet should always be based on the template	Pass
			Step 3	Changed the name of excel sheet	Should not make any modification on the name of excel sheet	Fail
			Step 4	Added 10000 or above records	Can add any number of records	Pass

Results:

Data Exploration:



Confusion Matrix :

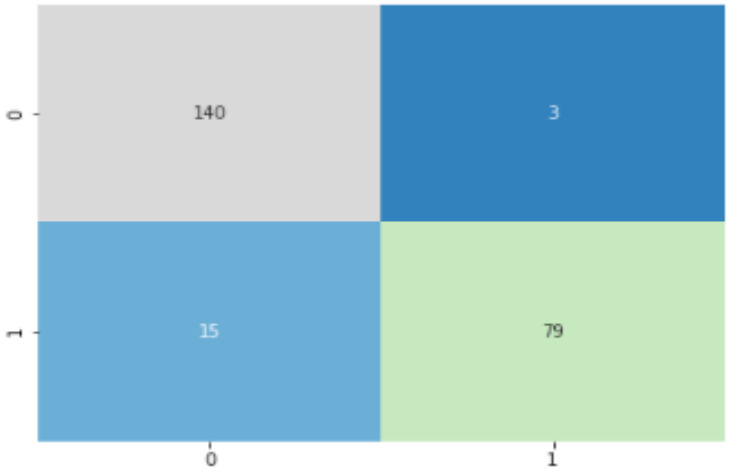


Fig : SVM confusion matrix

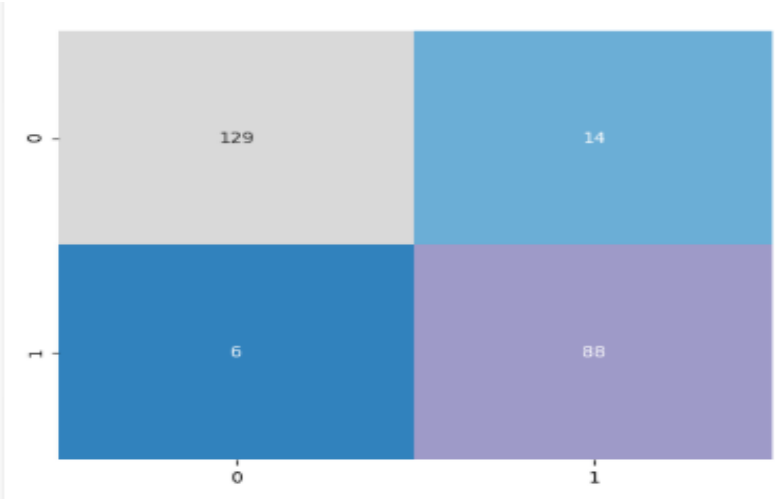


Fig : Decision tree confusion matrix

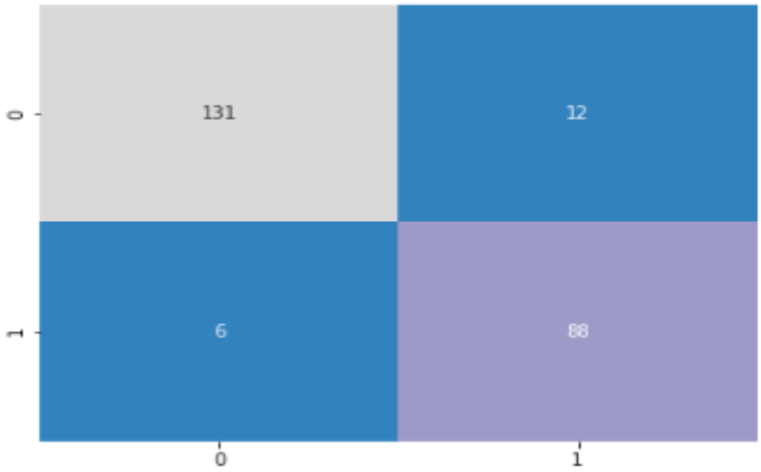


Fig : Adaboost confusion matrix

Models Evaluation:

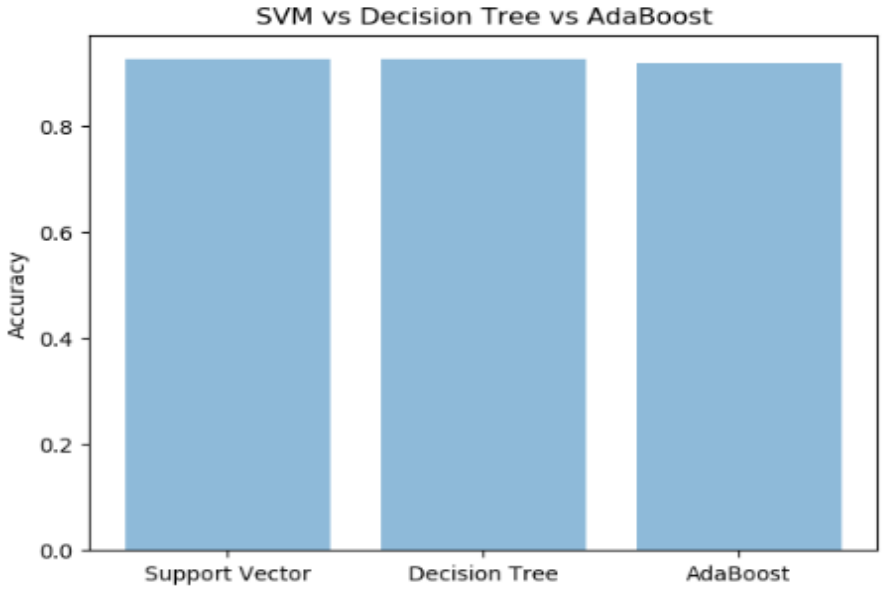


Fig : Comparative accuracy result between several classifiers

Conclusion

The proposed paper evaluate that the Decision tree and support vector machine algorithm will perform better with a larger number of training data comparing to Adaboost classifier, but speed during testing and application will suffer. Application of more pre-processing techniques would also help. The SVM algorithm still suffers from the imbalanced dataset problem and requires more pre-processing to give better results at the results shown by SVM is great but it could have been better if more pre-processing have been done on the data.so, in proposed work we balanced the imbalanced data with up-sampling technique during pre-processing. We review the existing works on credit card fraud prediction in three different perspectives: datasets, methods, and metrics. Firstly, we present the details about the availability of public datasets and what kinds of details are available in each dataset for predicting credit card fraud. Secondly, we compare and contrast the various predictive modeling methods that have been used in the literature for predicting, and then quantitatively compare their performances in terms of accuracy.

REFERENCES

[1] P. Richhariya and P. K. Singh, "Evaluating and emerging payment card fraud challenges and resolution," International Journal of Computer Applications, vol. 107, no. 14, pp. 5 – 10, 2014.

[2] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," Decision Support Systems, vol. 50, no. 3, pp. 602–613, 2011.

[3] A.DalPozzolo,O.Caelen,Y.-A.LeBorgne,S.Waterschoot,andG.Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," Expert systems with applications, vol. 41, no. 10, pp. 4915–4928, 2014.

[4] C. Phua, D. Alahakoon, and V. Lee, "Minority report in fraud detection: classification of skewed data," ACM SIGKDD explorations newsletter, vol. 6, no. 1, pp. 50–59, 2004.

[5] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 1, pp. 63–77, 2006.

[6] S. Ertekin, J. Huang, and C. L. Giles, "Active learning for class imbalance problem," The 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 823–824, 2007.

[7] M.WasikowskiandX.-w.Chen,"Combatingthesmallsampleclassimbalance problem using feature selection," IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1388–1400, 2010. [8] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, no. 4, pp. 1119–1130, 2012.

[9] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," Statistical science, pp. 235–249, 2002.

[10] D. J. Weston, D. J. Hand, N. M. Adams, and C. Whitrow, "Plastic card fraud detection using peer group analysis," vol. 2, pp. 45–62, 2008.

[11] E. Duman and M. H. Ozcelik, "Detecting credit card fraud by genetic algorithm and scatter search," Expert Systems with Applications, vol. 38, no. 10, pp. 13057–13063, 2011.

[12] K. Ramakalyani and D. Umadevi, “Fraud Detection of Credit Card Payment System by Genetic Algorithm,” International Journal of Scientific & Engineering Research, vol. 3, no. 7, pp. 1–6, 2012.
[13] P. J. Bentley, J. Kim, G.-h. Jung, and J.-u. Choi, “Fuzzy Darwinian Detection of Credit Card Fraud,” pp. 1–4, 2007.
[14]A.Srivastava,A.Kundu,S.Sural,andS.Member,“CreditCardFraudDetection Using Hidden Markov Model,” IEEE Transactions on Dependable and Secure Computing, vol. 5, no. 1, pp. 37–48, 2008.
[15] S. Esakkiraj and S. Chidambaram, “Predictive Approach for Fraud Detection Using Hidden Markov Model,” International Journal of Engineering Research & Technology, vol. 2, no. 1, pp. 1–7, 2013.
[16] J. S. Mishra, S. Panda, and A. K. Mishra, “A Novel Approach for Credit Card Fraud Detection Targeting the Indian Market,” International Journal of Computer Science, vol. 10, no. 3, pp. 172–179, 2013.
[17] A. Brabazon, J. Cahill, P. Keenan, and D. Walsh, “Identifying Online Credit Card Fraud using Artificial Immune Systems,” IEEE Congress on Evolutionary Computation, pp. 1 – 7, 2010.
[18] N. Wong, P. Ray, G. Stephens, and L. Lewis, “Artificial immune systems for the detection of credit card fraud: an architecture, prototype and preliminary results,” Information systems, vol. 22, pp. 53–76, 2012.
[19] D. Sanchez, M. A. Vila, L. Cerda, and J. M. Serrano, “Association rules applied to credit card fraud detection,” ScienceDirect, vol. 36, pp. 3630– 3640, 2009.
[20] Y. Sahin, S. Bulkan, and E. Duman, “ A cost-sensitive decision tree approach for fraud detection,” Expert Systems with Applications, vol. 40, pp. 5916–5918, 2013.
[21] A. C. Bahnsen, A. Stojanovic, and D. Aouada, “Cost Sensitive Credit Card Fraud Detection using Bayes Minimum Risk,” 12th International Conference on Machine Learning and Applications, pp. 333–338, 2013.
[22] A. E. Pasarica, “Card fraud detection using learning machines,” The Bulletin of the Polytechnic Institute of Jassy, pp. 29 – 45, 2014.
[23] Y. Sahin and E. Duman, “Detecting Credit Card Fraud by Decision Trees and Support Vector Machines,” International Multiconference of Engineers and computer scientists, vol. 1, pp. 442–447, 2011.
[24]V.R.GanjiandS.N.P.Mannem,“Creditcardfrauddetectionusinganti-k nearest neighbor algorithm,” International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 06, pp. 1035–1039, 2012.

[25] S. Ghosh and D. L. Reilly, “Credit Card Fraud Detection with a NeuralNetwork,” Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences, 1994, pp. 621–630, 1994.
[26]R.Dorronsoro,F.Ginel,S.Carmen,andC.S.Cruz,“NeuralFraudDetection in Credit Card Operations,” IEEE Transactions on Neural Networks, vol. 8, no. 4, pp. 827–834, 1997.
[27] V. Zaslavsky and A. Strizhak, “Credit card fraud detection using selforganizing maps,” Information and Security, vol. 18, pp. 48–63, 2006.
[28] F. N. Ogwueleka, “Data Mining Application in Credit Card Fraud Detection System,” Journal of Engineering Science and Technology, vol. 6, no. 3, pp. 311–322, 2011. [29] R. Patidar and L. Sharma, “Credit Card Fraud Detection Using Neural Network,” International Journal of Soft Computing and Engineering, no. May, pp. 13–14, 2011.
[30] M. Syeda, Y.-q. Zbang, Y. Pan, and C. Science, “Parallel Granular Neural Networks for Fast Credit Card Fraud Detection,” Proceedings of the IEEE International Conference on Fuzzy Systems., vol. 1, pp. 572–577, 2002. [31] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, “Credit card fraud detection using bayesian and neural networks,” Interactive imageguided neurosurgery. American Association Neurological Surgeons, pp. 261–270, 1993.