

A Project on

FRAUD DETECTION USING MACHINE

LEARNING

Submitted for partial fulfillment of award of

MASTER OF SCIENCE

Dr. Homi Bhabha University, Mumbai

Degree in

MATHEMATICS

By

Pooja Patil

Pranjali Jadhav

Wahabuddin Ansari

Under the Guidance of

Dr. Selby Jose

Department of Mathematics

The Institute of Science

Mumbai - 400032

JUNE, 2022

Declaration

I hereby declare that the project work entitled “**FRAUD DETECTION USING MACHINE LEARNING**” carried out at the Department of Mathematics, The Institute of Science, Mumbai, is a record of an original work done by me under the guidance of **Dr. Selby Jose**, The Institute of Science, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Master of Science in Mathematics, Dr.Homi Bhabha State University, Mumbai The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Pooja Patil

Seat No. **202020**

Certificate

This is to certify that the project report entitled **FRAUD DETECTION USING MACHINE LEARNING**, carried out at the Department of Mathematics, The Institute of Science, Mumbai, in partial fulfillment for the award of the degree of Master of Science in Mathematics, Dr. Homi Bhabha State University, Mumbai, is a record of bonafide work carried out by **Ms.Pooja Patil**, Seat No. **202020**, **Ms.Pranjali Jadhav**, Seat No. **202021**, **Mr.Wahabuddin Ansari**, Seat No. **202040**, under the supervision and guidance of **Dr. Selby Jose** during the academic year 2020 – 2022.

Dr. Selby Jose
Project Guide

Dr .Sunil Singh
External Examiner

Head of Dept.
Mathematics

Place: Mumbai

Date:

Acknowledgment

Foremost, we thank the Almighty for showering his blessings upon me. This research would not have been possible without his grace.

We wish to express a sense of gratitude and love to our beloved parents for their encouragement and unwavering support throughout. We would like to express our deep regards to **Dr. Selby Jose**, Head of the Department of Mathematics and the entire Department of Mathematics for giving us this opportunity.

With immense gratitude, we would like to give a very special honor and respect to our mentor **Dr. Selby Jose** who took keen interest and guided us throughout the project work. His continuous support, patience, motivation, enthusiasm and immense knowledge not only helped in making my efforts fruitful but also transformed the whole process of learning and implementation into an enjoyable experience. We are also pleased to thank other project guides Prof. **Miss Prachee Angane** for her insightful guidance.

Pooja Patil

Abstract

It is vital that Automobile companies are able to identify frauds so that customers do not face any loss. Such problems can be tackled with Data Science and its importance, along with Machine Learning, cannot be overstated. This project intends to illustrate the modelling of a data set using machine learning with Automobile Fraud Detection.

The Automobile Fraud Detection Problem includes modelling frauds report with the data of the ones that turned out to be fraud. This model is then used to recognize whether a fraud is reported or not. Our objective here is to detect the fraudulent while minimizing the incorrect fraud classifications.

Automobile Fraud Detection is a typical sample of classification. In this process, we have focused on analyzing and pre-processing data sets. We built models and split the data into training set and testing set. Then we used various models like **Logistic Regression, Support Vector Machine, K -Nearest Neighborhood, Decision Tree and Random Forest** to check the accuracy to decide which model is best choice for our data set. Since **DECISION TREE** gave the highest accuracy after comparing the models. So we selected **Decision Tree** as the best model.

Contents

1	Introduction	1
1.1	History	1
1.2	Why we are interested in this topic?	2
2	Preliminaries	4
2.1	Basic Terms and Concepts	4
2.2	Machine Learning Process	5
2.3	Classification Analysis	6
3	Fraud Detection and Machine Learning	15
3.1	Why is machine learning suited to fraud detection?	15
3.2	How does a machine learning system work?	17
3.3	How can one tell the model is working?	19
4	Analysis of Fraud Detection	21
4.1	Basic Tools and Codes	21
4.2	Models and Their Predictions	32
4.3	Comparison of Models	37
	Bibliography	38

Chapter 1

Introduction

FRAUD: The simplest and easiest way of gaining money or financial benefits by a trick or by lying is called as “**Fraud**”. Fraud is a broad legal term referring to dishonest acts that intentionally use deception to illegally deprive another person or entity of money, property, or legal rights.

1.1 History

FIRST PERSON TO COMMIT FRAUD: The first recorded instance comes from 300 BC, in Greece, when shipping merchant Hegestratos changed the world by attempting to con the insurers of a shipload of valuable goods by sinking his boat but keeping the cargo, and claiming the loss anyway.

OCCURRENCE OF FRAUD: For fraud to occur, the three elements of the Fraud Triangle must be present.: **Opportunity**, **Pressure** (also known as incentive or motivation) and **Rationalization** (sometimes called justification or attitude).

TYPES OF FRAUD:

- Identity theft and identity fraud.
- Tax fraud.
- Credit card fraud.
- Insurance fraud.
- Property Fraud, etc.

1.2 Why we are interested in this topic?

Fraud has touched nearly every area of business – from data breaches that affect end customers’ privacy rights and payment security, to ransom attacks that demand vast sums of money from organizations. When not dealt with proactively, fraud can affect companies’ bottom lines by pulling resources from the core business and priorities, damaging brand reputation, and squandering profits. In extreme cases, it can even cost human lives.

In the past, fraud prevention efforts lagged behind the incredible speed with which fraudsters acted. Fraudsters were – and still are – a constantly moving target. But investigators were slow, cross-referencing data and findings manually, which was a heavy time and resource-consuming undertaking. It also meant a fragmented approach. Investigators couldn’t generate deep analytics in real-time, monitor activity in all corners of the web, and act fast.

Most investigations had a starting and an ending point whereas fraudsters never stopped. Their efforts were ongoing, snowballing into a global contagion. Attackers and fraudsters exploited investigators' and cybersecurity teams' inefficiencies and continued to evolve the sophistication of their methods to levels never seen before.

In the Later Chapters, we discuss the following:

1. Defined all the Machine Learning Models that were gonna be used for the detection of fraud in our dataset.
2. Talked about the relationship between Fraud Detection and Machine Learning, that how Super fast, Scalable, More accurate and Efficient it is.
3. How Machine learning system works?
4. How a Machine learning model is created?
5. How can you tell a Machine learning model is good?
6. Comparing all the Models to see which one is the best.

Chapter 2

Preliminaries

2.1 Basic Terms and Concepts

1. **Machine Learning (ML):** Machine learning is a subset of Artificial Intelligence (AI) which provides machine the ability to learn automatically and improve from experience without being explicitly programmed.
2. **Algorithm:** An algorithm is a set of rules and statistical techniques used to learn patterns from the data. It is the logic behind the ML model.
3. **Model:** A model is the main component which is trained by using a ML algorithm.
4. **Predictor Variable (X):** It is a feature(s) of the data that can be used to predict the output.
5. **Response Variable (Y):** It is the feature or the output variable that needs to be predicted by using the predictor variable.
6. **Training Data:** The machine learning model is built using the training data.

7. **Testing Data:** The machine learning model is evaluated using the testing data.

2.2 Machine Learning Process

1. **Define the Objective of the Problem**

This step involves clearing understanding and defining the problem for which we have to find the solution.

2. **Data Gathering**

Choosing where to get the data from. The outcome of this step is generally a representation of data which we will use for training

3. **Preparing Data**

Wrangle data and prepare it for training. This step involves cleaning the data which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

4. **Data Exploration**

Visualize data to help detect relevant relationships between variables or class imbalances, or perform other exploratory analysis. Split the data into training and evaluation sets

5. **Building a Model**

Developing a model that does better than a baseline

6. **Model Evaluation and Optimization Scaling up**

Developing a model that overfits. Regularizing the model and tuning the parameters

7. Predictions

Using further (test set) data which have, until this point, been withheld from the model (and for which class labels are known), are used to test the model; a better approximation of how the model will perform in the real world

2.3 Classification Analysis

The Classification Analysis is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.

Types of ML Classification Algorithms:

1. Linear Classification Models

A linear classifier is a model that makes a decision to categories a set of data points to a discrete class based on a linear combination of its explanatory variables.

Types of Linear Classification Models:

■ Logistic Regression

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No.

In logistic regression, a categorical dependent variable Y having n (usually $n = 2$) unique values is regressed on a set of p independent variables X_1, X_2, \dots, X_p . The logistic regression model is given by

the equation

$$\ln \frac{p_n}{p_1} = \ln \frac{p_n}{p_1} + \beta_{n_1}X_1 + \beta_{n_2}X_2 + \dots + \beta_{n_p}X_p$$

Here, p_n is the probability that an individual with values X_1, X_2, \dots, X_p is in outcome n . That is,

$$p_n = P_r(Y = n|X)$$

The quantities p_1, p_2, \dots, p_n represent the prior probabilities of outcome membership. If these prior probabilities are assumed equal, then the term $\ln(p_n/p)$ becomes zero and drops out. If the priors are not assumed equal, they change the values of the intercepts in the logistic regression equation.

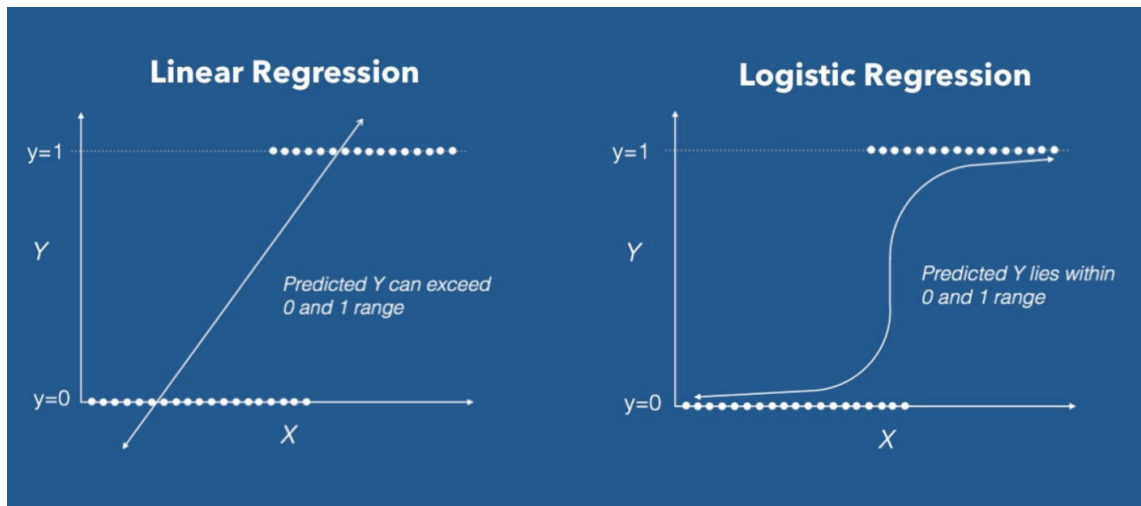


Figure 2.1: Logistic regression

■ Support Vector Machines.

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n -dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

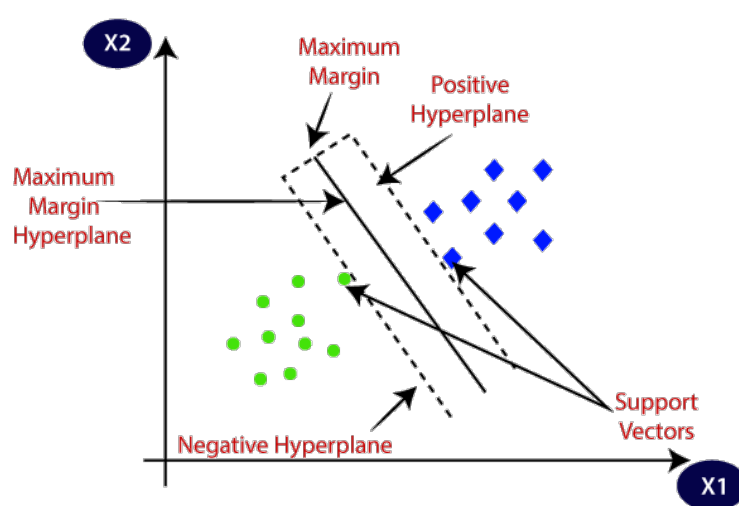


Figure 2.2: support-vector-machine-algorithm

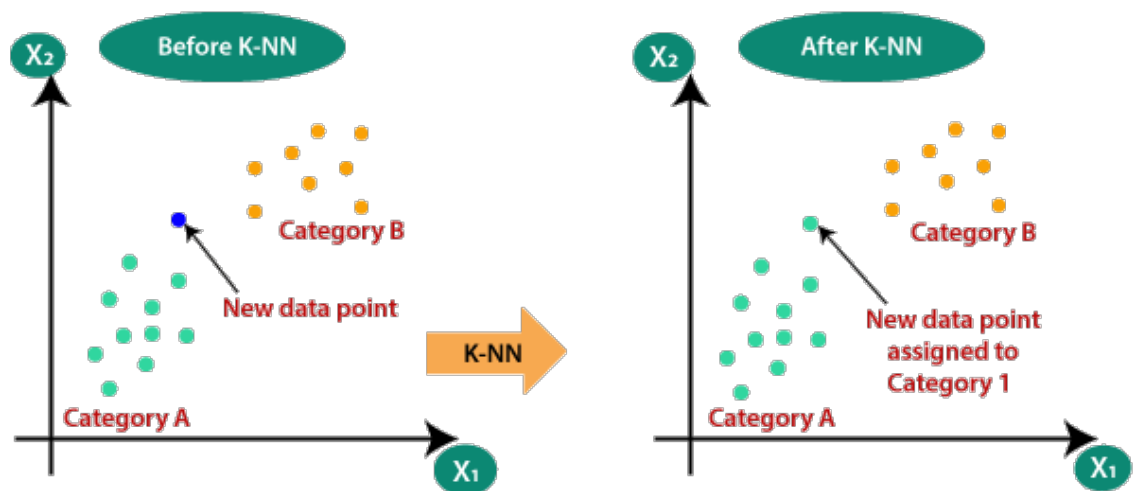
2. Non-Linear Classification Models

Nonlinear functions can be used to separate instances that are not linearly separable.

Types of non linear Classification models

■ *K*-Nearest Neighbours.

- *K*-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- *K*-NN algorithm assumes the similarity between the new case/-data and available cases and put the new case into the category that is most similar to the available categories.
- *K*-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using *K*- NN algorithm.
- *K*-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- *K*-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- *K*-NN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Figure 2.3: k -nearest-neighbor-algorithmFigure 2.4: k -nearest-neighbor-algorithm2

■ Kernel SVM.

The below example is of non-linear data:

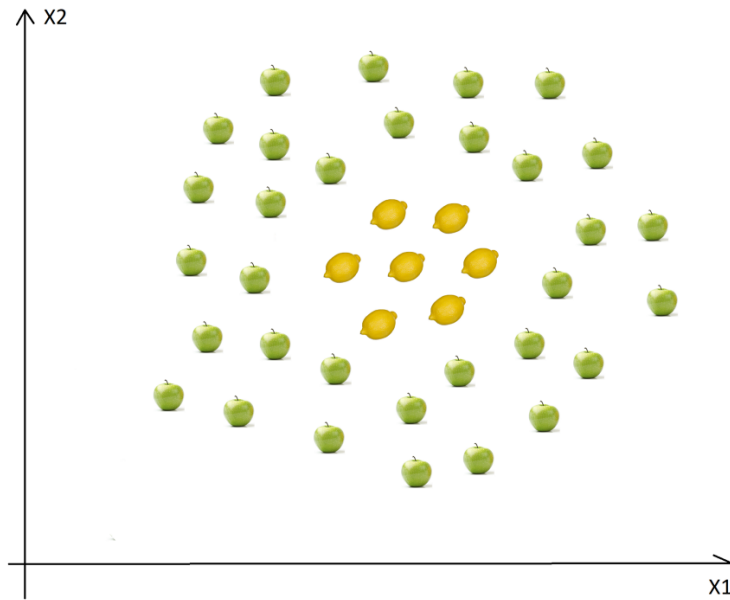


Figure 2.5: Kernel svm

In this case we cannot find a straight line to separate apples from lemons. So how can we solve this problem. We will use the Kernel Trick!

The basic idea is that when a data set is inseparable in the current dimensions, add another dimension, maybe that way the data will be separable. Just think about it, the example above is in 2D and it is inseparable, but maybe in 3D there is a gap between the apples and the lemons, maybe there is a level difference, so lemons are on level one and apples are on level two. In this case, we can easily draw a separating hyperplane (in 3D a hyperplane is a plane) between level 1

and 2.

To solve this problem we shouldn't just blindly add another dimension, we should transform the space so we generate this level difference intentionally.

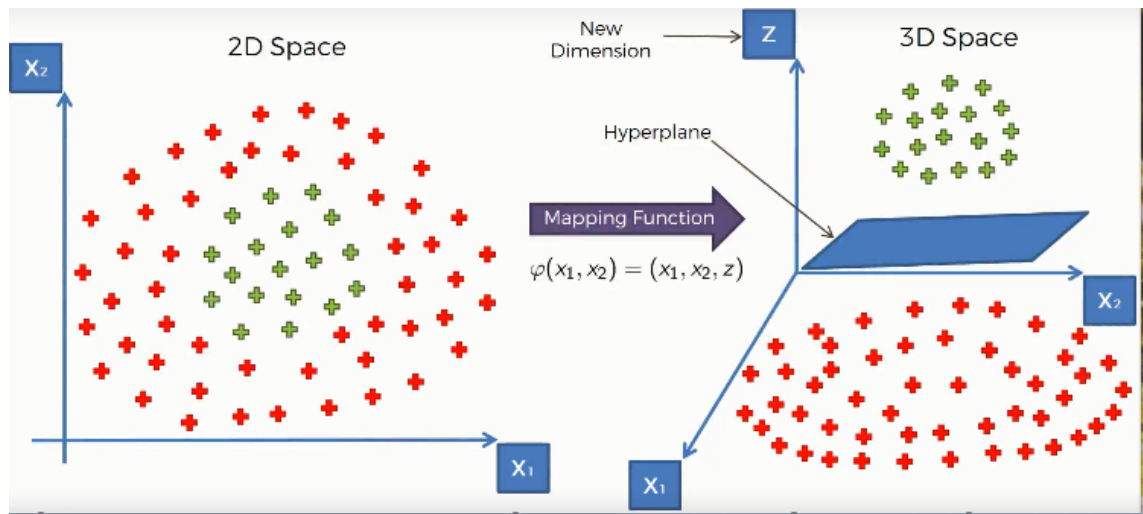


Figure 2.6: Kernel_svm

■ Decision Tree Classification.

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with

the root node, which expands on further branches and constructs a tree-like structure.

- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

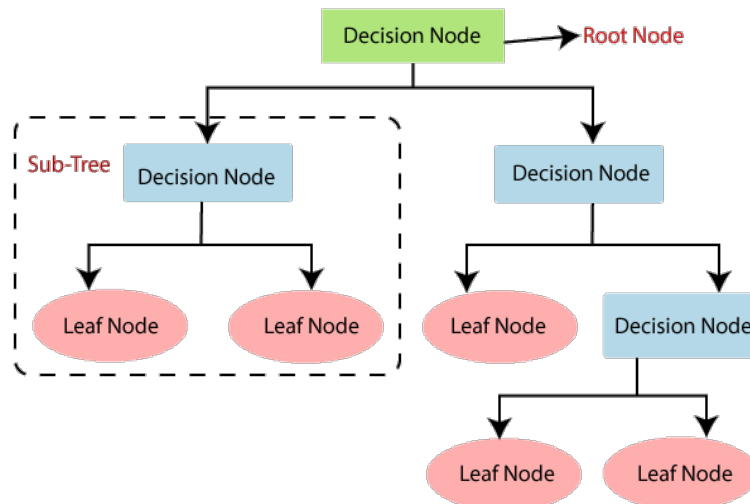


Figure 2.7: decision-tree-classification-algorithm

■ Random Forest Classification.

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

- As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.
- The below diagram explains the working of the Random Forest algorithm:

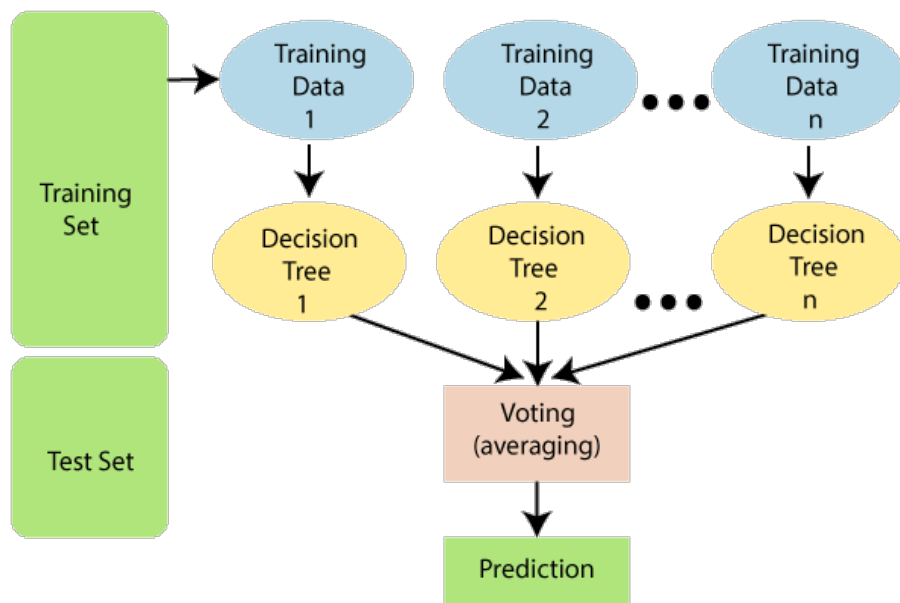


Figure 2.8: random-forest-algorithm

Chapter 3

Fraud Detection and Machine Learning

3.1 Why is machine learning suited to fraud detection?

1. Super fast

When it comes to fraud decisions, you need results FAST! Research shows that the longer a buyer's journey takes the less likely they are to complete checkout.

Machine learning is like having several teams of analysts running hundreds of thousands of queries and comparing the outcomes to find the best result - this is all done in real-time and only takes milliseconds.

As well as making real-time decisions, machine learning is assessing individual customer behaviour as it happens. It's constantly analyzing 'normal' customer activity, so when it spots an anomaly it can automatically block or flag a payment for analyst review.

2. Scalable

Every online business wants to increase its transaction volume. With a rules only system, increasing amounts of payment and customer data puts more pressure on the rules library to expand. But with machine learning it's the opposite - the more data the better.

Machine learning systems improve with larger datasets because this gives the system more examples of good and bad eg. genuine and fraudulent customers. This means the model can pick out the differences and similarities between behaviors more quickly and use this to predict fraud in future transactions.

3. Efficient (and cheap!)

Remember that machine learning is like having several teams running analysis on hundreds of thousands of payments per second. The human cost of this would be immense - the cost of machine learning is just the cost of the servers running.

Machine learning does all the dirty work of data analysis in a fraction of the time it would take for even 100 fraud analysts. Unlike humans, machines can perform repetitive, tedious tasks 24/7 and only need to escalate decisions to a human when specific insight is needed.

4. More accurate

In the same way, machine learning can often be more effective than humans at uncovering non-intuitive patterns or subtle trends which might only be obvious to a fraud analyst much later.

Machine learning models are able to learn from patterns of normal behavior. They are very fast to adapt to changes in that normal behaviour and can quickly identify patterns of fraud transactions.

3.2 How does a machine learning system work?

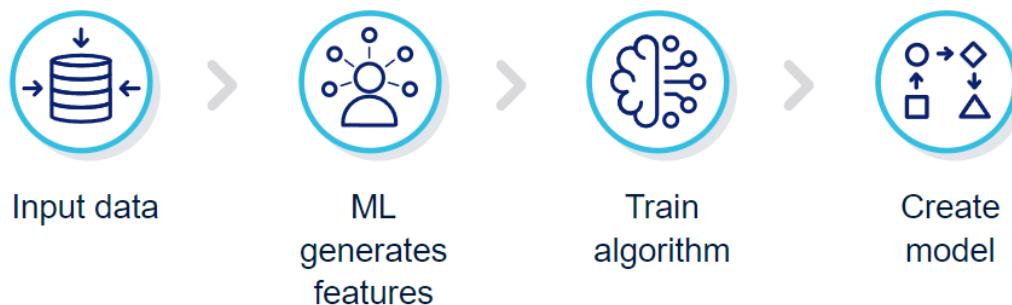


Figure 3.1: working of ML

1. Input data

When it comes to fraud detection, the more data the better.

For supervised machine learning, the data must be labelled as good (genuine customers who have never committed fraud) or bad (customers with a chargeback associated with them or have been manually labelled as fraudsters).

2. Extract features

Features describe customer behaviour, and fraudulent behaviours are known as fraud signals.

At Ravelin, we group features into five main categories, each of which has hundreds or thousands of individual features:



Figure 3.2: working of ML

- **Identity**

Number of digits in the customer's email address, age of their account, number of devices customer was seen on, fraud rate of customer's IP address.

- **Orders**

Number of orders they made in their first week, number of failed transactions, average order value, risky basket contents.

- **Payment methods**

Fraud rate of issuing bank, similarity between customer name and billing name, cards from different countries.

- **Locations**

Shipping address matches the billing address, shipping country matches country of customer's IP address, fraud rate at customer's location.

- **Network**

Number of emails, phone numbers or payment methods shared within a network, age of the customer's network.

3. Train algorithm

An algorithm is a set of rules to be followed when solving complex problems, like a mathematical equation or even a recipe. The algorithm uses customer data described by our features to learn how to make predictions eg. fraud/not fraud.

In the beginning, we'll train the algorithm on an online seller's own historical data, we call this a training set. The more fraud in this training set the better, so that the machine has lots of examples to learn from.

4. Create a model

When training is complete you have a model specific to your business, which can detect fraud in milliseconds.

We constantly keep an eye on the model to make sure it is behaving as it should, and we're always looking for ways to improve it. We regularly improve, update and upload a new model for every client so that the system will always detect the latest fraud techniques.

3.3 How can one tell the model is working?

After the training, to check that the model is working correctly, we show the model some data which it has never seen before, but which we know the fraud outcomes for. If the model detects the fraud correctly, we can deploy it to be used against the online business's transactions. We also do some automatic common-sense analysis on recent data for which we do not have fraud labels to ensure the model will behave correctly when it is deployed.

There are certain fraudulent situations which the model should always pick up on - some examples are:

- High velocity of new payment methods eg. a customer adds new 10 payment cards in an hour
- Suspicious email address eg. a mismatch between the account name or name on the card, or rude/naughty words in the email
- A customer placing lots of orders of high value good eg. luxury alcohol
- Orders from a particularly fraudy location, shipping to a known fraud hotspot or a PO Box rather than a residential address

Chapter 4

Analysis of Fraud Detection

4.1 Basic Tools and Codes

1. Importing Libraries

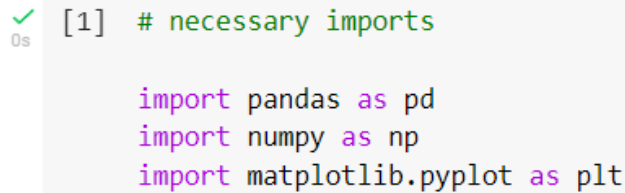
Importing libraries into a program is a cost-effective way of minimizing the high-risk aspect of designing, developing, and testing software that has already gone through the development cycle.

Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy.

The import statement allows you to import one or more modules into your Python program, letting you make use of the definitions constructed in those modules.

How to import modules in Python?

#import statement is used to import Libraries.

A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark and the text '0s'. The cell contains the following Python code:

```
[1] # necessary imports

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Figure 4.1: importing libraries

2. Scikitlearn

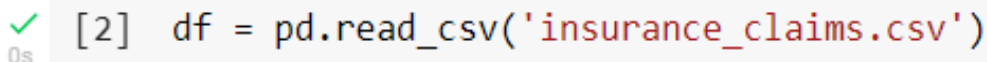
Scikitlearn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

3. Uploading Data

Data is the most important part of all Data Analytics, Machine Learning, Artificial Intelligence. Without data, we can't train any model and all modern research and automation will go in vain.

It is the data that we need to load for starting any of the ML project. With respect to data, the most common format of data for ML projects is CSV (comma-separated values). Basically, CSV is a simple file format which is used to store tabular data (number and text) such as a spreadsheet in plain text.

The best way to load CSV data file is by Pandas and `pandas.read_csv()` function. This is the very flexible function that returns a `pandas.DataFrame` which can be used immediately for plotting.

A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark and the text '0s'. The cell contains the following Python code:

```
[2] df = pd.read_csv('insurance_claims.csv')
```

Figure 4.2: uploading Data

4. Describing the DataFrame

The **describe()** method returns description of the data in the DataFrame.

If the DataFrame contains numerical data, the description contains these information for each column:

count - The number of not-empty values.

mean - The average (mean) value.

std - The standard deviation.

min - the minimum value.

25% - The 25% percentile*.

50% - The 50% percentile*.

75% - The 75% percentile*.

max - the maximum value.

Note: Percentile* meaning: how many of the values are less than the given percentile.

✓ [6] df.describe()

	months_as_customer	age	policy_number	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	capital-gains
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000
mean	203.954000	38.948000	546238.648000	1136.000000	1256.406150	1.101000e+06	501214.488000	25126.100000
std	115.113174	9.140287	257063.005276	611.864673	244.167395	2.297407e+06	71701.610941	27872.187708
min	0.000000	19.000000	100804.000000	500.000000	433.330000	-1.000000e+06	430104.000000	0.000000
25%	115.750000	32.000000	335980.250000	500.000000	1089.607500	0.000000e+00	448404.500000	0.000000
50%	199.500000	38.000000	533135.000000	1000.000000	1257.200000	0.000000e+00	466445.500000	0.000000
75%	276.250000	44.000000	759099.750000	2000.000000	1415.695000	0.000000e+00	603251.000000	51025.000000
max	479.000000	64.000000	999435.000000	2000.000000	2047.590000	1.000000e+07	620962.000000	100500.000000

Figure 4.3: describe

5. Information about the DataFrame

The `info()` method prints information about the DataFrame.

The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

```
[7] df.info()
```


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 40 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   months_as_customer                    1000 non-null   int64
1   age                                   1000 non-null   int64
2   policy_number                         1000 non-null   int64
3   policy_bind_date                      1000 non-null   object
4   policy_state                          1000 non-null   object
5   policy_csl                            1000 non-null   object
6   policy_deductable                     1000 non-null   int64
7   policy_annual_premium                 1000 non-null   float64
8   umbrella_limit                        1000 non-null   int64
9   insured_zip                           1000 non-null   int64
10  insured_sex                           1000 non-null   object
11  insured_education_level                1000 non-null   object
12  insured_occupation                     1000 non-null   object
13  insured_hobbies                        1000 non-null   object
14  insured_relationship                   1000 non-null   object
15  capital_gains                          1000 non-null   int64
16  capital_loss                           1000 non-null   int64
17  incident_date                          1000 non-null   object
18  incident_type                          1000 non-null   object
19  collision_type                         822 non-null    object
20  incident_severity                      1000 non-null   object
21  authorities_contacted                  1000 non-null   object
22  incident_state                         1000 non-null   object
23  incident_city                          1000 non-null   object
24  incident_location                      1000 non-null   object
25  incident_hour_of_the_day               1000 non-null   int64
26  number_of_vehicles_involved            1000 non-null   int64
27  property_damage                        640 non-null    object
28  bodily_injuries                       1000 non-null   int64
29  witnesses                              1000 non-null   int64
30  police_report_available                657 non-null    object
31  total_claim_amount                     1000 non-null   int64
32  injury_claim                           1000 non-null   int64
33  property_claim                         1000 non-null   int64
34  vehicle_claim                          1000 non-null   int64
35  auto_make                              1000 non-null   object
36  auto_model                             1000 non-null   object
37  auto_year                              1000 non-null   int64
38  fraud_reported                         1000 non-null   object
39  _c39                                   0 non-null      float64
dtypes: float64(2), int64(17), object(21)
memory usage: 312.6+ KB
```

6. Data Preprocessing

Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models.

(a) Finding missing values in the DataFrame

The `isna()` method returns a DataFrame object where all the values are replaced with a Boolean value True for NA (not-a -number) values, and otherwise False.



```
# missing values
df.isna().sum()

months_as_customer      0
age                     0
policy_number           0
policy_bind_date        0
policy_state            0
policy_csl              0
policy_deductable       0
policy_annual_premium   0
umbrella_limit          0
insured_zip             0
insured_sex             0
insured_education_level 0
insured_occupation      0
insured_hobbies         0
insured_relationship    0
capital-gains           0
capital-loss            0
incident_date           0
incident_type           0
collision_type          178
incident_severity       0
authorities_contacted   0
incident_state          0
incident_city           0
incident_location       0
incident_hour_of_the_day 0
number_of_vehicles_involved 0
property_damage         360
bodily_injuries         0
witnesses              0
police_report_available 343
total_claim_amount      0
injury_claim            0
property_claim          0
vehicle_claim           0
auto_make              0
auto_model             0
auto_year              0
fraud_reported          0
_c39                   1000
dtype: int64
```

Figure 4.4: isna

(b) Visualizing missing values

Missingno is a Library used as msno. It lets you quickly visually pick out patterns in data completion.

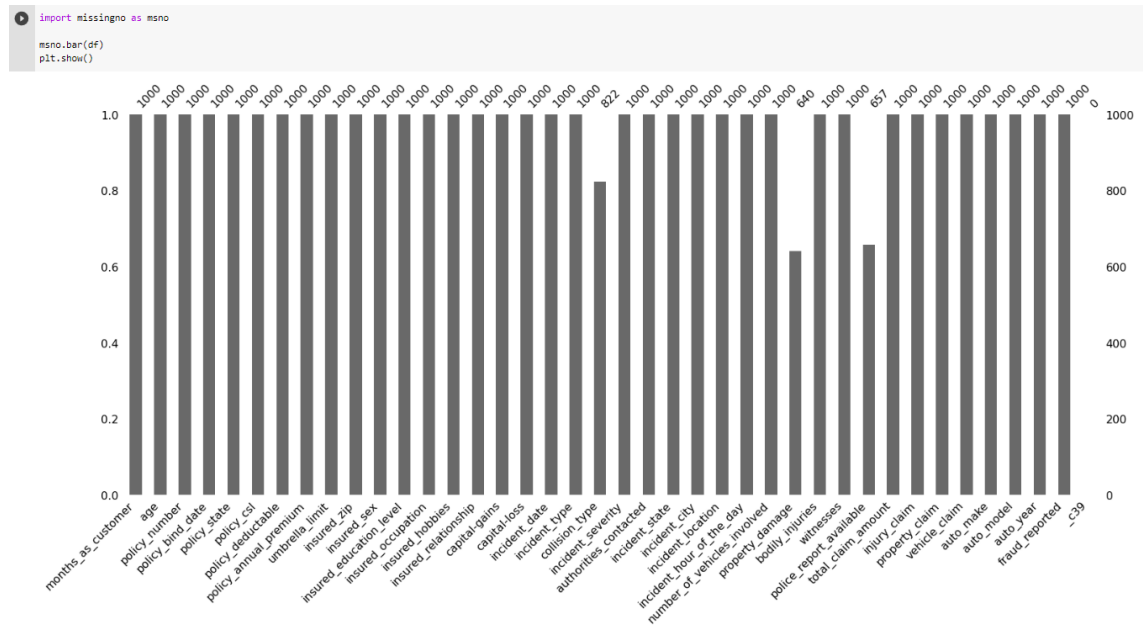


Figure 4.5: Missing Values

(c) Handling missing values

There are three methods for handling the missing values:

- **The mean method**

The **mean()** method returns a Series with the mean value of each column.

By specifying the column axis (`axis='columns'`), the **mean()** method searches column-wise and returns the mean value for each row.

- **The median method**

The **median()** method returns a Series with the median value of each column.

By specifying the column axis (`axis='columns'`), the `median()` method searches column-wise and returns the median value for each row.

- **The mode method**

The `mode()` method returns the mode value of each column.

By specifying the column axis (`axis='columns'`), the `mode()` method searches column-wise and returns the mode value for each row.

Note: **Mean** - The average value

Median - The mid point value

Mode - The most common value

7. Correlation

Correlation is an indication about the changes between two variables. We can plot correlation matrix to show which variable is having a high or low correlation in respect to another variable

Python script will generate and plot correlation matrix. It can be generated with the help of `corr()` function on Pandas DataFrame and plotted with the help of `pyplot`.



Figure 4.6: Correlation

8. Preparing data for training

- Separating the features and target columns

```
[946] # separating the feature and target columns  
  
x = df.drop('fraud_reported', axis = 1)  
y = df['fraud_reported']
```

Figure 4.7: Separating X and y

- Extracting Categorical Values from X

```
# extracting categorical columns  
cat_df = X.select_dtypes(include = ['object'])
```

Figure 4.8: Extracting Categorical variables

- Encoding Categorical Variables

```
cat_df_num = pd.get_dummies(cat_df)
```

Figure 4.9: Encoding Categorical variables

- **Splitting dataset into Training and Test Set**

```
# splitting data into training set and test set  
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Figure 4.10: Splitting dataset into Training and test set

- **Feature Scaling**

```
# Scaling the numeric values in the dataset  
from sklearn.preprocessing import StandardScaler  
  
scaler = StandardScaler()  
scaled_data = scaler.fit_transform(num_df)
```

Figure 4.11: Feature Scaling

9. Model Validating Tools

- **Accuracy Score**

Accuracy is a metric used in classification problems used to tell the percentage of accurate predictions. We calculate it by dividing the number of correct predictions by the total number of predictions.

$$ACCURACY = \frac{NUMBER OF CORRECT PREDICTIONS}{TOTAL NUMBER OF PREDICTIONS}$$

- **Confusion Matrix**

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 4.12: Confusion matrix

- **Classification Report**

A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model. If you have never used it before to evaluate the performance of your model then this article is for you. It is one of the performance evaluation metrics of a classification-based machine learning model. It displays your model's precision,

recall, F1 score and support. It provides a better understanding of the overall performance of our trained model. To understand the classification report of a machine learning model, you need to know all of the metrics displayed in the report.

- **Precision**

Precision is defined as the ratio of true positives to the sum of true and false positives.

- **Recall**

Recall is defined as the ratio of true positives to the sum of true positives and false negatives.

- **F1 Score**

The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.

- **Support**

Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models, it just diagnoses the performance evaluation process.

- **Macro-average scores**

It is the simple mean of scores of all classes

- **Weighted average scores**

The sum of the scores of all classes after multiplying their respective class proportions.

4.2 Models and Their Predictions

1. Logistic Regression

Logistic Regression is a Linear Classification Model

```
[182] from sklearn.linear_model import LogisticRegression
      logreg=LogisticRegression()
      logreg.fit(X_train,y_train)
      y_pred=logreg.predict(X_test)
```

Figure 4.13: Logistic Regression Model

```
# accuracy_score, confusion_matrix and classification_report
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

log_train_acc = accuracy_score(y_train, logreg.predict(X_train))
log_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Logistic Regression Classifier is : {log_train_acc}")
print(f"Test accuracy of Logistic Regression Classifier is : {log_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Training accuracy of Logistic Regression Classifier is : 0.86125
 Test accuracy of Logistic Regression Classifier is : 0.53
 [[76 81]
 [13 30]]

	precision	recall	f1-score	support
0	0.85	0.48	0.62	157
1	0.27	0.70	0.39	43
accuracy			0.53	200
macro avg	0.56	0.59	0.50	200
weighted avg	0.73	0.53	0.57	200

Figure 4.14: Logistic Regression Model Predictions

As you can see from the above results, the predictions are very bad, that is because the logistic regression is used as a linear Classification Model and our dataset is a non linear dataset and have higher dimension

2. Kernel SVM

SVM is one of the most popular supervised learning algorithm.

SVC Dataset is used for Classification problem. As our dataset is nonlinear we will use the *kerneltrick*. There are different types of kernel trick, but the most used is rbf and that is what we will use.

```
[184] from sklearn.svm import SVC

svc = SVC(kernel = 'rbf', random_state = 0)
svc.fit(X_train, y_train)

y_pred = svc.predict(X_test)
```

Figure 4.15: Support Vector Classification Model

```
[185] # accuracy_score, confusion_matrix and classification_report

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

svc_train_acc = accuracy_score(y_train, svc.predict(X_train))
svc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Support Vector Classifier is : {svc_train_acc}")
print(f"Test accuracy of Support Vector Classifier is : {svc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

Training accuracy of Support Vector Classifier is : 0.745
Test accuracy of Support Vector Classifier is : 0.785
[[157  0]
 [ 43  0]]
      precision    recall  f1-score   support

     0       0.79      1.00      0.88        157
     1       0.00      0.00      0.00         43

   accuracy          0.79
  macro avg          0.39
 weighted avg          0.62
```

Figure 4.16: Support Vector Classification Model Predictions

From the above results, we can see that the predictions are quite good.

3. *K*-Nearest Neighbours

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

here we have used the parameter `n_neighbors`. **`n_neighbors`** parameter is used for *k*-neighbors queries.

for our dataset we have used $n_neighbors = 30$.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
```

Figure 4.17: *K*-Nearest Neighbours Classification Model

```
# accuracy_score, confusion_matrix and classification_report
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

knn_train_acc = accuracy_score(y_train, knn.predict(X_train))
knn_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of KNN is : {knn_train_acc}")
print(f"Test accuracy of KNN is : {knn_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

Training accuracy of KNN is : 0.77625
Test accuracy of KNN is : 0.725
[[127  30]
 [ 25  18]]
      precision    recall  f1-score   support

     0       0.84      0.81      0.82       157
     1       0.38      0.42      0.40        43

 accuracy          0.61
 macro avg         0.61
 weighted avg      0.74
```

Figure 4.18: *K*-Nearest Neighbours Classification Model Predictions

From the above results, we can see that the predictions are as good as SVC.

4. Decision Tree Classification

A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

```

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(max_leaf_nodes = 4)
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)

```

Figure 4.19: Decision Tree Classification Model

```

# accuracy_score, confusion_matrix and classification_report
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

dtc_train_acc = accuracy_score(y_train, dtc.predict(X_train))
dtc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Decision Tree is : {dtc_train_acc}")
print(f"Test accuracy of Decision Tree is : {dtc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

```

Training accuracy of Decision Tree is : 0.86
 Test accuracy of Decision Tree is : 0.83
 [[129 28]
 [6 37]]

	precision	recall	f1-score	support
0	0.96	0.82	0.88	157
1	0.57	0.86	0.69	43
accuracy			0.83	200
macro avg	0.76	0.84	0.78	200
weighted avg	0.87	0.83	0.84	200

Figure 4.20: Decision Tree Classification Model Predictions

From the above results, we can see that the predictions of the Decision Tree Classification Model is the **BEST** out of the Other Classification Models.

5. Random Forest Classification

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

Here we have used the parameters $n_estimators = 100$ (Which is the default setting), $n_estimators$ means the number of trees in the forest and $criterion = 'entropy'$ where $criterion$ means the quality of split and entropy is the supported criteria for the Shannon Information Gain.

```
[208]
from sklearn.ensemble import RandomForestClassifier

rand_clf = RandomForestClassifier(criterion= 'entropy', max_depth= 5, max_features= 'sqrt',
                                min_samples_leaf= 1, min_samples_split= 3, n_estimators= 65)
rand_clf.fit(X_train, y_train)

y_pred = rand_clf.predict(X_test)
```

Figure 4.21: Random Forest Classification Model

```
[209]
# accuracy_score, confusion_matrix and classification_report
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

rand_clf_train_acc = accuracy_score(y_train, rand_clf.predict(X_train))
rand_clf_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of Random Forest is : {rand_clf_train_acc}")
print(f"Test accuracy of Random Forest is : {rand_clf_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

Training accuracy of Random Forest is : 0.81625
Test accuracy of Random Forest is : 0.805
[[153  4]
 [ 35  8]]
      precision    recall  f1-score   support

     0       0.81      0.97      0.89       157
     1       0.67      0.19      0.29        43

   accuracy          0.81          200
  macro avg       0.74      0.58      0.59       200
 weighted avg       0.78      0.81      0.76       200
```


Figure 4.22: Random Forest Classification Model Predictions

From the above results, we can see that the predictions of the Random Forest

Classification Model is the Second **BEST** out of the Other Classification Models

4.3 Comparison of Models

```
[269] models = pd.DataFrame({  
    'Model' : ['SVC', 'KNN', 'Decision Tree', 'Random Forest'],  
    'Score' : [svc_test_acc, knn_test_acc, dtc_test_acc, rand_clf_test_acc]  
})  
  
models.sort_values(by = 'Score', ascending = False)
```



	Model	Score
2	Decision Tree	0.830
3	Random Forest	0.805
0	SVC	0.785
1	KNN	0.725

Figure 4.23: Model Comparison Code

Decision Tree Classification Model with highest predictions has shown that it is the best Classification Model for our dataset.

Bibliography

- [1] <https://www.javatpoint.com>
- [2] <https://www.ravelin.com/insights/machine-learning-for-fraud-detection>
- [3] <https://www.udemy.com/course/machinelearning/learn/lecture/5714406#overview>
- [4] <https://machinelearningmastery.com/basic-data-cleaning-for-machine-learning/>

