

# Report

## Introduction

This study is based on a dataset that contains 10 000 observations of bank customers with 14 variables, 1 dependent variable and 13 independent variables. I predict if a customer is likely to churn or not based on multiple features. I use different Machine learning algorithms and predict the exited feature. It is a binary classification supervised learning problem so I predict that a customer ended their engagement with the bank or a customer did not end their engagement with the bank based on multiple features. I evaluate the performance of machine learning algorithms by using precision, recall and accuracy score and interpret the result in detail that I get from Random Forest, SVM and Decision Tree.

## Description of the data:

The dataset for the study is called 'Bank Customer' and is downloaded from Kaggle.com (2020- 03-23). Originally, the data was collected from an unknown bank with customers in France, Spain and Germany, and the aim of the study was to predict customer churn. The data contains a total of 10 000 observations with 11 variables, 1 dependent variable and 10 independent variables. There are no missing values or wrong imputations in the dataset, meaning that no observations are lost. As previously mentioned, there are 11 variables included in this dataset. The dependent variable is 'Exited'. If the dependent variable takes on a value of 1 then the customer has ended their engagement with the bank. If instead the dependent variable takes on a value of 0 the customer has not ended their engagement with the bank. As previously mentioned, a total of 10 independent variables are available in the dataset. These are: credit score, geography (Spain, France and Germany), gender, age, tenure (how long a customer has had an engagement with the bank), account balance, has credit card, if the individual is an active member or not, and estimated salary per year.

## Import the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.metrics import classification_report, confusion_matrix, precision_score, recall_score, f1_score, accuracy_score
```

## Import the dataset

```
churn = pd.read_csv('Churn_Modelling.csv')
churn.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

## Describing the data

✓ [3] churn.shape

0s

(10000, 14)

✓ [4] churn.size

0s

140000

✓ ▶ churn.info()

0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   RowNumber              10000 non-null  int64  
1   CustomerId             10000 non-null  int64  
2   Surname                 10000 non-null  object  
3   CreditScore             10000 non-null  int64  
4   Geography               10000 non-null  object  
5   Gender                  10000 non-null  object  
6   Age                    10000 non-null  int64  
7   Tenure                  10000 non-null  int64  
8   Balance                 10000 non-null  float64 
9   NumOfProducts          10000 non-null  int64  
10  HasCrCard               10000 non-null  int64  
11  IsActiveMember          10000 non-null  int64  
12  EstimatedSalary         10000 non-null  float64 
13  Exited                  10000 non-null  int64  
```

## Statistics of the data:

✓ [6] churn.describe()

0s

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000



## cleaning the data

There is no missing values in the dataset so no need to handle missing values

```
✓ 0s ▶ churn.isnull().sum()

┌──────────┬──────────┐
│ RowNumber │ 0         │
│ CustomerId │ 0         │
│ Surname    │ 0         │
│ CreditScore │ 0         │
│ Geography  │ 0         │
│ Gender     │ 0         │
│ Age        │ 0         │
│ Tenure     │ 0         │
│ Balance    │ 0         │
│ NumOfProducts │ 0        │
│ HasCrCard  │ 0         │
│ IsActiveMember │ 0        │
│ EstimatedSalary │ 0        │
│ Exited     │ 0         │
└──────────┴──────────┘
dtype: int64
```

## Outliers:

Outlier is an extremely high or extremely low data point relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset you're working with.

Outliers are extreme values that stand out greatly from the overall pattern of values in a dataset or graph

Outliers are an important part of a dataset. They can hold useful information about your data.

Outliers can give helpful insights into the data you're studying, and they can have an effect on statistical results. This can potentially help you discover inconsistencies and detect any errors in your statistical processes.

So, knowing how to find outliers in a dataset will help you better understand your data[1]

## Showing outliers

There are some methods to showing the outliers in dataset

You can choose from several methods to detect outliers depending on your time and resources.

## Sorting method

You can sort quantitative variables from low to high and scan for extremely low or extremely high values. Flag any extreme values that you find.

## Using visualizations

You can use software to visualize your data with a box plot, or a box-and-whisker plot, so you can see the data distribution at a glance. This type of chart highlights minimum and maximum values (the range), the median, and the interquartile range for your data.

Many computer programs highlight an outlier on a chart with an asterisk, and these will lie outside the bounds of the graph.

## Statistical outlier detection

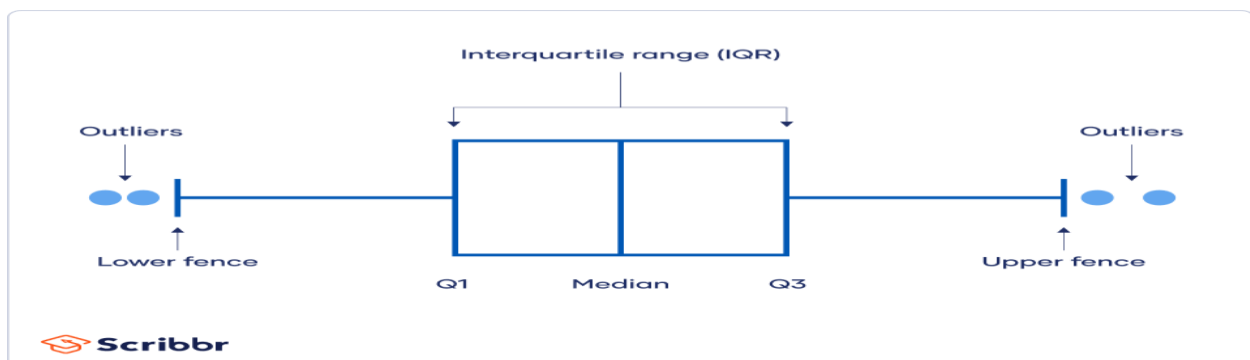
Statistical outlier detection involves applying statistical tests or procedures to identify extreme values.

You can convert extreme data points into z scores that tell you how many standard deviations away they are from the mean.

If a value has a high enough or low enough z score, it can be considered an outlier. As a rule of thumb, values with a z score greater than 3 or less than  $-3$  are often determined to be outliers

## Using the interquartile range

The interquartile range (IQR) tells you the range of the middle half of your dataset. You can use the IQR to create “fences” around your data and then define outliers as any values that fall outside those fences[2]



I write a function of find outliers returns the number of outliers by this function and call this function again and again to get the outliers of different features.

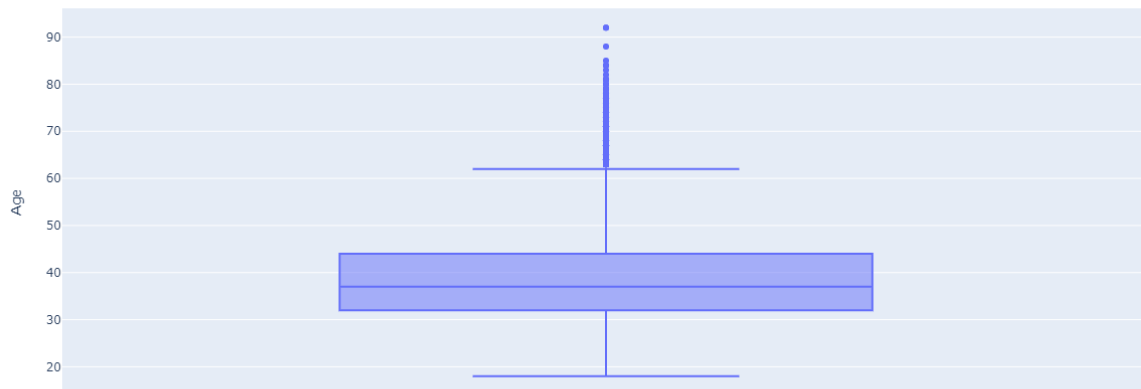
```

def find_outliers_IQR(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers

```

## Outliers of Age Column:

Box plot is use a five number summary that tell us the median , minimum , maximum , q1 and q3 so I clearly see that there is many outliers that is lying very far from median and q3.



```

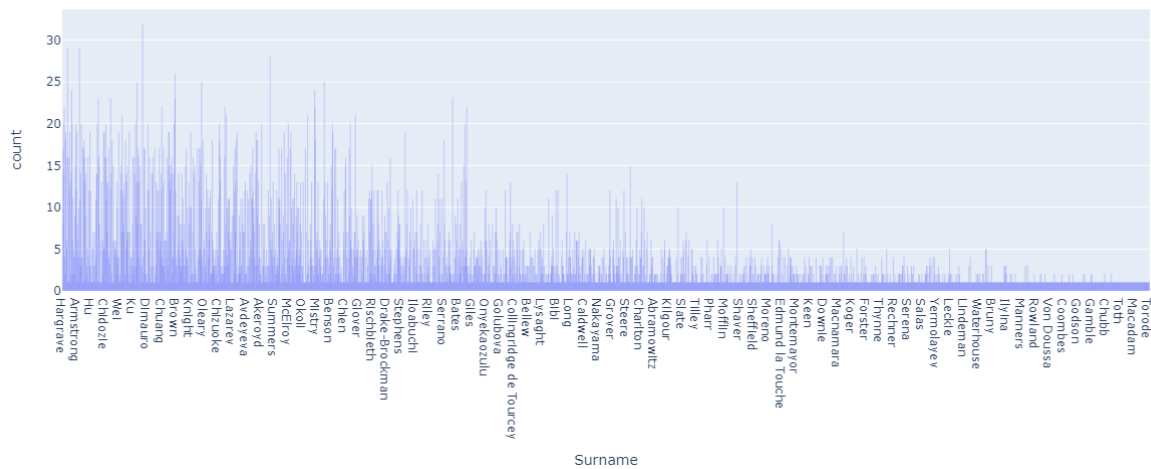
[12] outliers = find_outliers_IQR(churn['Age'])

print('number of outliers: ' + str(len(outliers)))
print('max outlier value: ' + str(outliers.max()))
print('min outlier value: ' + str(outliers.min()))

number of outliers: 359
max outlier value: 92
min outlier value: 63

```

## Outliers of Sur-name columns:



## Total Outliers in credit score columns:

```
[13] outliers = find_outliers_IQR(churn['CreditScore'])

print('number of outliers: ' + str(len(outliers)))

print('max outlier value: ' + str(outliers.max()))

print('min outlier value: ' + str(outliers.min()))

number of outliers: 15
max outlier value: 382
min outlier value: 350
```

## Total Outliers in Number of products columns:

```
▶ outliers = find_outliers_IQR(churn['NumOfProducts'])

print('number of outliers: ' + str(len(outliers)))

print('max outlier value: ' + str(outliers.max()))

print('min outlier value: ' + str(outliers.min()))

📄 number of outliers: 60
max outlier value: 4
min outlier value: 4
```

## Handle the outliers

```
✓ 3s ▶ def find_np_outliers_IQR(df):
```

```
    q1=np.quantile(df,0.25)

    q3=np.quantile(df,0.75)

    IQR=q3-q1

    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]

    return outliers
```

```
✓ 3s [59] outliers = find_np_outliers_IQR(b)
```

```
    print('number of outliers in Age column: ' + str(len(outliers)))
```

```
number of outliers in Age column: 0
```

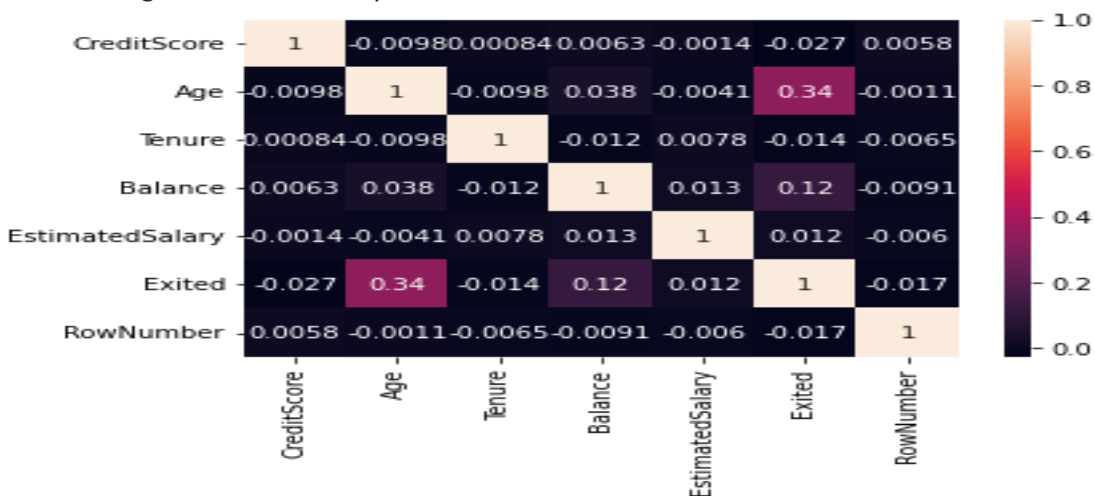
```
▶ outliers = find_np_outliers_IQR(prod)
```

```
print('number of outliers in Number of product : ' + str(len(outliers)))
```

```
number of outliers: 0
```

## Feature Selection:

Correlation states how the features are related to each other or the target variable(Exited). Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable) Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.



You see that how all feature correlated with exited column so I drop some unwanted columns that's make no impact target feature so these features are not important.

```
✓ [30] churn.drop(['Surname', 'RowNumber'], axis=1, inplace=True)
```

## Dependent and independent features

Dependent variables are nothing but the variable which holds the phenomena which we are studying. Independent variables are the ones which through we are trying to explain the value or effect of the output variable (dependent variable) by creating a relationship between an independent and dependent variable.

**In mathematics, this is generally explained with a formula**

**$y=f(x)$                       Where,**

**$x=$  independent variables**

**$y=$  dependent variable**

This means any changes in  $x$  will cause a change in the value of  $y$ . The change can be negative or positive.  
[3]

In our case dependent features is exited and independent features are CreditScore, Age, Tenure, Balance, EstimatedSalary, Exited, RowNumber.

```
✓ [31] x=churn.drop('Exited', axis=1)
```

```
✓ [32] y=churn['Exited']
```

## Handle categorical features

All Machine Learning models are some kind of mathematical model that needs numbers to work with. Categorical data have possible values (categories) and it can be in text form. For example, Gender: Male/Female/Others, Ranks: 1st/2nd/3rd, etc. While working on a data science project after handling the missing value of datasets. The next work is to handle categorical data in datasets before applying any ML models.

First, let's understand the types of categorical data:

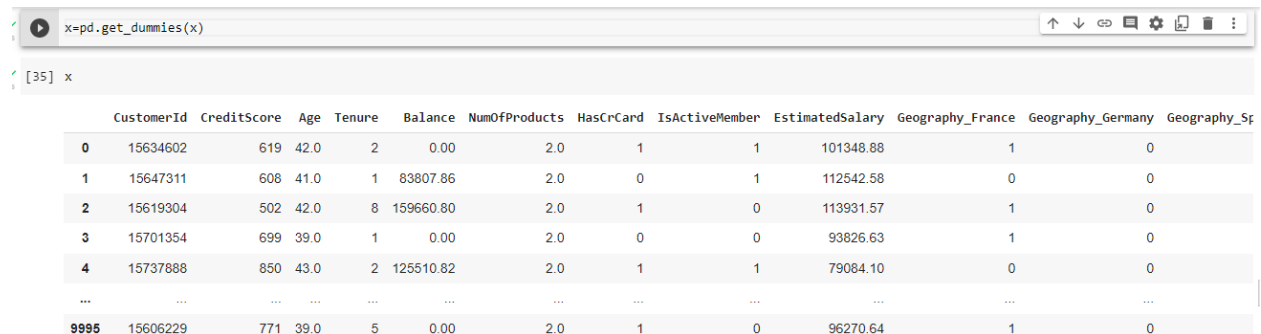


1. **Nominal Data:** The nominal data called *labelled/named* data. Allowed to change the order of categories, change in order doesn't affect its value. For example, Gender (Male/Female/Other), Age Groups (Young/Adult/Old), etc.
2. **Ordinal Data:** Represent *discretely and ordered units*. Same as nominal data but have ordered/rank. Not allowed to change the order of categories. For example, Ranks: 1st/2nd/3rd, Education: (High School/Undergrads/Postgrads/Doctorate), etc.

Create dummies or binary type columns for each category in the object/ category type feature. The value for each row is 1 if that category is available in that row else 0. To create dummies use pandas `get_dummies()` function

**Advantage:** Easy to use and fast way to handle categorical column values.

**Disadvantage:** `get_dummies` method is not useful when data have many categorical columns. If the category column has many categories leads to add many features into the dataset. Hence, This method is only useful when data having less categorical columns with fewer categories.[4]



```
x=pd.get_dummies(x)
```

[35] x

	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germany	Geography_Sp
0	15634602	619	42.0	2	0.00	2.0	1	1	101348.88	1	0	
1	15647311	608	41.0	1	83807.86	2.0	0	1	112542.58	0	0	
2	15619304	502	42.0	8	159660.80	2.0	1	0	113931.57	1	0	
3	15701354	699	39.0	1	0.00	2.0	0	0	93826.63	1	0	
4	15737888	850	43.0	2	125510.82	2.0	1	1	79084.10	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	15606229	771	39.0	5	0.00	2.0	1	0	96270.64	1	0	

## Train Test split the dataset

Scikit-learn alias **sklearn** is the most useful and robust library for machine learning in Python. The **scikit-learn library** provides us with the `model_selection` module in which we have the splitter function `train_test_split()`.

The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results. By default, the Test set is split into 30 % of actual data and the training set is split into 70% of the actual data.

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

## Parameters:

1. \*arrays: inputs such as lists, arrays, data frames, or matrices
2. test\_size: this is a float value whose value ranges between 0.0 and 1.0. it represents the proportion of our test size. its default value is none.
3. train\_size: this is a float value whose value ranges between 0.0 and 1.0. it represents the proportion of our train size. its default value is none.
4. random\_state: this parameter is used to control the shuffling applied to the data before applying the split. it acts as a seed.
5. shuffle: This parameter is used to shuffle the data before splitting. Its default value is true.
6. stratify: This parameter is used to split the data in a stratified fashion [5]

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
print("the shape of x_train",x_train.shape)
print("the shape of y_train",y_train.shape)
print("the shape of x_test",x_test.shape)
print("the shape of y_test",y_test.shape)
```

the shape of x\_train (7500, 14)  
the shape of y\_train (7500,)  
the shape of x\_test (2500, 14)  
the shape of y\_test (2500,)

## Feature Scaling:

**Feature scaling** is one of the most important data preprocessing step in machine learning. Algorithms that compute the distance between the features are biased towards numerically larger values if the data is not scaled.

Tree-based algorithms are fairly insensitive to the scale of the features. Also, feature scaling helps machine learning, and deep learning algorithms train and converge faster.

There are some feature scaling techniques such as Normalization and Standardization that are the most popular and at the same time, the most confusing ones.

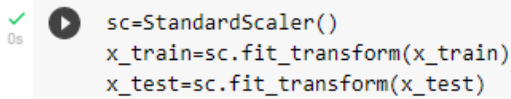
**Standardization** is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score.

$$X_{\text{new}} = (X - \text{mean})/\text{Std}$$

Standardization can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Geometrically speaking, it translates the data to the mean vector of original data to the origin and squishes or expands the points if std is 1 respectively. We can see that we are just changing mean and standard deviation to a standard normal distribution which is still normal thus the shape of the distribution is not affected.

Standardization does not get affected by outliers because there is no predefined range of transformed features [6]

In python we use the library of scikit learn and do standardization, so for standardization I import the standard scaler and fit\_transform function transform the data. First transform the x\_train and then transform the x\_test and then fit in the model in the next step.

A code snippet in a light gray box with a green checkmark icon on the left. The code is as follows:

```
sc=StandardScaler()  
x_train=sc.fit_transform(x_train)  
x_test=sc.fit_transform(x_test)
```

## Machine learning

Machine learning is a field of computer science that uses statistical techniques to give computer programs the ability to learn from past experiences and improve how they perform specific tasks.

### Supervised Learning:

Supervised machine learning requires labelled input and output data during the training phase of the machine learning lifecycle. This training data is often labelled by a data scientist in the preparation phase, before being used to train and test the model. Once the model has learned the relationship between the input and output data, it can be used to classify new and unseen datasets and predict outcomes.

The reason it is called supervised machine learning is because at least part of this approach requires human oversight. The vast majority of available data is unlabelled, raw data. Human interaction is generally required to accurately label data ready for supervised learning. Naturally, this can be a resource intensive process, as large arrays of accurately labelled training data is needed.

Supervised machine learning is used to classify unseen data into established categories and forecast trends and future change as a predictive model. A model developed through supervised machine learning will learn to recognise objects and the features that classify them. Predictive models are also often trained with supervised machine learning techniques. By learning patterns between input and output data, supervised machine learning models can predict outcomes from new and unseen data. This could be in forecasting changes in house prices or customer purchase trends.

### Supervised machine learning is often used for:

Classifying different file types such as images, documents, or written words.

Forecasting future trends and outcomes through learning patterns in training data.

## **Unsupervised learning:**

Unsupervised machine learning is the training of models on raw and unlabelled training data. It is often used to identify patterns and trends in raw datasets, or to cluster similar data into a specific number of groups. It's also often an approach used in the early exploratory phase to better understand the datasets.

As the name suggests, unsupervised machine learning is more of a hands-off approach compared to supervised machine learning. A human will set model hyperparameters such as the number of cluster points, but the model will process huge arrays of data effectively and without human oversight. Unsupervised machine learning is therefore suited to answer questions about unseen trends and relationships within data itself. But because of less human oversight, extra consideration should be made for the explainability of unsupervised machine learning.

The vast majority of available data is unlabelled, raw data. By grouping data along similar features or analysing datasets for underlying patterns, unsupervised learning is a powerful tool used to gain insight from this data. In contrast, supervised machine learning can be resource intensive because of the need for labelled data.

## **Unsupervised machine learning is mainly used to:**

Cluster datasets on similarities between features or segment data

Understand relationship between different data point such as automated music recommendations

Perform initial data analysis [7]

## **Evaluation Metrics**

Classification is about predicting the class labels given input data. In binary classification, there are only two possible output classes (i.e., Dichotomy). In multiclass classification, more than two possible classes can be present. I'll focus only on binary classification.

An example of binary classification is Bank churn prediction, where the input data could include the row number, Age, surname and number of products of customer and the output label is either 1 and 0.

## Accuracy

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

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

When any model gives an accuracy rate of 99%, you might think that model is performing very good but this is not always true and can be misleading in some situations.

**Precision:** Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives. Precision for a label is defined as the number of true positives divided by the number of predicted positives.

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

## Recall

**Recall (Sensitivity)** — Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive

Recall for a label is defined as the number of true positives divided by the total number of actual positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

### Confusion Matrix:

Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.

A confusion matrix is defined as the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known .

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

### F1 Score

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

F1 Score is the harmonic mean of precision and recall.

$$F1 = 2. \frac{Precision \times Recall}{Precision + Recall}$$

The F1 score punishes extreme values more. F1 Score could be an effective evaluation metric in the following cases:

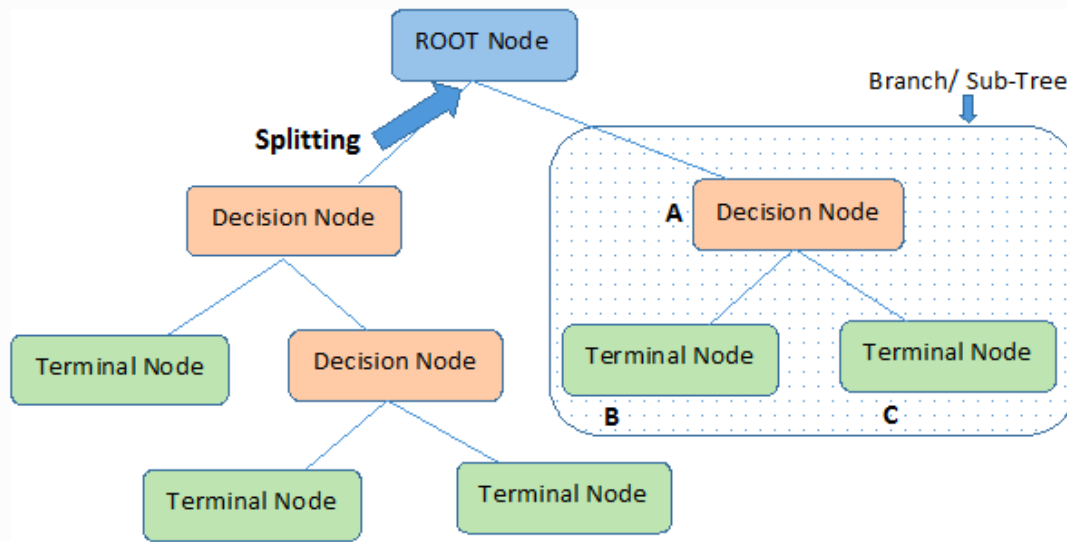
- When FP and FN are equally costly.
- Adding more data doesn't effectively change the outcome
- True Negative is high [8]

## Decision Tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving **regression and classification problems** too. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).

## Important Terminology related to Decision Tree

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.



**Note:-** A is parent node of B and C.

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.[9]

In below Screenshot of decision tree I train the model on training dataset and predict the result using unseen dataset using predict function. After that I plot the tree that is generated according to our dataset.

#### Decision Tree Classifier

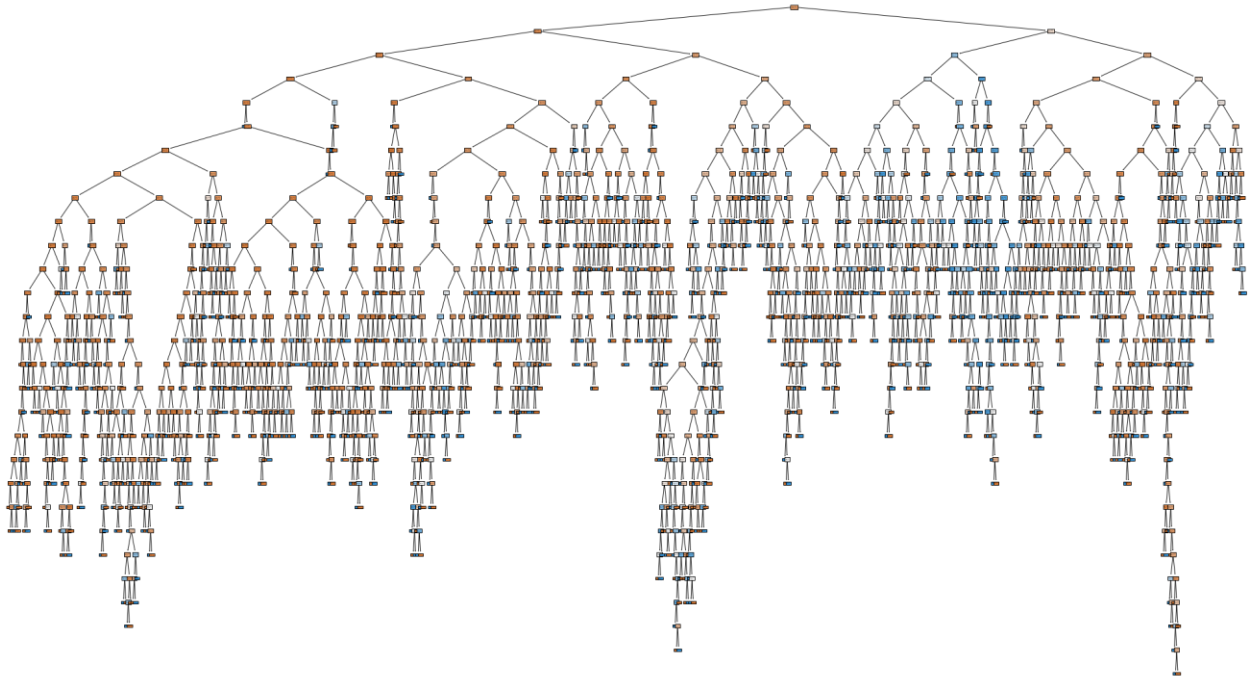
```
[38] # fit the model
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
```

```
DecisionTreeClassifier()
```

#### Prediction

```
[39] # make prediction
pred_1 = model.predict(x_test)
```





## Interpret the result of Decision Tree

Here I verify that there is any overfitting or not so for this I print the training score and testing score and conclude that there is no overfitting here and accuracy of decision tree is 73%. We already know the formula of accuracy score that is discussed earlier.

### Training Score and testing score

```
print("Training set score: {:.4f}".format(model.score(x_train,y_train)))
print("Validation set score: {:.4f}".format(model.score(x_test,y_test)))
```

```
Training set score: 1.0000
Validation set score: 0.7376
```

### Accuracy

```
[42] # calculate accuracy score
accuracy_score(y_test,pred_1)

0.7376
```

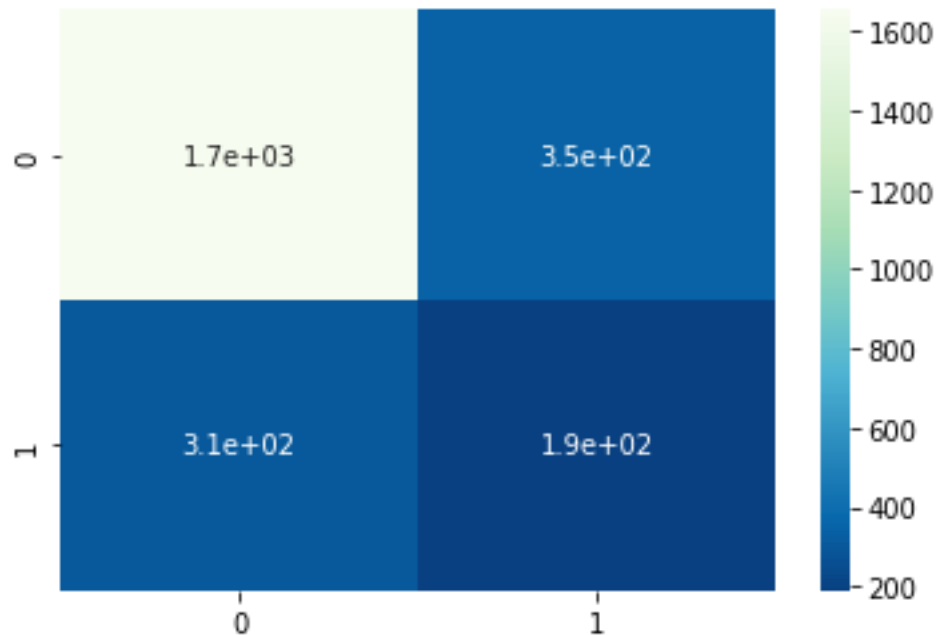
## Evaluation Metrics of Decision Tree:

We already discussed the formula of precision, recall and f1 score so precision is 35% , recall is 38% and f1 score is 36%

```
print("the precision score is : ",precision_score(y_test,pred_1))
print("the recall score is :",recall_score(y_test,pred_1))
print("the f1 measure is ",f1_score(y_test,pred_1))
```

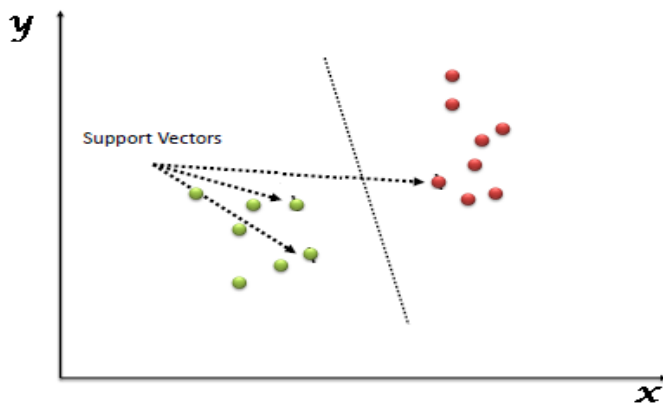
```
the precision score is : 0.3525046382189239
the recall score is : 0.3822937625754527
the f1 measure is 0.3667953667953668
```

**Confusion Matrix:**



## SVM

“Support Vector Machine” (SVM) is a supervised algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line) [10]

Performance of SVM is much more stable than decision tree the accuracy of SVM is 83%. Which is very good and results of other evaluation metrics is also very stable. So precision score is 79% and recall is low it is around 23% and f1 score is 36%

#### SVM

```
✓ [45] model_1 = svm.SVC()  
0s model_1.fit(x_train, y_train)  
  
SVC()
```

#### Prediction

```
✓ [46] # make prediction  
0s pred_2 = model_1.predict(x_test)
```

### Interpret the results of SVM

Accuracy of SVM is very good and results of other evaluation metrics is also very stable. So precision score is 79% and recall is low it is around 23% and f1 score is 36%

#### Accuracy score

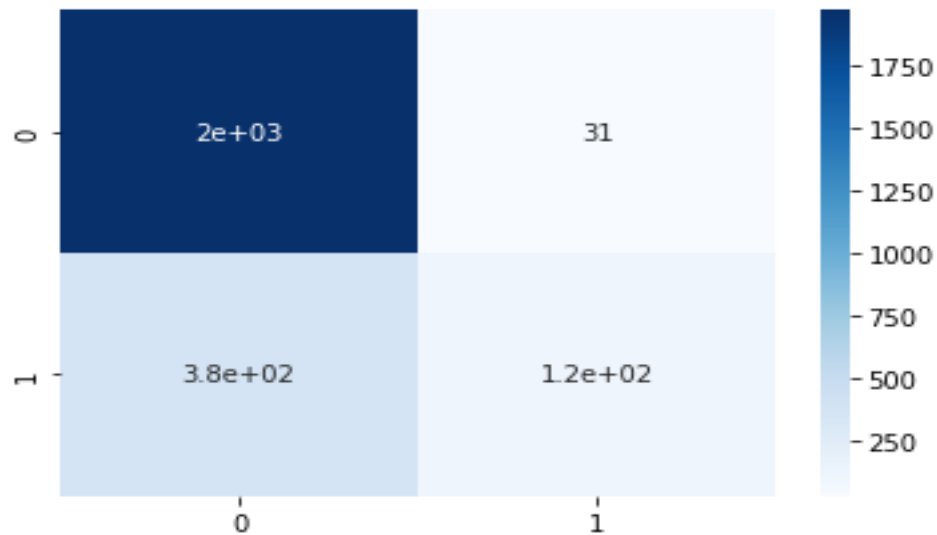
```
✓ [48] # calculate accuracy score  
0s accuracy_score(y_test, pred_2)  
  
0.836
```

#### Evaluation Metrics

```
✓ [49] print("the precision score is : ",precision_score(y_test, pred_2))  
0s print("the recall score is :",recall_score(y_test, pred_2))  
print("the f1 measure is ",f1_score(y_test, pred_2))  
  
the precision score is : 0.7919463087248322  
the recall score is : 0.23742454728370221  
the f1 measure is 0.3653250773993808
```

### Confusion Matrix:

As we already know confusion matrix is used to compare the predicted and actual results.

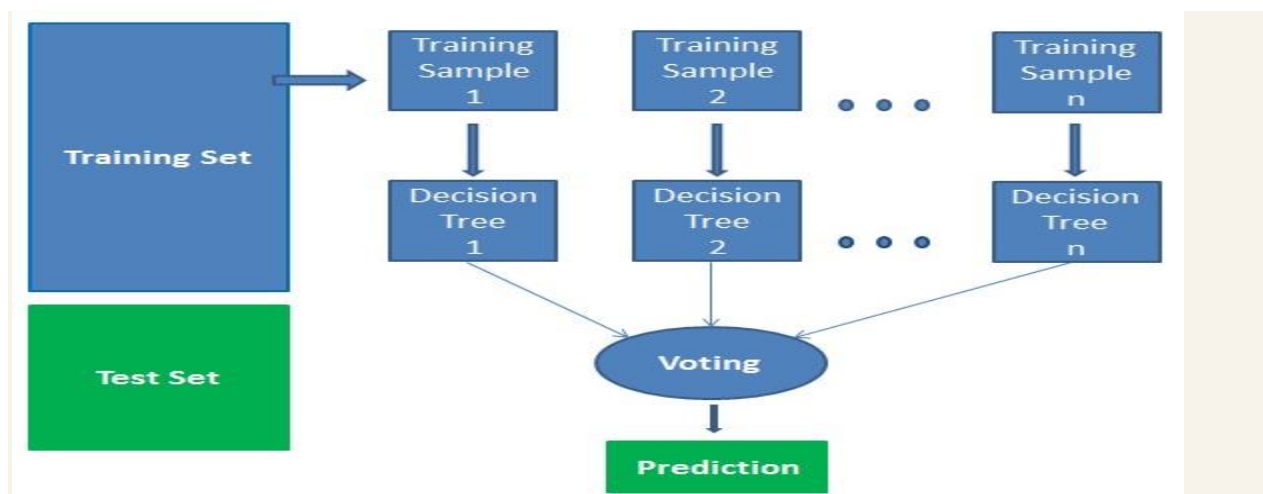


## Random Forest

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests create decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.

Select the prediction result with the most votes as the final prediction.[11]



### Random Forest

```
✓ [51] model_2=RandomForestClassifier()  
0s model_2.fit(x_train, y_train)  
  
RandomForestClassifier()
```

### Prediction

```
✓ [52] # make prediction  
0s pred_3 = model_2.predict(x_test)
```

## Interpret the result of Random Feature

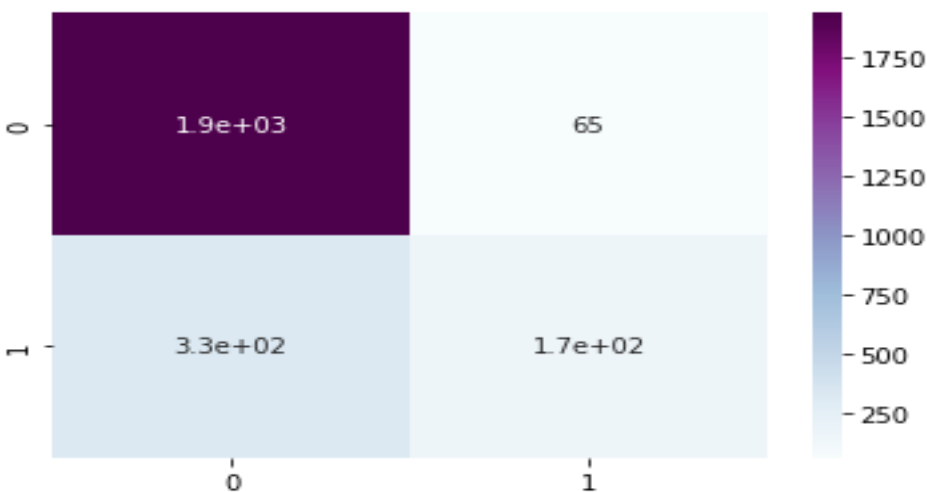
The performance of Random Forest is very good. It is better than SVM and much better than decision tree. The accuracy of RF is more than SVM and Decision tree. The evaluation metrics of Random Forest is also more than SVM and decision tree.

### Accuracy\_Score

```
✓ [54] # calculate accuracy score  
0s accuracy_score(y_test, pred_3)  
  
0.8416
```

### Model evaluation

```
✓ [55] print("the precision score is : ",precision_score(y_test, pred_3))  
0s print("the recall score is :",recall_score(y_test, pred_3))  
print("the f1 measure is ",f1_score(y_test, pred_3))  
  
the precision score is : 0.7186147186147186  
the recall score is : 0.33400402414486924  
the f1 measure is 0.45604395604395603
```



## Best Model:

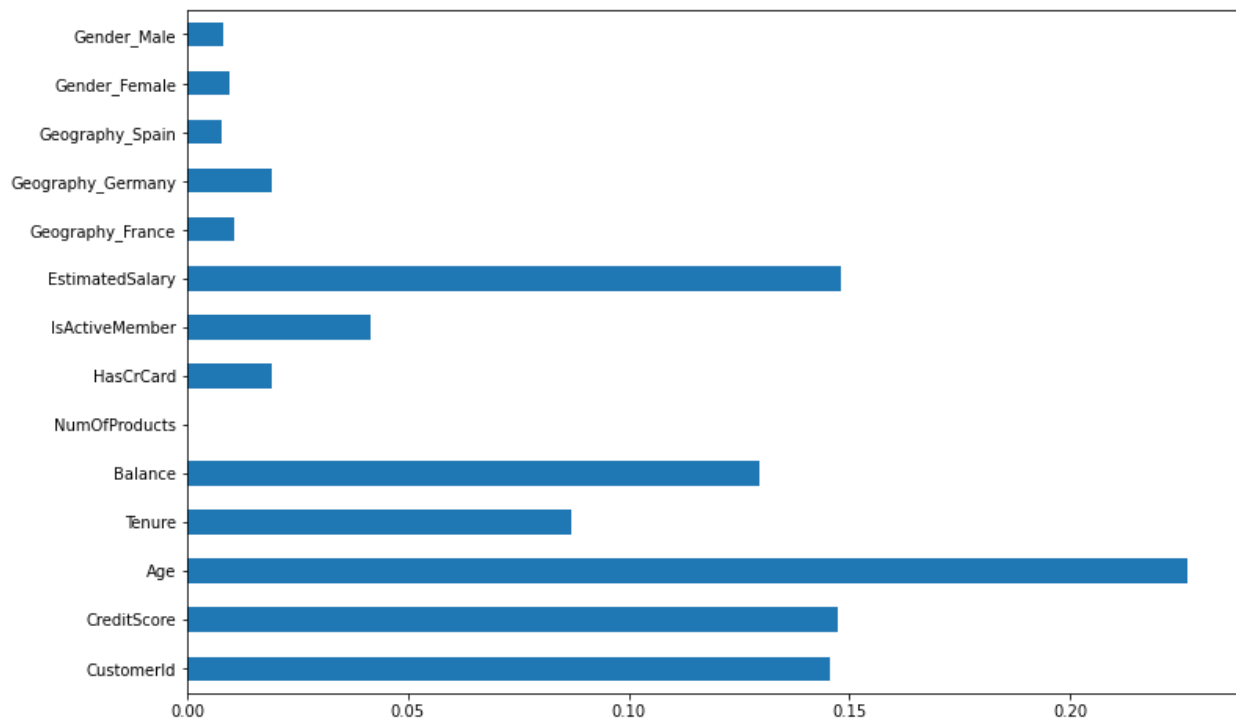
I select random forest is the best model. Because result of RF is much stable then decision tree, SVM and random forest. The precision of random forest is 71% and accuracy is 84%.

	Decision_tree_results	SVM_results	Random Forest results
metricss			
Accuracy	75%	83%	84%
Precision	36%	79%	71%
recall	38%	23%	32%
F1_Measure	37%	36%	44%

## Important features for this Model:

I plot the feature that plays a important role in predicting the customer that Is likely to churn or not. So age and salary is play a huge role because if person is satisfied with salary and still young then its very rare chance that this person is like to churn.

Balance, credit score also make impact in prediction but gender and geography not play any role so its better to delete this because accuracy of model is not affected by these columns



## Reference:

[1] <https://www.freecodecamp.org/news/what-is-an-outlier-definition-and-how-to-find-outliers-in-statistics/>

[2] <https://www.scribbr.com/statistics/outliers/>

[3] <https://sanjaysw05.medium.com/types-of-variables-in-machine-learning-dea91dffdab7>

[4] <https://towardsdatascience.com/ways-to-handle-categorical-data-before-train-ml-models-with-implementation-ffc213dc84ec>

[5] <https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/#:~:text=The%20train%20test%20split%20is,model%20results%20to%20machine%20results.>

[6] <https://www.geeksforgeeks.org/normalization-vs-standardization/>

7] <https://www.seldon.io/supervised-vs-unsupervised-learning-explained>

[8] <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>

[9] <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>

[10] <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

[11] <https://www.datacamp.com/tutorial/random-forests-classifier-python>

## Appendix:

### Data Analysis and visualization:

