Capstone Project Final Report

1. Summary of problem statement, data and findings Every good abstract describes succinctly what was intended at the outset, and summarises findings and implications.

Customer churn represents a basic problem within the competitive atmosphere of the banking industry. Given the importance of customers as the most valuable assets of organizations, customer retention seems to be an essential, basic requirement for any organization. Banks are no exception to this rule. The competitive atmosphere within which banking services are provided by different banks increases the necessity of customer retention. According to surveys, a bank can increase its profits by up to 85 % by improving the retention rate by up to 5 %. In addition, customer retention is seen as more important than in the past.

Project Outcome Across industries, data has become an extremely valuable resource. This is especially true in the financial services sector, where big data has opened up new opportunities, delivering benefits to customers and employees alike. Understanding how banking and big data work in practice requires familiarity with the technologies used to collect, clean and analyze the data sets of information gathered from a variety of channels.

The banking market and consumers who utilize finance products generate an enormous amount of data on a daily basis. Analytics and modelling has changed the way this information is processed, making it possible to identify trends and patterns which can then be used to inform business decisions at scale. While one piece of data is a single data point, multiple pieces of information can create a larger picture that can be used to recognize patterns in customer behaviour, purchasing choices and other key insights. Predictive analytics allows for faster movement and long-term planning to decide what types of products to offer customers and when to offer them. AI, specifically, helps drive this proactive strategy, prevent banking customer churn and promote best actions when it comes to retail.

The key to solve the above limitations lies in extracting early warning signs from the already existing data. Advanced machine learning (ML) and data science (DS) techniques can learn from past customer behavior and external triggers that led to churn and use this learning to predict the future occurrence of a churn -like event.

Using predictive analytics, companies can effectively manage their risks, especially since it can monitor so many diverse forms of data sets at once, be it raw or structured. So it can assess the potential risks involved in any field, be it marketing or be it workforce-related. Most importantly, it can be used to detect the root of past mistakes or bad fiscal phases, and to determine ways to fix loopholes.

With analytics, certain calculated parameters can be put in place to assess high risk decisions better, since there's always an element of unpredictability when it comes to the market.

Customer churn is a common problem across businesses in many sectors. If you want to grow as a company, you have to invest in acquiring new clients. Every time a client leaves, it represents a significant investment lost. Both time and effort need to be channelled into replacing them.

Being able to predict when a client is likely to leave, and offer them incentives to stay, can offer huge savings to a business. As a result, understanding what keeps customers engaged is extremely valuable knowledge, as it can help you to develop your retention strategies, and to roll out operational practices aimed at keeping customers from walking out the door.

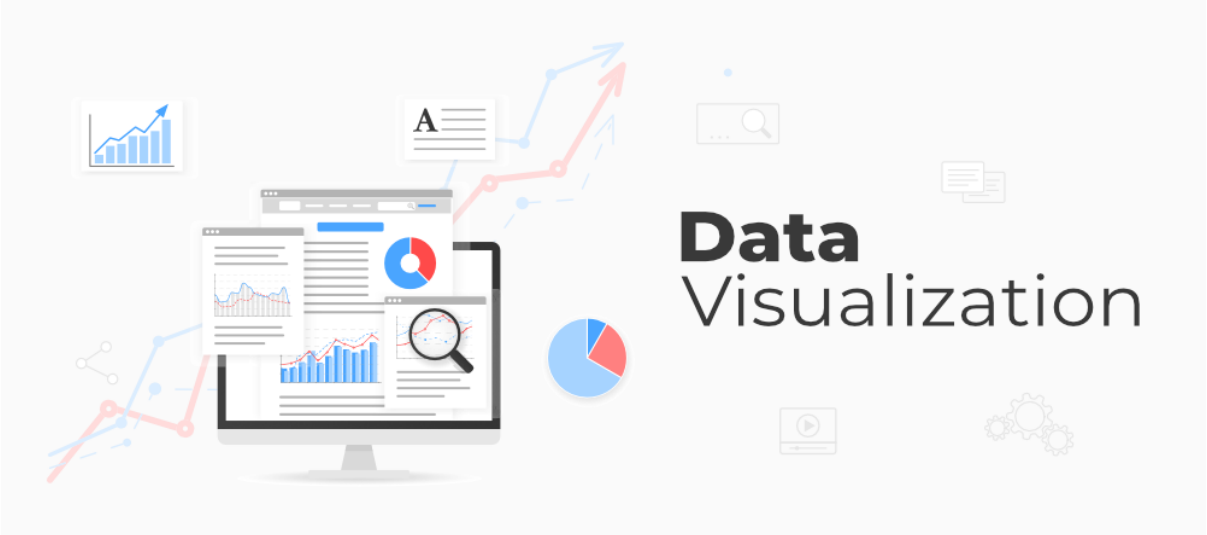
2. Overview of the final process Briefly describe your problem solving methodology. Include information about the salient features of your data, data pre-processing steps, the algorithms you used, and how you combined techniques.

Data Understanding

Real-world data is in most cases incomplete, noisy, and inconsistent.With the exponentially growing data generation and the increasing number of heterogeneous data sources, the probability of gathering anomalous or incorrect data is quite high.

But only high-quality data can lead to accurate models and, ultimately, accurate predictions. Hence, it's crucial to process data for the best possible quality. In order to achieve that we must start with reading and understanding the data and apply the essential data processing steps to build a good and accurate machine learning model. Each column and the related data were thoroughly analysed in order to be familiarised with the domain knowledge.

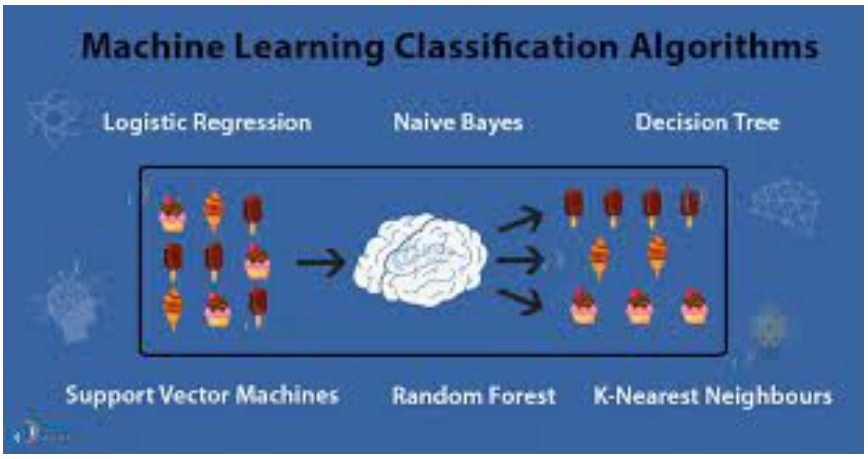
After analysing the data, data visualization was carried out. Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets. The term is often used interchangeably with others, including information graphics, information visualization and statistical graphics.



Data visualization is one of the steps of the data science process, which states that after data has been collected, processed and modelled, it must be visualized for conclusions to be made. Data visualization is also an element of the broader data presentation architecture (DPA) discipline, which aims to identify, locate, manipulate, format and deliver data in the most efficient way possible.

After the data was visualised and interpreted, the data preprocessing steps began. We started with checking basic defects in the data like presence of null values, presence of outliers,checking if the datatype of all the columns are appropriate as per the data description,scaling the data and checking for the presence of multicollinearity.

Based on the results, the corresponding treatments were done as per the current industry standards and the data was ready for building a machine learning model.

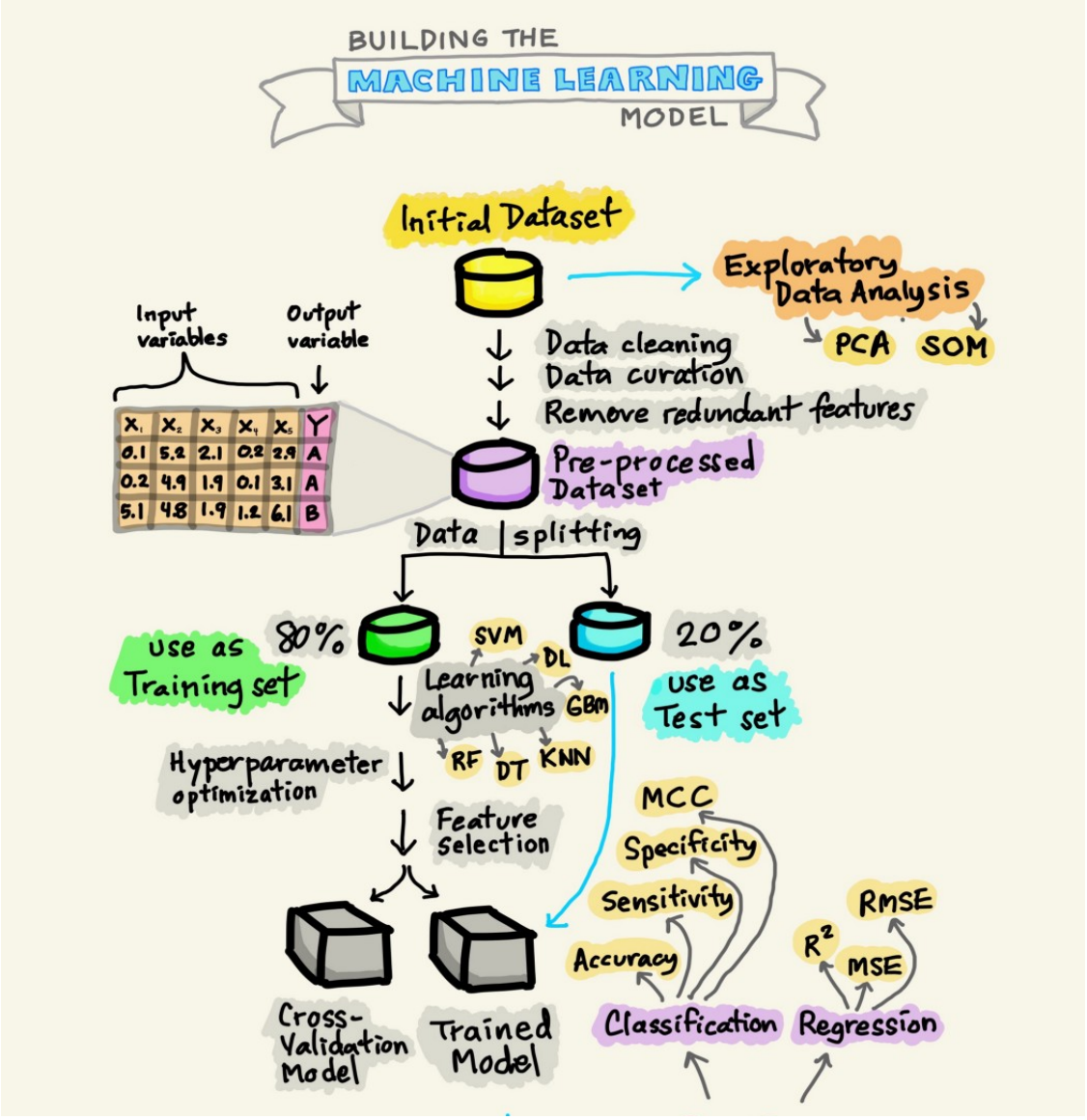


We decided to use three classifications algorithms namely Decision Tree classification, Logistic Regression and Decision Tree Classification.

Our goal was to build a successful model using all three algorithms with optimal accuracy and absence of overfitting/underfitting.

Once all three models are built, we would use different metrics like AUC-ROC score, Cohen-Kappa score,precision, recall and F1 score to select the final model that can be deployed for further predictions.

3. Step-by-step walk through of the solution Describe the steps you took to solve the problem. What did you find at each stage, and how did it inform the next steps? Build up to the final solution.



Data Understanding

Reading the given data and understanding the data is the first and integral part of the entire machine learning process. It is very important for a data scientist to read the data, understand its nuances and do an in depth analysis on the domain to get more context on how each variable contributes to the predictions that the business is looking for.

Interrelated to each other, yet clearly distinguishable, three aspects of Domain Knowledge, a Data Scientist should keep in mind, can be defined in context to the —

1. The source problem, the business is trying to resolve and/or capitalize on.
2. The set of specialized information or expertise held by the business.
3. The exact know-how, for domain specific data collection mechanisms.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df=pd.read_csv('Churn_Modelling.csv')
df.head()
```

Out[1]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedS
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	1013
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	1125
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	1139
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	938
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	790

In [2]:
```
df.shape
```

Out[2]:
```
(10000, 14)
```

In [3]:
```
df['HasCrCard'] = df['HasCrCard'].astype('object')
df['IsActiveMember'] = df['IsActiveMember'].astype('object')
df['Exited'] = df['Exited'].astype('object')
```

In [4]:
```
df_cat = df.select_dtypes('O')
df_cat.columns
```

Out[4]:
```
Index(['Surname', 'Geography', 'Gender', 'HasCrCard', 'IsActiveMember',
       'Exited'],
      dtype='object')
```

In [5]:
```
df_num = df.select_dtypes(np.number)
df_num.columns
```

Out[5]:
```
Index(['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'EstimatedSalary'],
      dtype='object')
```

In [6]:
```
df_num.describe()
```

Out[6]:

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

In [7]:
```
df_cat.describe()
```

Out[7]:

	Surname	Geography	Gender	HasCrCard	IsActiveMember	Exited
count	10000	10000	10000	10000	10000	10000
unique	2932	3	2	2	2	2
top	Smith	France	Male	1	1	0
freq	32	5014	5457	7055	5151	7963

Data Visualization

Data visualization is the graphical representation of information and data in a pictorial or graphical format(Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analyzing massive amounts of information and making data-driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are Charts, Tables, Graphs, Maps, Dashboards.

We made use of Seaborn and matplotlib libraries to provide insightful charts and graphs for interpreting the data visually.

We first visualized the target variable to understand how balanced/imbalanced our target variable is. A pie chart is one of the most common and easy ways to understand how a certain data is split and hence we made use of pie plots from matplotlib for the same.

It is also important to understand how well the data is distributed. For continuous columns, we made use of Histograms to understand the distribution

For Categorical variables, we made use of count plots from the seaborn library to understand how the values are split among themselves in the given data.

Finally, we checked how each independent variable in the data is associated with the target variable that we are looking to predict. We achieved this using box plots and Pivot tables.

Now that we have performed the necessary steps to understand the data in a visual manner, we will be going forward with preprocessing the data.

Data Preprocessing

Before you're ready to feed a dataset into your machine learning model of choice, it's important to do some preprocessing so the data behaves nicely for our model. In an ideal world, you'll have a perfectly clean dataset with no errors or missing values present. In the real world, however, such datasets are rare and uncommon. Data preprocessing is required for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. The steps involved in data preprocessing are explained below.

Checking for null values

The simplest way to deal with missing data is to remove all rows that have a missing value but valuable information can be lost or you could introduce bias. Consequently, it is important to try to understand if there is a reason or pattern for the missing values.

However, in our dataset fortunately, there are no missing values and hence we did not remove any rows or perform any kind of imputation in our data.

```
In [8]: df.isnull().sum()

Out[8]: 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

In [9]: df.isnull().sum()/len(df)*100

Out[9]: RowNumber          0.0
        CustomerId         0.0
        Surname            0.0
        CreditScore        0.0
        Geography          0.0
        Gender             0.0
        Age                0.0
        Tenure             0.0
        Balance            0.0
        NumOfProducts      0.0
        HasCrCard          0.0
        IsActiveMember     0.0
        EstimatedSalary    0.0
        Exited             0.0
        dtype: float64
```

Checking the datatypes of each independent variable

Based on the given data description, we would need to check whether each and every independent variable in the data is associated with the right datatype. If not, we would need to perform appropriate conversion wherever necessary.

In our dataset, HarCrCard, IsActiveMember and the target variable Exited are encoded values of a categorical data.Hence, we need to convert them to object data type to continue with preprocessing the data.

In [3]:
```
df['HasCrCard'] = df['HasCrCard'].astype('object')
df['IsActiveMember'] = df['IsActiveMember'].astype('object')
df['Exited'] = df['Exited'].astype('object')
```

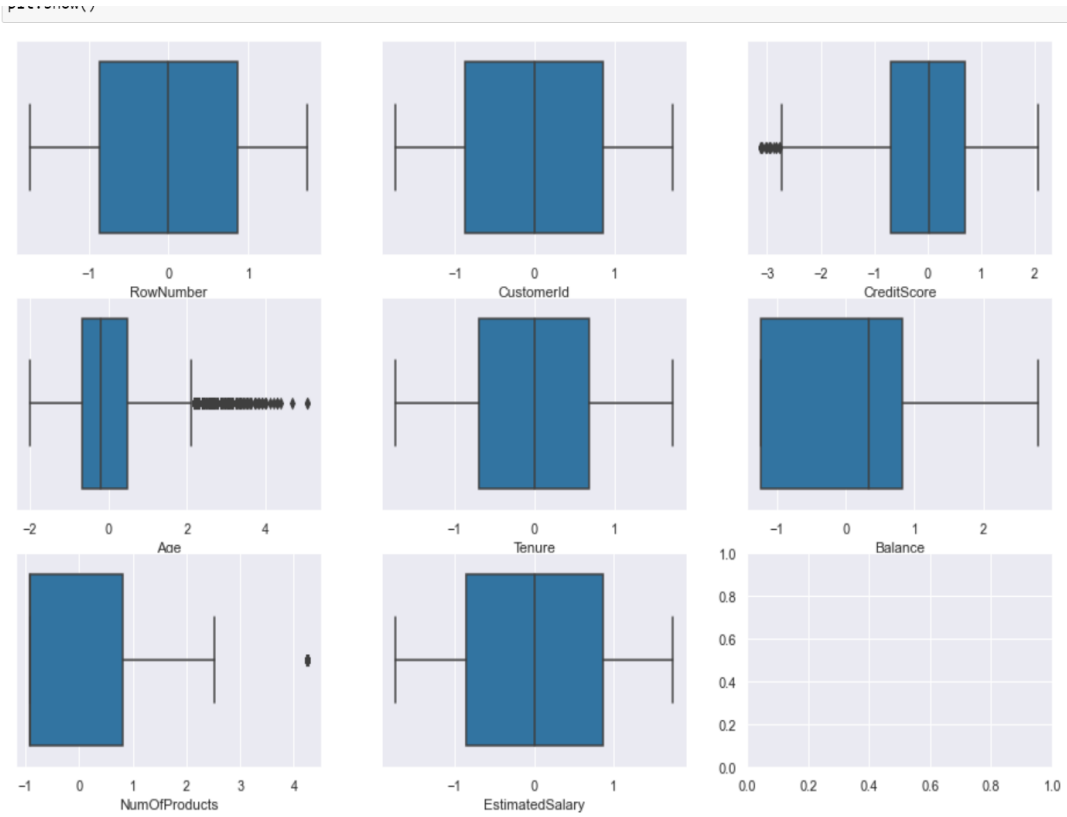
Checking for outliers

An outlier is a data point that is significantly different from other observations. Once you identify outliers, you should also investigate what may have caused them.

Outliers could indicate bad data: data that was incorrectly collected. If this is the case, you may wish to remove these data points or replace them (similar to how you impute values for missing data). Alternatively, these values could be interesting and useful for your machine learning model.

Some machine learning algorithms, such as linear regression, can be sensitive to outliers. Consequently, you may wish to only use algorithms more robust to outliers, such as random forest or gradient boosted trees.

In our data, we made use of boxplots from the seaborn library to understand if there are outliers in the continuous data.



From the boxplot, we could see that outliers are present in few independent variables. Hence, we had used the IQR method to eliminate the outliers from the data.

In [20]:
```
q1 = df_num.quantile(0.25)
q3 = df_num.quantile(0.75)

iqr = q3 - q1

df = df[~((df < (q1 - 1.5 * iqr))| (df > (q3 + 1.5 * iqr))).any(axis = 1)]
df.shape
```

Out[20]: (9515, 14)

After performing the IQR method, we could see that the number of rows decreased as a result of removing the outliers in the data. It is a small price to pay for building an accurate prediction model.

Checking for imbalance

A dataset is unbalanced if each class does not have a similar number of examples. This is common with classification problems such as fraud detection; the majority of transactions are normal, whilst a small proportion are fraudulent.

Machine learning algorithms learn from examples; the more examples it has, the more confident it can be in the patterns it has discovered. If your data is unbalanced, the model may not be able to identify what patterns are associated with the minority categories.

Care must be taken with the performance metric you use when working with unbalanced data. In our case, the ‘Yes’ class i.e the class predicting that a customer would churn is very less when compared to the ‘No’ class. In such scenarios, it is recommended to consider other metrics such as precision, recall or F1-score, Auc-Roc score, Cohen-Kappa score.

Checking for presence of multicollinearity

Multicollinearity is a statistical concept where several independent variables in a model are correlated. Two variables are considered to be perfectly collinear if their correlation coefficient is +/- 1.0. Multicollinearity among independent variables will result in less reliable statistical inferences.

We have used heatmap from the seaborn library to understand the correlation among the independent variables

```
4]:
plt.figure(figsize = (15,8))

sns.heatmap(df.corr(),annot = True)
```

```
ut[24]: <AxesSubplot:>
```

We could notice that there is a high positive correlation between Age and the Target variable. However, there is no multicollinearity present within the independent variables. We also confirmed the same using Variance inflation factor for the independent variables.

```
In [26]:
#Check for multi-collinearity using VIF


x = df.drop('Exited',axis = 1)

x = pd.get_dummies(x,drop_first = True)

y = df['Exited']

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()

vif['vif_factor'] = [variance_inflation_factor(x.values,i) for i in range(x.shape[1])]

vif['features'] = x.columns

multi = vif.sort_values('vif_factor',ascending = False).head()
multi
```

Out[26]:

	vif_factor	features
9	2.185012	HasCrCard_1
8	1.832480	Gender_Male
10	1.714972	IsActiveMember_1
6	1.580130	Geography_Germany
3	1.348576	Balance

There are no features with a high Variance inflation factor and hence we can conclude that there is no multicollinearity present in our data.

Feature Engineering

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. In order to make machine learning work well on new tasks, it might be necessary to design and train better features. Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches.

Feature engineering efforts mainly focus on two goals,

- Preparing the dataset which is compatible with the machine learning algorithm requirements.
- Improving the performance of machine learning models to yield optimal results

We had noticed that columns like RowNumber, CustomerId, and Surname consist of only unique values and hence do not add any meaningful insights for our machine learning algorithm to process.

Hence, we dropped those columns from our dataframe and proceeded with the remaining columns for further analysis.

```
[22]: df = df.drop(['CustomerId','Surname','RowNumber'],axis = 1)
```

Feature Encoding

Feature Encoding is the conversion of Categorical features to numeric values as Machine Learning models cannot handle the text data directly. Most of the Machine Learning Algorithms performance varies based on the way in which the Categorical data is encoded.We have made use of dummy encoding to encode the categorical variables in our data.

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Standardization scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one. Standardization of a dataset is a common requirement for many machine learning estimators We will be using Standard Scaler from the sklearn library preprocessing module for our data standardization

In [17]:
```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler().fit_transform(df_num.values)

df_num = pd.DataFrame(data = ss,index = df_num.index,columns = df_num.columns)

df_num.describe()
```

Out[17]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
count	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04
mean	-2.338130e-17	7.588219e-15	-4.870326e-16	2.484679e-16	-1.400324e-16	-5.978551e-17	-8.652634e-16	-1.580958e-17
std	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00
min	-1.731878e+00	-1.741069e+00	-3.109504e+00	-1.994969e+00	-1.733315e+00	-1.225848e+00	-9.115835e-01	-1.740268e+00
25%	-8.659388e-01	-8.676501e-01	-6.883586e-01	-6.600185e-01	-6.959818e-01	-1.225848e+00	-9.115835e-01	-8.535935e-01
50%	0.000000e+00	-2.816100e-03	1.522218e-02	-1.832505e-01	-4.425957e-03	3.319639e-01	-9.115835e-01	1.802807e-03
75%	8.659388e-01	8.659939e-01	6.981094e-01	4.842246e-01	6.871299e-01	8.199205e-01	8.077366e-01	8.572431e-01
max	1.731878e+00	1.734255e+00	2.063884e+00	5.061197e+00	1.724464e+00	2.795323e+00	4.246377e+00	1.737200e+00

Splitting the data into training and test

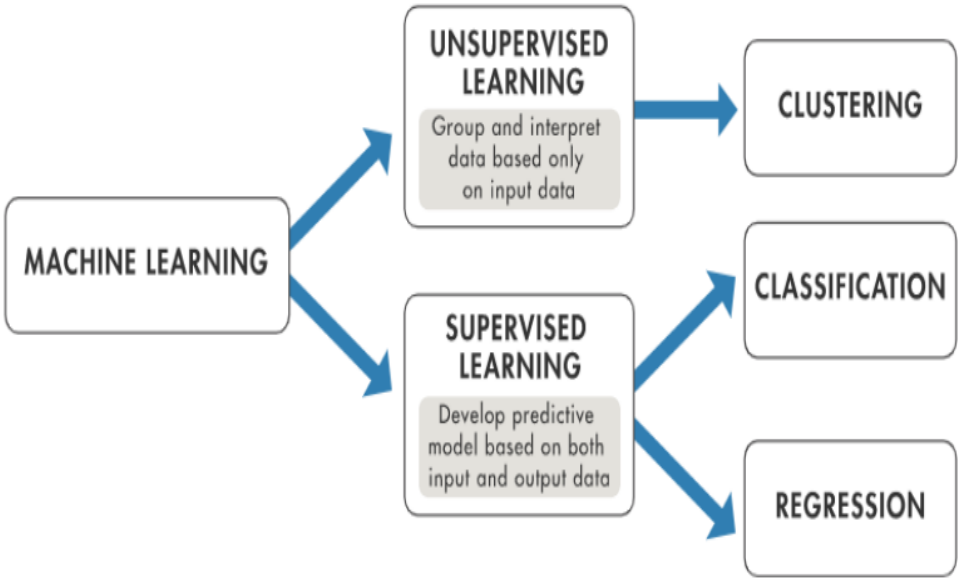
Our project deals with Supervised Learning since we have both the input data(features) and the output data (Target column : ‘Exited’)

Since we have completed data preprocessing, The data is now ready to be trained by the machine learning model. As a first step,we need to isolate the target variable from the Dataset.

x contains all the columns except for the target variable and y contains only the target column (‘Exited’) Now,the data needs to be split into a training and test set. The training set contains the data that will be used to train our machine learning model. The test set will be used to evaluate how good our model performs in unseen data.

In [28]:
```
from sklearn.model_selection import train_test_split

x = df.drop('Exited',axis = 1)

x = pd.get_dummies(x,drop_first = True)
y = df['Exited']

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 1)
```

Building a Machine Learning Model



We made use of three different machine learning algorithms,

1) Decision Tree classifier
2) Logistic Regression
3) Random Forest classifier

Once the train test split was completed, we started with the Decision tree classifier algorithm and imported the necessary libraries from sklearn library.

The base model was fit and trained using the training data. We made use of accuracy_score metrics to understand the accuracy of how our model performs in the training and test data.

```
In [29]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import accuracy_score

In [30]: dtc = DecisionTreeClassifier(random_state = 1)

         model = dtc.fit(x_train,y_train)

In [31]: print('The train accuracy is ',model.score(x_train,y_train))
         print('The test accuracy is ',model.score(x_test,y_test))

         The train accuracy is  1.0
         The test accuracy is  0.8010507880910683
```

We received a training accuracy of 100% and test accuracy of 80% in our base model. Ideally, we can never have a complete 100% accuracy in either train/test data. This basically means that the model is capturing unwanted noise in the data or in other words overfitting is present in the model.

At this point, we decided to make use of Recursive feature elimination and eliminate the insignificant features in the model.

Recursive Feature Elimination

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model's coef_ or feature_importances_ attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model.

In [32]:
```
from sklearn.feature_selection import RFE
dtc = DecisionTreeClassifier(random_state = 1,class_weight = 'balanced')
rfe_model = RFE(estimator = dtc,n_features_to_select = 8)
rfe_model = rfe_model.fit(x_train,y_train)
```

In [33]:
```
df_rfe = pd.DataFrame()
df_rfe['features'] = x.columns
df_rfe['ranking'] = rfe_model.ranking_

top = df_rfe[df_rfe['ranking']==1]
top = top.features.to_list()
top
```

Out[33]:
```
['CreditScore',
 'Age',
 'Tenure',
 'Balance',
 'NumOfProducts',
 'EstimatedSalary',
 'Geography_Germany',
 'IsActiveMember_1']
```

As you can see, we received the top 8 features that actually contribute to the dependent/target variable. The second model was built using only the selected features from RFE and the performance was once again measured using accuracy_score.

In [34]:
```
x_top = x[top]


x_train,x_test,y_train,y_test = train_test_split(x_top,y,test_size = 0.3,random_state = 1)

model_top = dtc.fit(x_train,y_train)
```

In [35]:
```
print('The train accuracy is ',model_top.score(x_train,y_train))
print('The test accuracy is ',model_top.score(x_test,y_test))
```

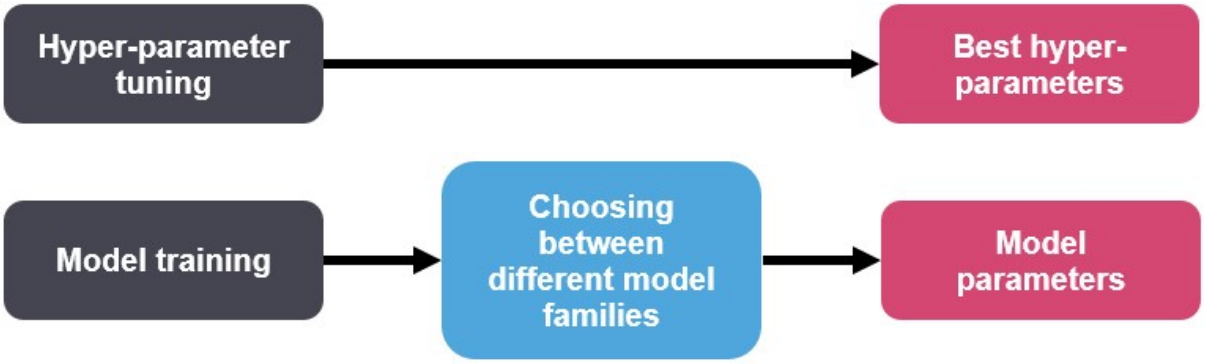
```
The train accuracy is  1.0
The test accuracy is  0.7950963222416813
```

However, the overfitting was still not handled in the second model. Hence, we decided to perform Hyper parameter tuning in order to get the best possible accuracy using the optimal parameters of the selected algorithm.

Hyperparameter Tuning

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. Two best strategies for Hyperparameter tuning are,

- GridSearchCV
- RandomizedSearchCV



We made use of Grid Search CV to tune our model. In GridSearchCV approach, machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV, because it searches for the best set of hyperparameters from a grid of hyperparameters values.

```
In [36]: from sklearn.model_selection import GridSearchCV

         tuned_params = [{'criterion':['gini','entropy'],'max_depth': range(2,10),'min_samples_split' : range(2,10)}]

         grid = GridSearchCV(estimator = dtc,param_grid = tuned_params,cv = 5)

         grid.fit(x_train,y_train)

Out[36]: GridSearchCV(cv=5,
                       estimator=DecisionTreeClassifier(class_weight='balanced',
                                                        random_state=1),
                       param_grid=[{'criterion': ['gini', 'entropy'],
                                    'max_depth': range(2, 10),
                                    'min_samples_split': range(2, 10)}])

In [37]: grid.best_params_

Out[37]: {'criterion': 'gini', 'max_depth': 3, 'min_samples_split': 2}

In [38]: dtc_tuned = DecisionTreeClassifier(random_state = 1,criterion = 'gini',max_depth = 3,min_samples_split = 2,class_weight = 'ba

         model_final = dtc_tuned.fit(x_train,y_train)

In [39]: print('The train accuracy is ',model_final.score(x_train,y_train))
         print('The test accuracy is ',model_final.score(x_test,y_test))

         The train accuracy is  0.8361861861861862
         The test accuracy is  0.8329246935201401
```

Once the hyper parameter tuning was done, the model was fit using the best set of parameters that we received from the grid search analysis.

Now, the training accuracy was 83.6% and the test accuracy was 83.2%. Since the training accuracy is approximately equal to the test accuracy, we can now conclude that there is no overfitting/underfitting in the model.

We carried out the similar process for both Logistic Regression and Random Forest classification algorithms as well.

The final results for Logistic regression,

```
In [61]: from sklearn.model_selection import train_test_split

         x = df.drop('Exited',axis = 1)

         x = pd.get_dummies(x,drop_first = True)
         y = df['Exited']

         x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 1)

In [62]: from sklearn.linear_model import LogisticRegression

         lr = LogisticRegression(random_state = 1,C = 0.01,penalty='l2',class_weight = 'balanced')

         model_lr = lr.fit(x_train,y_train)

In [63]: print('The train accuracy is ',model_lr.score(x_train,y_train))
         print('The test accuracy is ',model_lr.score(x_test,y_test))

         The train accuracy is  0.7273273273273273
         The test accuracy is  0.7260945709281962
```

The final results for Random Forest,

```
In [54]: rf_tuned = RandomForestClassifier(random_state = 1,max_depth =9)

         model_rf = rf_tuned.fit(x_train,y_train)

In [55]: print('The train accuracy is ',model_rf.score(x_train,y_train))
         print('The test accuracy is ',model_rf.score(x_test,y_test))

         The train accuracy is  0.8924924924924925
         The test accuracy is  0.8644483362521891
```

Metrics calculated for the final models built using different classification algorithms

As of now, we have only used accuracy_score from sklearn metrics library. However, since our target class is highly imbalanced, we need to make use of other metrics to evaluate the success of the model.

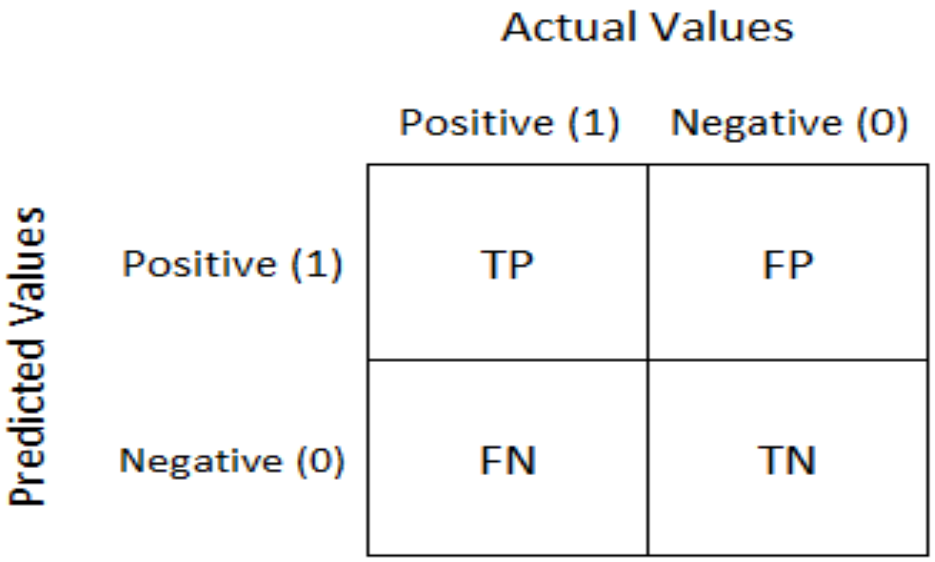
Classification Report

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. For a clear understanding, I have explained all of the metrics below so that you can easily understand the classification report of your machine learning model

Metrics	Definition
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.**

Confusion Matrix

Confusion Matrix is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

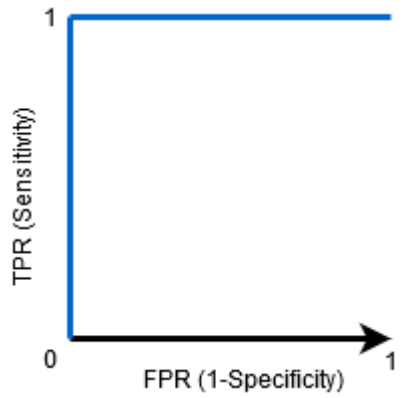


It is extremely useful for measuring Recall, Precision, Specificity, Accuracy, and most importantly AUC-ROC curves.

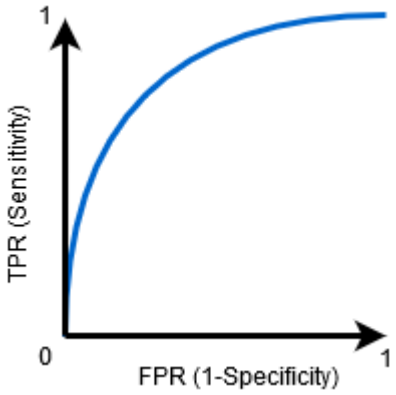
AUC-ROC curve

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the ‘signal’ from the ‘noise’. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

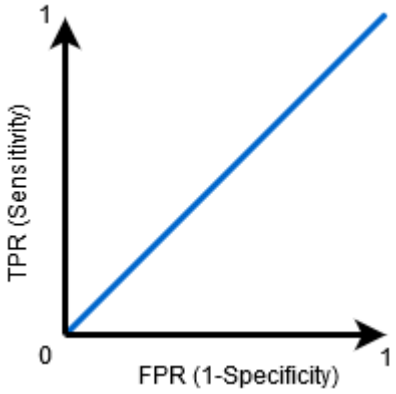
The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.



When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.



When AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points.

So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.

Cohen-Kappa

Cohen's Kappa statistic is a very useful, but under-utilised, metric. Sometimes in machine learning we might face a problem with imbalanced classes. Accuracy can be misleading, so we go for measures such as precision and recall. There are ways to combine the two, such as the F-measure, but the F-measure does not have a very good intuitive explanation, other than it being the harmonic mean of precision and recall.

Cohen's kappa statistic is a very good measure that can handle both multi-class and imbalanced class problems.

Cohen's kappa is defined as,

$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e},$$

where po is the observed agreement, and pe is the expected agreement. It basically tells you how much better your classifier is performing over the performance of a classifier that simply guesses at random according to the frequency of each class.

Cohen's kappa is always less than or equal to 1. Values of 0 or less, indicate that the classifier is useless. There is no standardised way to interpret its values. Landis and Koch (1977) provide a way to characterise values. According to their scheme a value < 0 is indicating no agreement , 0–0.20 as slight, 0.21–0.40 as fair, 0.41–0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1 as almost perfect agreement.

Cohen's kappa is provided by many software packages and libraries such as caret, Weka and scikit-learn.

4. Model evaluation Describe the final model (or ensemble) in detail. What was the objective, what parameters were prominent, and how did you evaluate the success of your models(s)? A convincing explanation of the robustness of your solution will go a long way to supporting your solution.

The metrics that were mentioned in the previous paragraph were considered for all the final models that we build for each Algorithm.

You may find the results below,

Accuracy

Decision Tree Classifier

```
print('The train accuracy is ',model_final.score(x_train,y_train))
print('The test accuracy is ',model_final.score(x_test,y_test))
```

```
The train accuracy is  0.8361861861861862
The test accuracy is  0.8329246935201401
```

Logistic Regression

```
print('The train accuracy is ',model_lr.score(x_train,y_train))
print('The test accuracy is ',model_lr.score(x_test,y_test))
```

```
The train accuracy is  0.7273273273273273
The test accuracy is  0.7260945709281962
```

Random Forest Classifier

```
print('The train accuracy is ',model_rf.score(x_train,y_train))
print('The test accuracy is ',model_rf.score(x_test,y_test))
```

```
The train accuracy is  0.8924924924924925
The test accuracy is  0.8644483362521891
```

Classification Report

Decision Tree Classifier

```
y_pred = model_final.predict(x_test)

from sklearn.metrics import classification_report

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

```
              precision    recall  f1-score   support

           0       0.88      0.91      0.90      2293
           1       0.59      0.52      0.55       562

    accuracy                           0.83      2855
   macro avg       0.74      0.71      0.72      2855
weighted avg       0.83      0.83      0.83      2855
```

Logistic Regression

```
from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.91      0.73      0.81      2293
           1       0.39      0.70      0.50       562

    accuracy                           0.73      2855
   macro avg       0.65      0.72      0.66      2855
weighted avg       0.81      0.73      0.75      2855
```

Random Forest Classifier

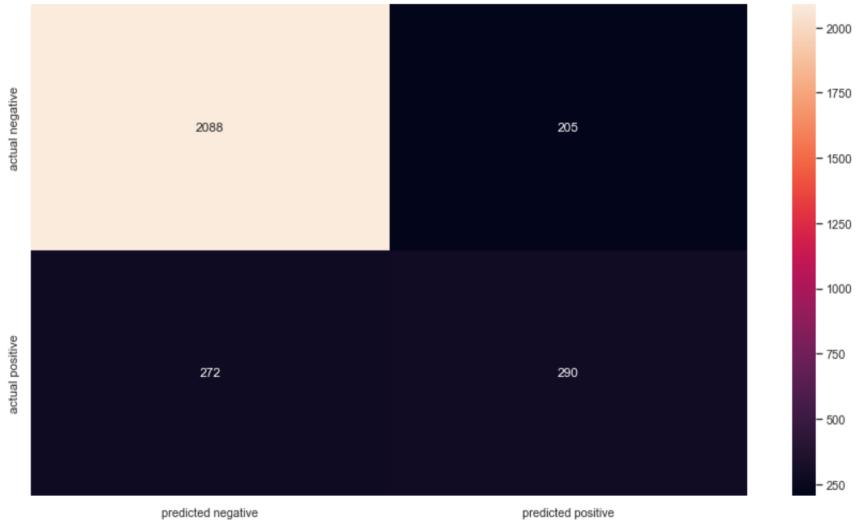
```
y_pred_rf = model_rf.predict(x_test)

from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      2293
           1       0.78      0.43      0.56       562

    accuracy                           0.86      2855
   macro avg       0.83      0.70      0.74      2855
weighted avg       0.86      0.86      0.85      2855
```

Confusion Matrix

Decision Tree Classifier

```
from sklearn.metrics import confusion_matrix
plt.figure(figsize = (15,8))
cm = confusion_matrix(y_test,y_pred)

conf = pd.DataFrame(data=cm,columns=['predicted negative','predicted positive'],index=['actual negative','actual positive'])
sns.heatmap(conf,annot=True,fmt='g')
plt.show()
```

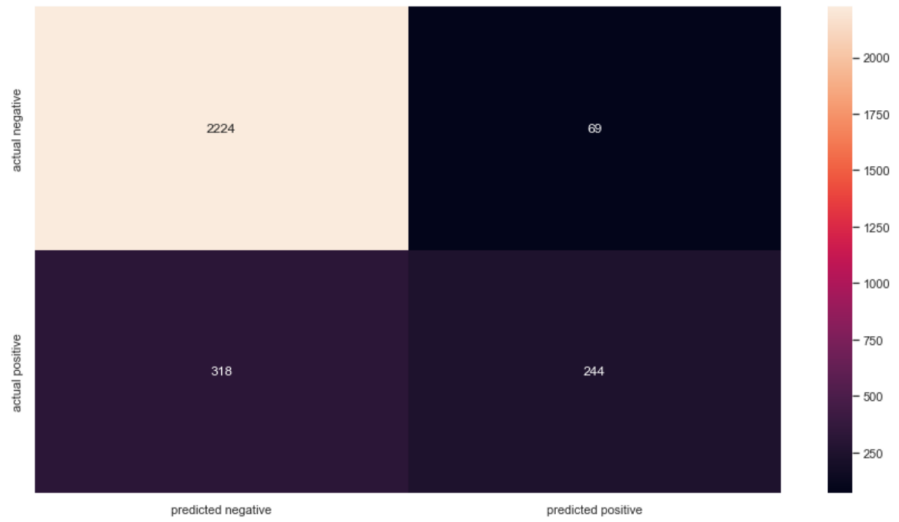


Logistic Regression

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred_lr)

plt.figure(figsize = (15,8))

conf = pd.DataFrame(data=cm,columns=['predicted negative','predicted positive'],index=['actual negative','actual positive'])
sns.heatmap(conf,annot=True,fmt='g')
plt.show()
```

Random Forest Classifier

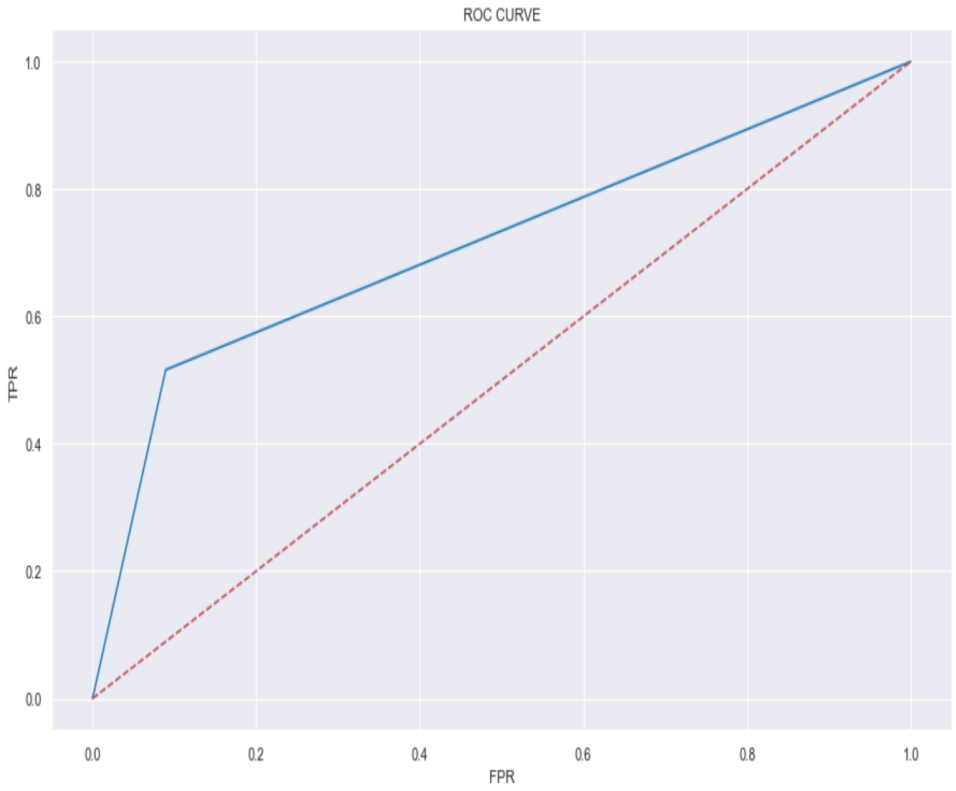
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred_rf)

plt.figure(figsize = (15,8))

conf = pd.DataFrame(data=cm,columns=['predicted negative','predicted positive'],index=['actual negative','actual positive'])
sns.heatmap(conf,annot=True,fmt='g')
plt.show()
```



ROC-AUC Curve

Decision Tree Classifier

In [42]:
```
from sklearn.metrics import roc_auc_score,roc_curve

y_pred = model_final.predict(x_test)

fpr,tpr,th = roc_curve(y_test,y_pred)
plt.figure(figsize = (15,8))
plt.plot(fpr,tpr,label = 'Decision Tree classifier')

plt.plot([0,1],[0,1],'r--')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```

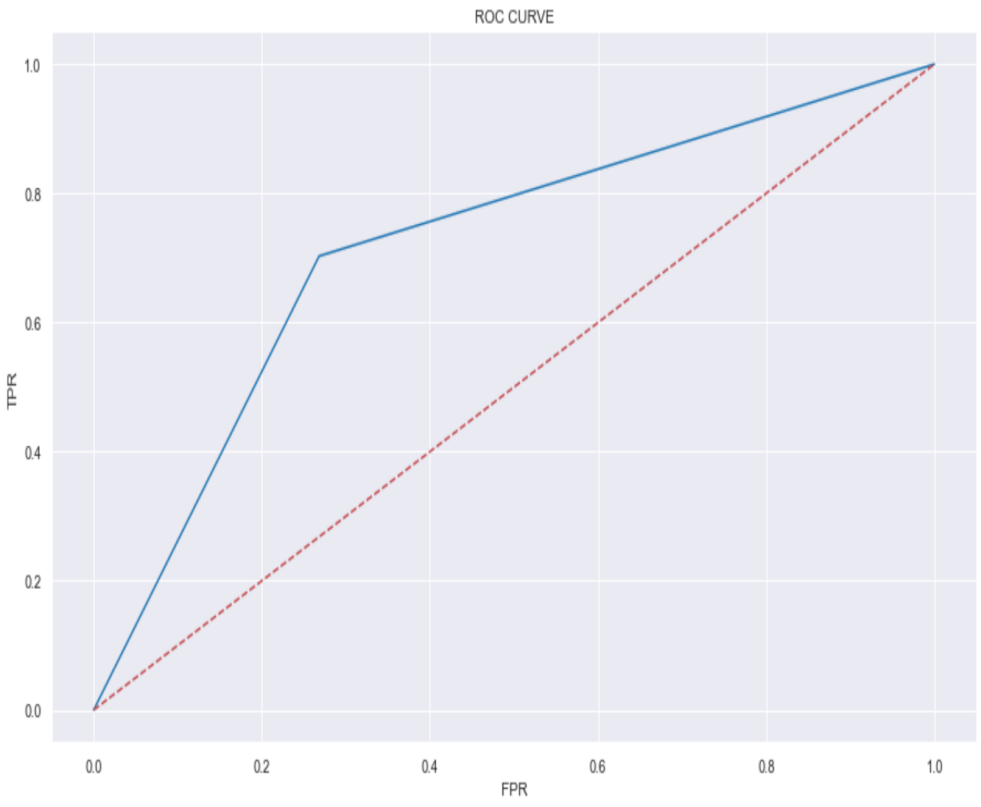


In [43]:
```
print(roc_auc_score(y_test,y_pred))
```

```
0.7133058527190134
```

Logistic Regression

In [67]:
```
fpr,tpr,th = roc_curve(y_test,y_pred_lr)
plt.figure(figsize = (15,8))
plt.plot(fpr,tpr,label = 'Logistic Regression')

plt.plot([0,1],[0,1],'r--')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```

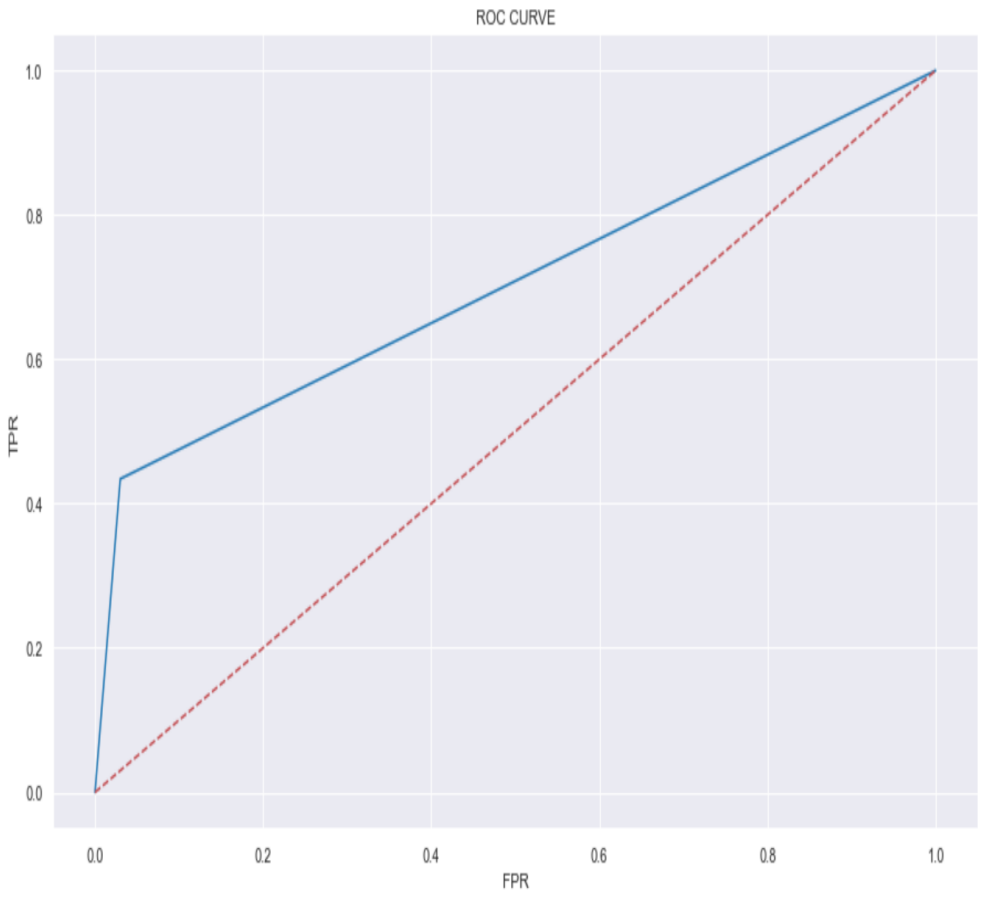


In [68]:
```
print(roc_auc_score(y_test,y_pred_lr))
```

```
0.7173196933883567
```

Random Forest Classifier

In [58]:
```
fpr,tpr,th = roc_curve(y_test,y_pred_rf)
plt.figure(figsize = (15,8))
plt.plot(fpr,tpr,label = 'Random Forest Classifier')

plt.plot([0,1],[0,1],'r--')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



In [59]:
```
print(roc_auc_score(y_test,y_pred_rf))
```

```
0.7020360589943399
```

After calculating all the metrics, we had compared all results for all three algorithms and created a DataFrame for better understanding.

```
In [70]: comparison = pd.DataFrame(data = [['Decision Tree Classifier',83,290,2088,205,272,0.71,0.44],['Logistic Regression',72,395,16
                                           ['Random Forest classifier',86,244,2224,69,318,0.70,0.48]],
                                 columns = ['Algorithm Name','Accuracy','True Positive','True Negative','False Positive','False Nega

         comparison
```

Out[70]:

	Algorithm Name	Accuracy	True Positive	True Negative	False Positive	False Negative	ROC-AUC Score	Cohen-Kappa Score
0	Decision Tree Classifier	83	290	2088	205	272	0.71	0.44
1	Logistic Regression	72	395	1678	615	167	0.71	0.33
2	Random Forest classifier	86	244	2224	69	318	0.70	0.48

Since our target class is highly imbalanced, we did not give much priority to the accuracy of the model. We set a benchmark of getting at least more than 70% accuracy and we successfully exceeded our benchmark for all the algorithms

Our main goal was to look for other metrics like Precision,recall,F1 score, Roc-auc score etc. From the classification reports, we could see that all three algorithms were able to predict the majority class well but failed to predict the minority class as well as the former. This was the trend that we observed in all the three algorithms.

Hence, we decided to use a confusion matrix and understand how the model was able to predict customer churn and finalise based on those results.

For our domain,

True Positive => Model predicting that a customer would churn and customer actually churns

False Positive => Model predicting that a customer would churn but customer does not churn

True Negative => Model predicting that a customer would not churn and the customer gets retained

False Negative => Model predicting that a custom would not churn but the customer actually churns

So when we keep in mind our business domain, False Negatives have a higher impact than False Positives. Since the purpose of the model was to predict if a customer would churn or not, it is essential for the model to not falsely predict that a customer would be retained when in reality the customer would churn.

Based on the above, we could see that Logistic Regression has the least amount of False negatives in the data and hence using this model, we would be able to predict the customers who would churn well in advance and provide attractive offers or build retention strategies for the same.

However, we also identified that the False positives were very abnormally high in the Logistic Regression model. This would end up giving lots of offers/discounts to customer who were in fact never considering leaving the business in the first place. This wouldn't be an ideal model for the business as well.

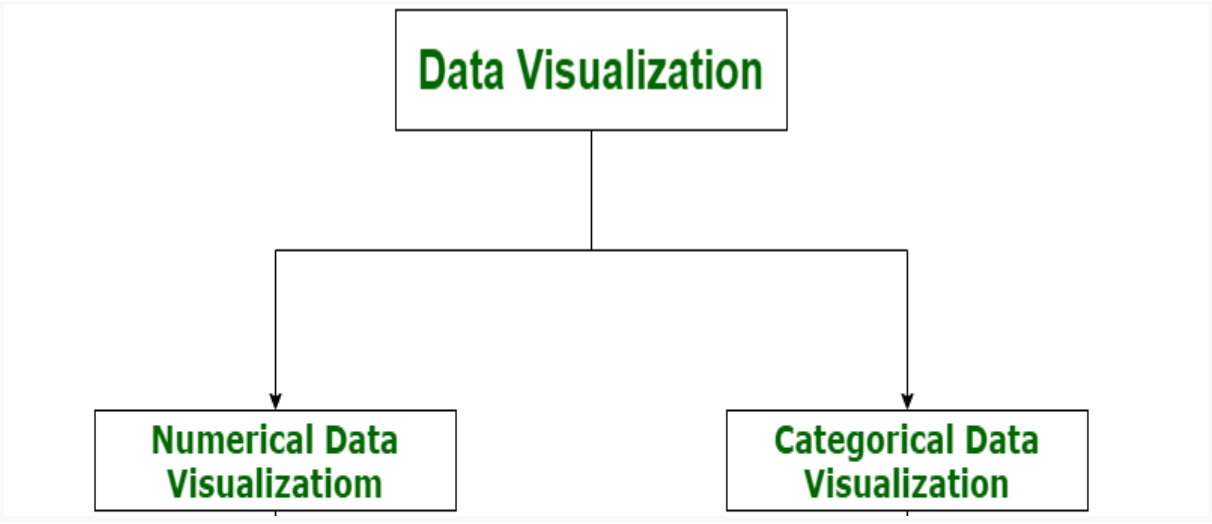
This leaves us with deciding between Random Forest or Decision tree model. Both the models had very similar results when compared to False Negatives, True Negative and True positives. However, the Random Forest classifier has the lowest amount of False positives when compared to all the models. Hence, we have chosen the model trained using Random Forest classifier as our final model.

6. Visualization(s) In addition to quantifying your model and the solution, please include all relevant visualizations that support the ideas/insights that you gleaned from the data.

Data visualization is the graphical representation of information and data in a pictorial or graphical format(Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analyzing massive amounts of information and making data-driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are Charts, Tables, Graphs, Maps, Dashboards.

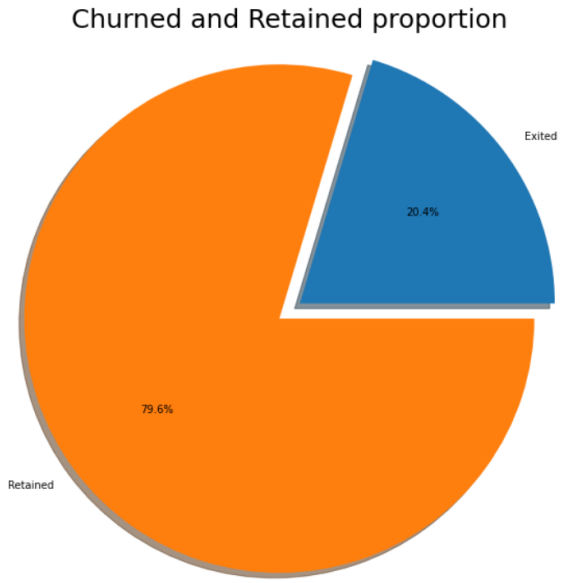
Categories of Data Visualization

Data visualization is very critical to market research where both numerical and categorical data can be visualized, which helps in an increase in the impact of insights and also helps in reducing the risk of analysis paralysis. So, data visualization is categorized into the following categories



Visualization of Percentage of customers Churned vs Retained

```
fig, axs = plt.subplots(figsize=(20, 10))
sizes = [df.Exited[df['Exited']==1].count(), df.Exited[df['Exited']==0].count()]
axs.pie(sizes, explode=(0, 0.1), labels=['Exited', 'Retained'], autopct='%1.1f%%',shadow=True)
axs.axis('equal')
plt.title("Churned and Retained proportion", size = 25)
plt.show()
```

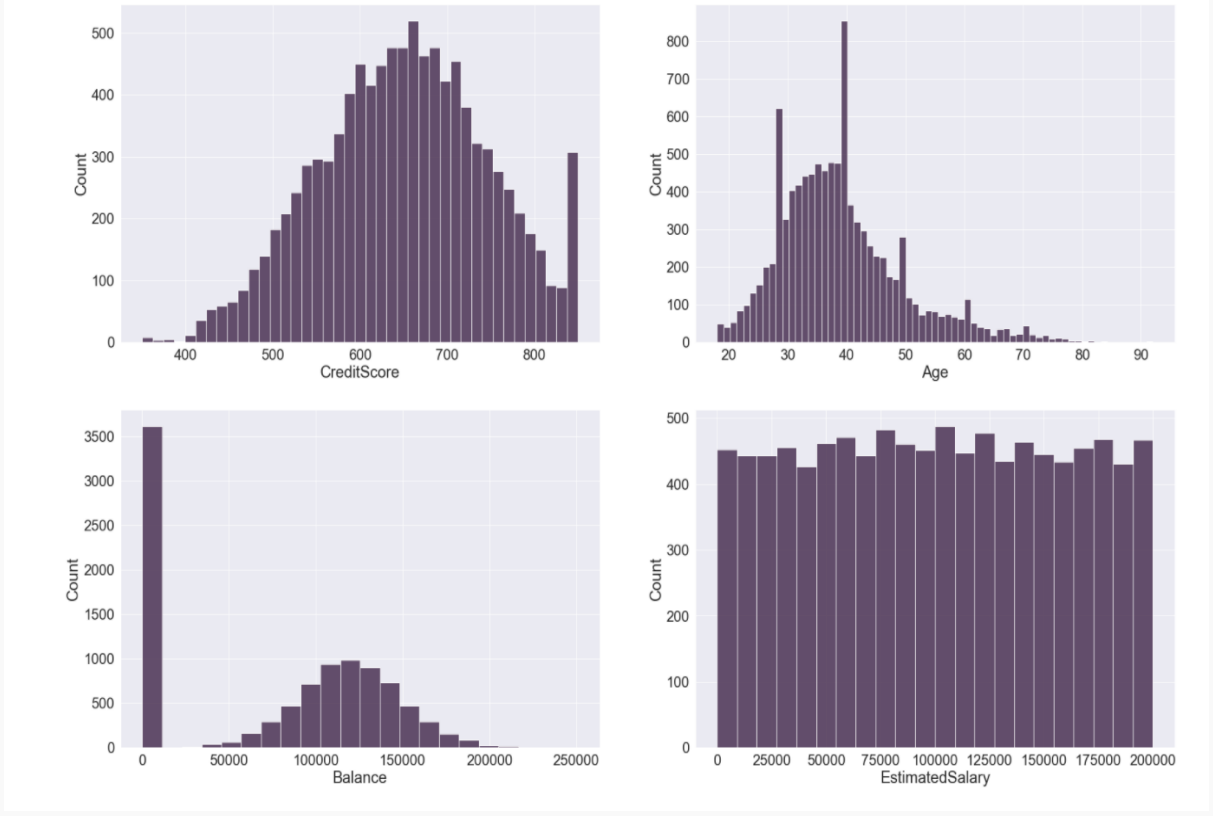


We are making use of the target column to check the number of customers who have churned/retained. The Target column has only Binary values where 1 indicates that the customer has churned/exited and 0 indicates that the customer has been retained.

From the above graph we can see that around 20% of customers have churned in the given time period

Continuous Variable Distribution

```
sns.set(rc={'figure.figsize':(30,20)})
sns.set(font_scale = 2)
fig,axs = plt.subplots(2,2)
sns.set_theme(palette="rocket")
sns.histplot(data = df,x = "CreditScore",ax=axs[0,0])
sns.histplot(data = df,x = "Age",ax=axs[0,1])
sns.histplot(data = df,x = "Balance",ax=axs[1,0])
sns.histplot(data = df,x = "EstimatedSalary",ax=axs[1,1])
plt.show()
```



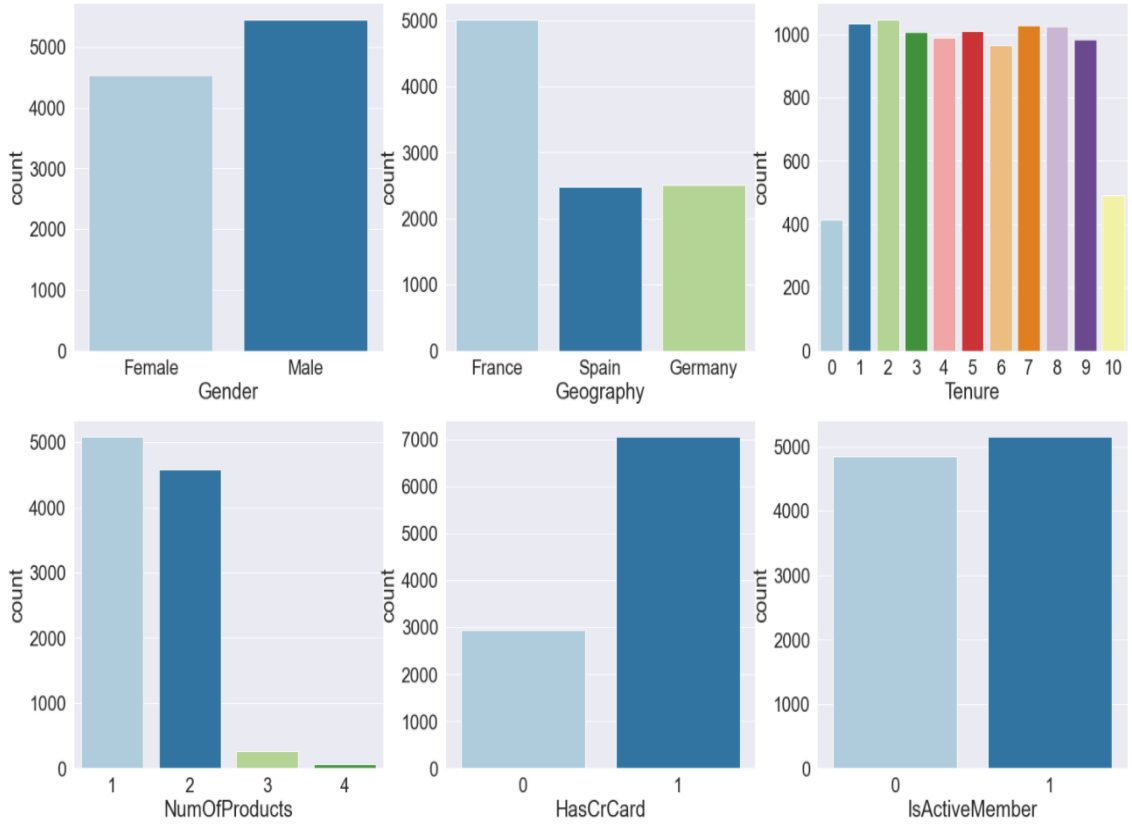
Insights from distribution of continuous data

- We can see that the majority of our customers are having Credit Score less than 700.However, the ideal credit score is 700 or above. We can interpret that there are more number of non-reliable customers for the bank
- The Age of the customers is skewed towards the right which means that there are few customers whose age is more when compared to mean age of customers
- There is a significant number of younger customers whose age ranges from ranging from 29 to 40 years

- The bank balance of most of the customers are normally distributed. However,approximately 3600 customers have zero bank balance in their account which contributes to one third of the total number of customers.
- The Estimated Salary of the customers have an uniform distribution. We can interpret that there are almost equal number of customers for different estimated salaries.

Categorical Variable Distribution

```
sns.set(rc={'figure.figsize':(25,15)})
sns.set(font_scale = 2)
fig,axs = plt.subplots(2,3)
sns.set_theme(palette="Paired")
sns.countplot(data = df,x = 'Gender',ax=axs[0,0])
sns.countplot(data = df,x = 'Geography',ax=axs[0,1])
sns.countplot(data = df,x = 'Tenure',ax=axs[0,2])
sns.countplot(data = df,x = 'NumOfProducts',ax=axs[1,0])
sns.countplot(data = df,x = 'HasCrCard',ax=axs[1,1])
sns.countplot(data = df,x = 'IsActiveMember',ax=axs[1,2])
plt.show()
```



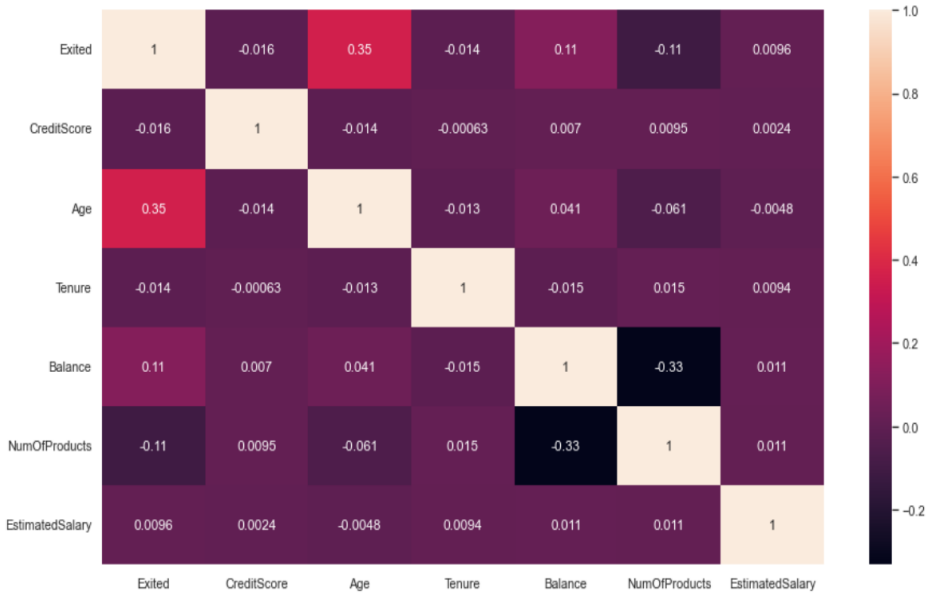
Insights from distribution of categorical columns

- Most of the customers in the bank are Male.
- We have more customers from France and almost the same number of customers from Germany and Spain.
- Majority of our customers have a good tenure with the bank ranging from 1 to 9 years.
- Most of the customers have at least one product purchased from the bank.
- Almost 70% of the customers use credit card.
- The number of customers who are digitally active are almost the same as digitally inactive customers

Correlation between continuous columns

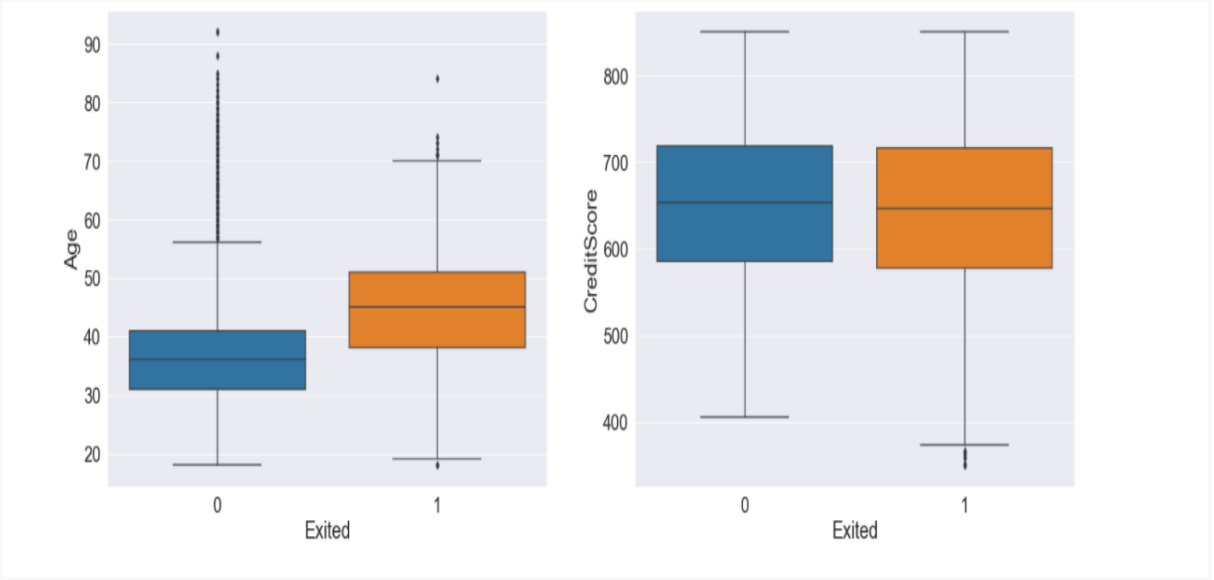
```
In [24]:
plt.figure(figsize = (15,8))

sns.heatmap(df.corr(),annot = True)
```

```
Out[24]: <AxesSubplot:>
```



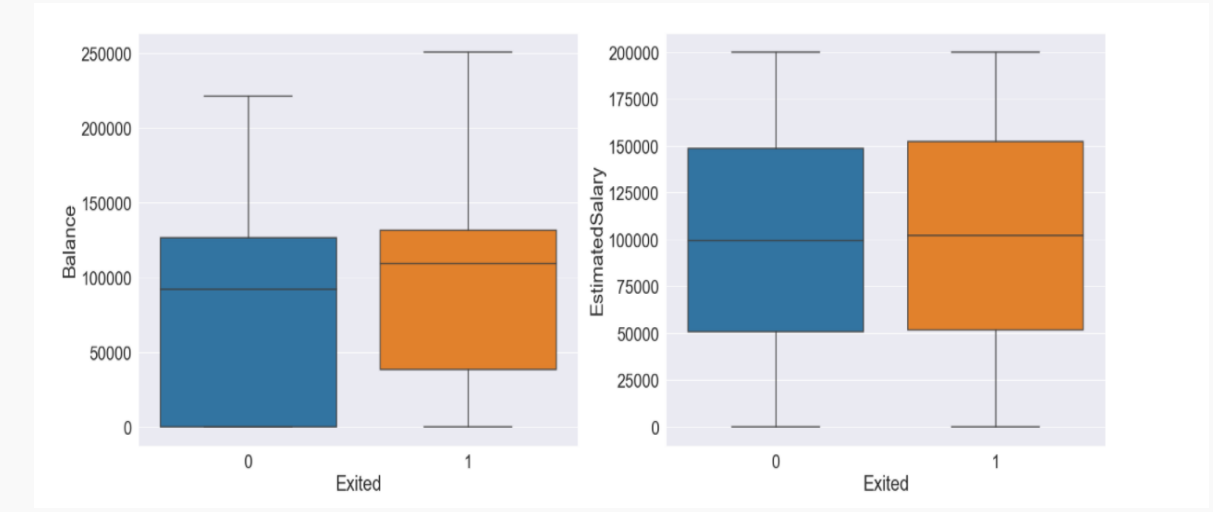
We could notice that there is a high positive correlation between Age and the Target variable. However, there is no multicollinearity present within the independent variables.

Continuous Variable Exploration with respect to Exit Status

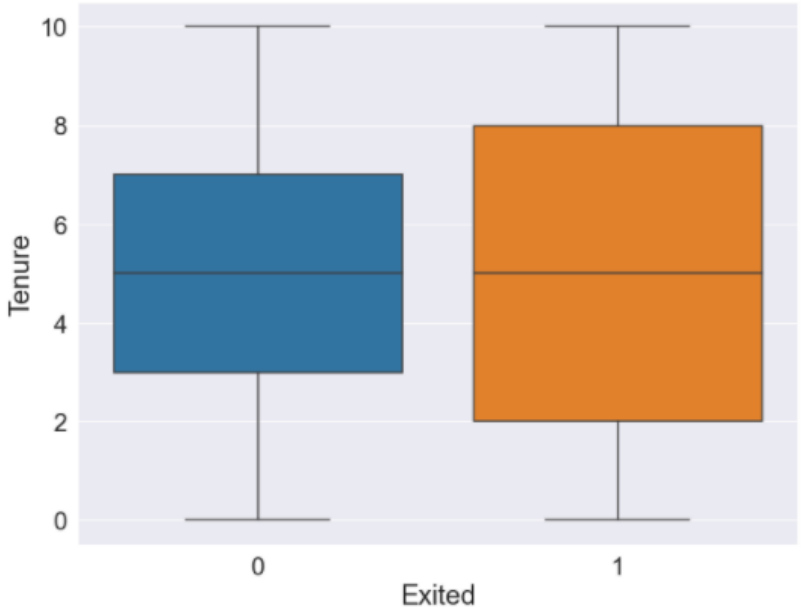


Age has a significant impact on customer churn. Customers with higher age tend to churn more as compared to younger customers.This also confirms the insight that we derived from the correlation plot.

There is no significant difference in the credit score distribution between retained and churned customers



Customers with high balance tend to churn more when compared to customers having lower balance. The estimated salary has no significant effect on customer churn.



The customers on either extreme (spent little time with the bank or a lot of time with the bank) are more likely to churn compared to those that have an average tenure with the bank.

Categorical Variable Exploration with respect to Exit Status

```
df[['Geography','Gender','Exited']].groupby(['Geography','Gender']).agg(['mean','count'])
```

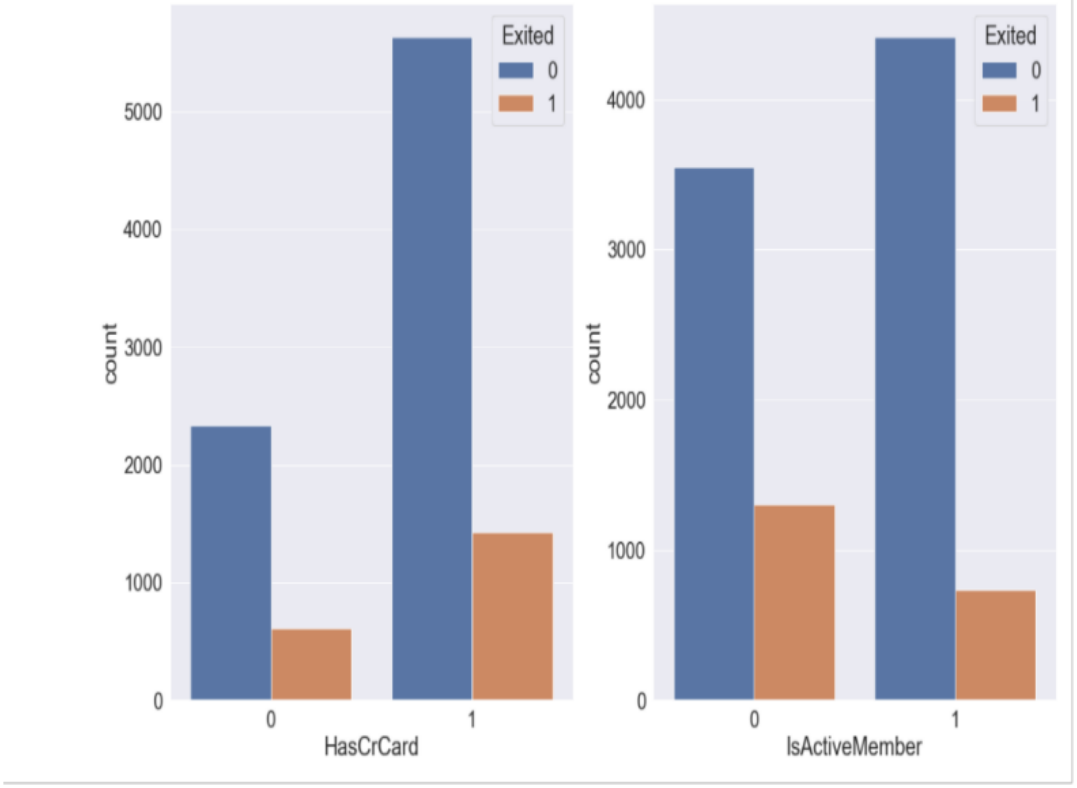
		Exited	
		mean	count
Geography	Gender		
France	Female	0.203450	2261
	Male	0.127134	2753
Germany	Female	0.375524	1193
	Male	0.278116	1316
Spain	Female	0.212121	1089
	Male	0.131124	1388

Though there are more number of male populations, females tend to churn more.

The churn rate in Germany is higher than France and Spain. France and Spain have an equal number of customers who have churned.

However there are more customers from France and hence France has the highest customer retention rate when compared with all three region.

```
In [72]: fig, axarr = plt.subplots(1,2 ,figsize=(20, 12))
         sns.set(font_scale = 2)
         sns.countplot(x='HasCrCard', hue = 'Exited',data = df, ax=axarr[0])
         sns.countplot(x='IsActiveMember', hue = 'Exited',data = df, ax=axarr[1])
Out[72]: <AxesSubplot:xlabel='IsActiveMember', ylabel='count'>
```



The customers who own a credit card have churned more in number when compared to customers who don't own a credit card.

Customers who are not digitally active tend to churn more when compared to digitally active customers

7. Implications How does your solution affect the problem in the domain or business? What recommendations would you make, and with what level of confidence?

If a company/business wants to improve their customer retention, then they need to focus on reducing churn rates through customer churn prediction.The best way to do customer churn prediction is by using machine learning and artificial intelligence-based algorithms. You can analyze data using different forms of analytics – such as predictive analytics to look at the association among different metrics.

Churn rate is a health indicator for subscription-based companies. The ability to identify customers that aren't happy with provided solutions allows businesses to learn about product or pricing plan weak points, operation issues, as well as customer preferences and expectations to proactively reduce reasons for churn.

It's important to define data sources and observation periods to have a full picture of the history of customer interaction. Selection of the most significant features for a model would influence its predictive performance: The more qualitative the dataset, the more precise forecasts are.

Companies with a large customer base and numerous offerings would benefit from customer segmentation. The number and choice of ML models may also depend on segmentation results. Data scientists also need to monitor deployed models, and revise and adapt features to maintain the desired level of prediction accuracy.

The final model that we have built will be able to predict the customers who are more likely to churn with 86% accuracy. The features that contribute significantly to the model are credit score,age,tenure,balance,number of products,estimated salary and whether the customer is an active member or not.

8. Limitations What are the limitations of your solution? Where does your model fall short in the real world? What can you do to enhance the solution?

Similar to many practical or real time classification problems, our data also faced the issue of class imbalance. The class imbalance problem typically occurs when there are many more instances of some classes than others

Eventually from the results, we could see that our model was not able to predict the minority class as much as we could predict for the majority class. To address the imbalance in the data, balancing schemes that augment the data to make it more balanced before training the classifier can be implemented.Oversampling the minority class by duplicating minority samples or undersampling the majority class is the simplest balancing method.

Apart from the class imbalance, we did not have more than one feature having high correlation with the target variable.Hence, as and when we have more significant data collected by the business, we would be able to increase the efficiency of the model and provide results with even better accuracy.

9. Closing Reflections What have you learned from the process? What would you do differently next time?

We went through the various tasks involved in Churn prediction in this project.One of important things that we learnt during this process was that Exploratory Data Analysis (EDA) is as important as the final prediction itself. We were able to draw many insights and represent it visually for any layman to view and understand the data.

We learnt how to apply various data preprocessing techniques that allows us to shape the data in such a way that it is tailor made for building a machine learning model.

Various machine learning algorithms were analysed, learnt and implemented in this project which gave us an overview how a model would behave for different kinds of data. We also learnt how to fine tune the model that we build and improve the accuracy of the model by selecting the best subset of features and parameters.

From the knowledge that we gained during the course of this project, we can now handle different kinds of data and provide meaningful visual inferences and also build a machine learning model from scratch that is capable of making predictions for the future.