PROJECT REPORT
ON

BANK CUSTOMER CHRUN PREDICTION

Submitted in partial fulfillment of requirements to

IT 353 – PROJECT I

By

P.V.RAMAKRISHNA(Y20IT094)
G.S.SRINIVASARAO(Y20IT132)
SK.SAINADH(Y20IT109)

FEBRUARY 2023

R.V.R & J.C.COLLEGE OF ENGINEERING(AUTONOMOUS)
(NAAC A+ GRADE) (Approved by A.I.C.T.E)
(Affiliated to Acharya Nagarjuna University)
Chandramoulipuram : : Chowdavaram
GUNTUR – 522 019

R.V.R & J.C.COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this project work titled BANK CUSTOMER CHRUN PERIDICTION is the bonafide work of (P.V.Ramakrishna(Y20IT094), G.S. Srinivasarao (Y20IT132), SK.Sainadh(Y20IT109)) who have carried out the work under my supervision, and submitted in partial fulfillment of the requirements to **IT-353, PROJECT I** during the year 2022-2023.

A.Yaswanth	**Dr. A.Srikrishna**
Lecturer Incharge	Prof.&HOD, Dept. of IT

ACKNOWLEDGEMENTS

The successful completion of any task would be incomplete without a proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Project **“BANK CUSTOMER CHRUN PREDICTION”.**

We are very much thankful to **Dr.K.Srinivas**, Principal of R.V.R. & J.C. College of Engineering, Guntur for having allowed delivering this Project - I.

We express our sincere thanks to **Dr.A.Srikrishna**, Professor, Head of the Department of Information Technology for her encouragement and support to carry out this mini project successfully.

We are very glad to express our special thanks to **A.Yaswanth** , Associate Professor in Department of Information Technology for timely, guidance and providing us with most essential materials required for the completion of this report.

Finally we express our sincere thanks to all the **Teaching** and **Non-Teaching staff** of **IT Department** who have contributed for the successful completion of this report.

P.V.RAMAKRISHNA(Y20IT094)
G.S.SRINIVASRAO(Y20IT132)
SK.SAINADH(Y20IT109)

CONTENTS

Chapter No. & Name	Page No.
1. Problem statement.	1
2. SRS Documentation - Requirements elicitation.	4
3. System Requirements Specification	5
4. Requirements modeling	6
5. Building a Business Process model using UML activity diagram.	7
6. Construction of Sequence diagrams.	10
7. Construction of UML Class diagram.	12
8. Test Definition	13
9. Code and Output	37
10. Conclusion	48
11. References	48

1.PROBLEM STATEMENT

BANK CUSTOMER CHRUN PREDICTION

Customer churn or customer attrition is a tendency of clients or customers to abandon a brand and stop being a paying client of a particular business or organization. The percentage of customers that discontinue using a company's services or products during a specific period is called a customer churn rate. Several bad experiences (or just one) are enough, and a customer may quit. And if a large chunk of unsatisfied customers churn at a time interval, both material losses and damage to reputation would be enormous. A reputed bank "ABC BANK" wants to predict the Churn rate.

Create a model by using different machine learning approaches that can predict the best result.

1.1 Introduction

Abstract:-

The Problem is based on the domain of the banking sector where the bank wants to predict the Churn of a customer depending upon the previous data of the customer. By churn it is meant that the bank wants to predict if a customer would be a defaulter in the next quarter depending upon its previous credit history.

What is the problem?

The main problem is to predict if a customer would be credit defaulter or not depending upon the previous data of the customer.

Why is it important?

It is important from a bank's perspective in order to maintain business and customer relationship/ Apart from that if someone could be predicted as a defaulter then primitive measures can be taken in order to ensure that such violations do not happen.

What is your basic approach?

The basic approach of solving this problem was first studying the data , then bringing out insights from the dataset and after that I have followed a machine learning pipeline in order to solve the problem.

The ML Pipeline that we have followed is :

- Importing the necessary libraries and the dataset
- Performing Data Preprocessing (Exploratory Data Analysis and Manipulation)

- Modelling using Logistic Regression, KNN and Random Forest
- Performing Prediction

Dataset Description :

This is a public dataset, The dataset format is given below. Inside the dataset, there are 10000 rows and 14 different columns. The target column here is Exited here. Download the dataset - Churn Dataset

There are multiple variables in the dataset which can be cleanly divided into 3 categories:

I. Demographic information about customers

- **customer_id** - Customer id
- **vintage** - Vintage of the customer with the bank in a number of days
- **age** - Age of customer
- **gender** - Gender of customer
- **dependents** - Number of dependents
- **occupation** - Occupation of the customer
- **city** - City of the customer (anonymized)

II. Customer Bank Relationship

- **customer_new_category** - Net worth of customer (3: Low 2: Medium 1: High)
- **branch_code** - Branch Code for a customer account
- **days_since_last_transaction** - No of Days Since Last Credit in Last 1 year

III. Transactional Information

- **current_balance** - Balance as of today
- **previous_month_end_balance** - End of Month Balance of previous month
- **average_monthly_balance_prevQ** - Average monthly balances (AMB) in Previous Quarter
- **average_monthly_balance_prevQ2** - Average monthly balances (AMB) in previous to the previous quarter
- **current_month_credit** - Total Credit Amount current month
- **previous_month_credit** - Total Credit Amount previous month
- **current_month_debit** - Total Debit Amount current month
- **previous_month_debit** - Total Debit Amount previous month
- **current_month_balance** - Average Balance of current month
- **previous_month_balance** - Average Balance of previous month
- **churn** - Average balance of customer falls below minimum balance in the next quarter (1/0)

1. SRS DOCUMENTATION - REQUIREMENTS ELICITATION

ID	**Variable**	**Definition**
R1	RowNumber	Unique Row Number
R2	CustomerId	Unique Customer Id
R3	Surname	Surname of a customer
R4	CreditScore	Credit Score of each Customer
R5	Geography	Graphical Location of Customers
R6	City_Category	Category of the City(A,B,C)
R7	Gender	Male or Female
R8	Age	Age of Each Customer
R9	Tenure	Number of years
R10	Balance	Current Balance Of Customers
R11	NumOfProducts	Number of Products
R12	HasCrCard	If a customer has a credit card or not
R13	IsActiveMember	If a customer is Active or not
R14	EstimatedSalary	Estimated Salary of each Customer
R15	Exited	Customer left the bank or Not (Target Variable)

2.SYSTEM REQUIREMENT SPECIFICATION

Software Requirements:

- Operating System : windows XP
- Coding language : PHP , HTML , PYTHON
- Data Base : .CSV file

Hardware Requirements:

- Personal computer with keyboard and mouse
- Processor : Intel® core™ i5
- Installed Memory (RAM) : 1.00 GB
- Hard Disc : 40 GB

3. REQUIREMENTS MODELING

The most important factor for the success of an IS project is whether the software product satisfies its users' requirements. Models constructed from an analysis perspective focuses on determining these requirements. This means Requirement Model includes gathering and documenting facts and requests.

The use case model gives a perspective on many user requirements and models them in terms of what the software system can do for the user. Before the design of software which satisfies user requirements, we must analyze both the logical structure of the problem situation, and also the ways that its logical elements interact with each other. We must also need to verify the way in which different, possibly conflicting, requirements affect each other. Then we must communicate this under standing clearly and unambiguously to those who will design and build the software.

Use-case diagrams graphically represents system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may contain all or some of the use cases of a system.

A use-case diagram can contain:

- ·actors ("things" outside the system)
- ·use cases (system boundaries identifying what the system should do)
- interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

4. BUILDING A BUSINESS PROCESS MODEL USING UML ACTIVITY DIAGRAM

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. You can also use activity diagrams to model code-specific information such as a class operation.

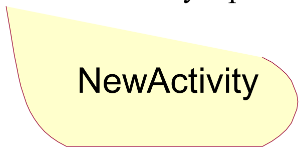
Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

- Activity Diagrams also may be created at this stage in the life cycle. These diagrams represent the dynamics of the system. They are flow charts that are used to show the workflow of a system; that is, they show the flow of control from activity to activity in the system, what activities can be done in parallel, and any alternate paths through the flow.
- At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.
- Later in the life cycle, activity diagrams may be created to show the workflow for an operation.

The following tools are used on the activity diagram toolbox to model activity diagrams:

Activities:

An activity represents the performance of some behavior in the workflow.

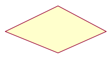


Transitions:

Transitions are used to show the passing of the flow of control from activity to activity. They are typically triggered by the completion of the behavior in the originating activity.

Decision Points:

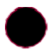
When modeling the workflow of a system it is often necessary to show where the flow of control branches based on a decision point. The transitions from a decision point contain a guard condition, which is used to determine which path from the decision point is taken. Decisions along with their guard conditions allow you to show alternate paths through a work flow.



Decision point

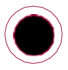
Start state:

A start state explicitly shows the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram.



End state:

An End state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.



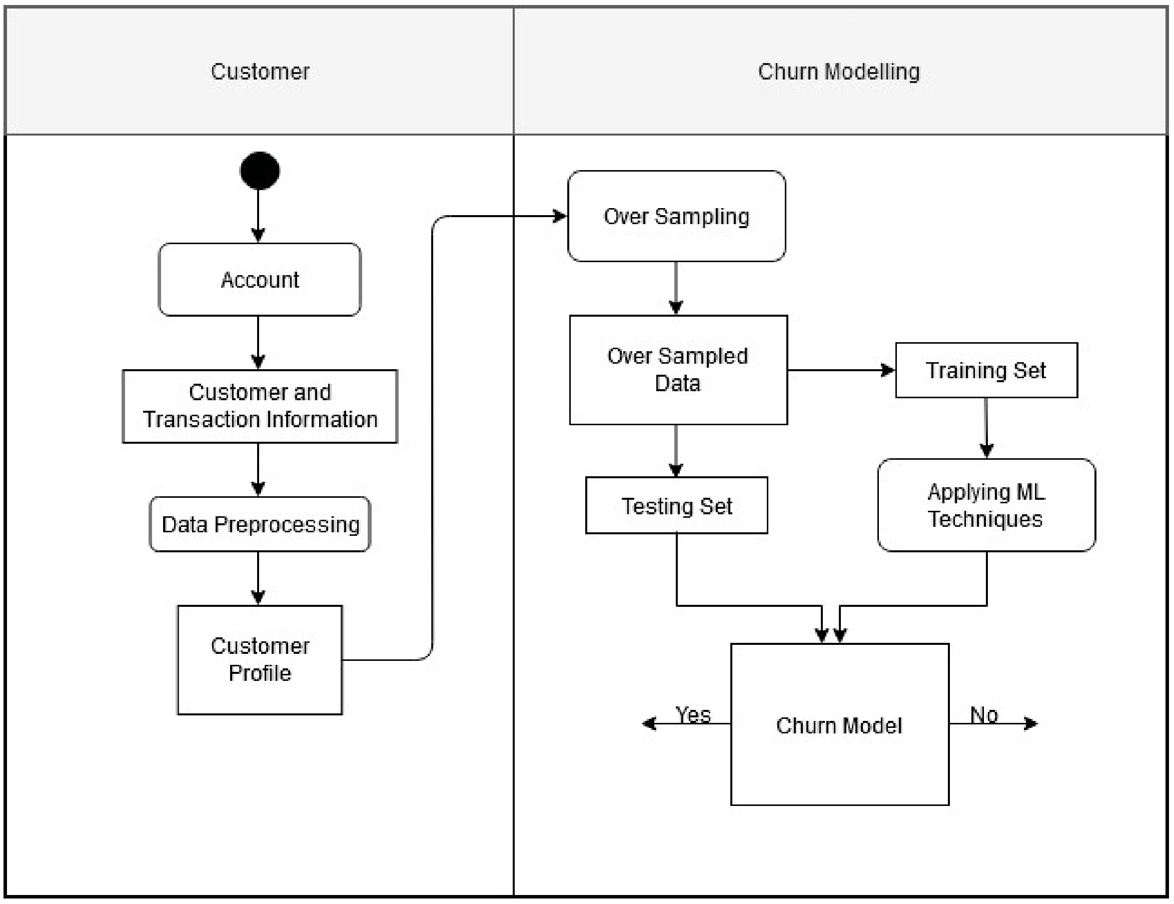
End state

Swim Lanes:

Swim lanes may be used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activities contained in the swim lane.

- Horizontal synchronization
- Vertical synchronization.

Activity Diagram



5. CONSTRUCTION OF SEQUENCE DIAGRAMS.

A sequence diagram is a graphical view of a scenario that shows object interaction in a time based sequence--what happens first, what happens next…

Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. The vertical line is called the object's lifeline. The lifeline represents the object's existence during the interaction.

Steps:

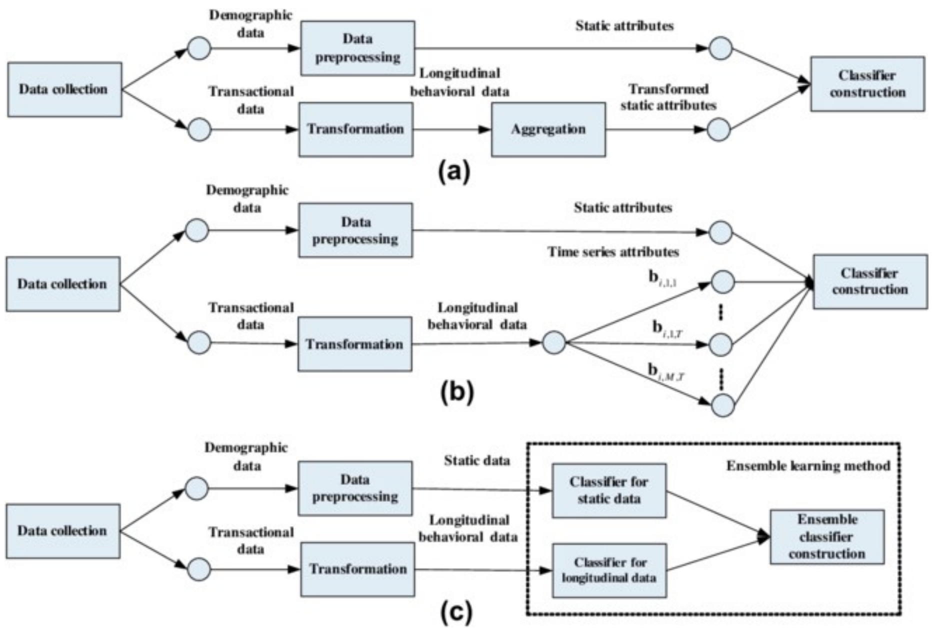
1. An object is shown as a box at the top of a dashed vertical line. Object names can be specific (e.g., Algebra 101, Section 1) or they can be general (e.g., a course offering). Often, an anonymous object (class name may be used to represent any object in the class.)
2. Each message is represented by an Arrow between the lifelines of two objects. The order in which these messages occur is shown top to bottom on the page. Each message is labeled with the message name.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

ELEMENTS OF SEQUENCE DIAGRAM:

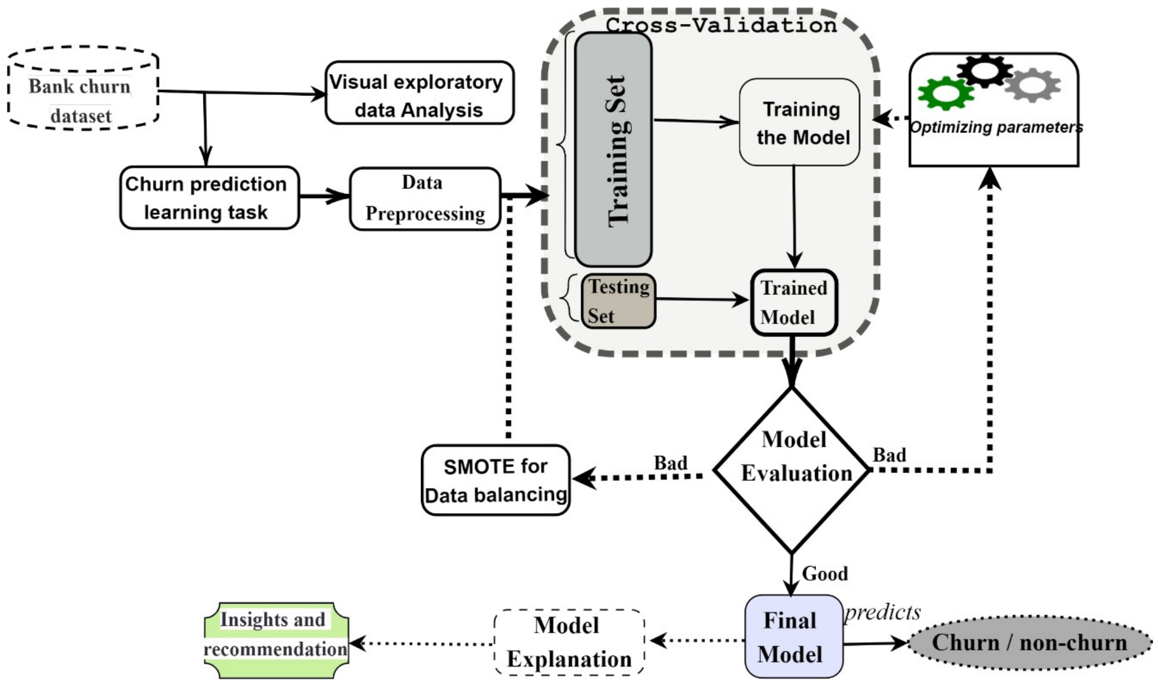
- Objects
- Links
- Messages
- Focus of control
- Object life line

SEQUENCE DIAGRAM



6. CONSTRUCTION OF UML STATIC CLASS DIAGRAM

Class diagrams contain icons representing classes, packages, interfaces, and their relationships. You can create one or more class diagrams to depict the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model.



7. TASK DEFINATION

The task was to predict those customers likely to churn balances below the minimum balance.The customers information such as age, gender, demographics along with their transactions with the bank.The task as a data scientist was to predict the propensity to churn for each customer.

The ML Pipeline that I have followed is :

- **Importing the necessary libraries and the dataset:**
  Here, the libraries such as numpy, pandas and matplotlib were called. **Numpy** is known to be Numerical Python which is responsible for performing all the Numerical tasks in this project whereas **Pandas** would make the data frame and **Matplotlib** was used for visualization.

- **Performing Data Preprocessing (Exploratory Data Analysis and Data Manipulation):-**
  In this step the data was thoroughly analysed and the steps such as Univariate Analysis and Bivariate Analysis were performed. Univariate Analysis would be the analysis of a data if we are studying/ analysing a particular data majorly depending upon its type (for e.g Continuous or Categorical Type)

If the data is said to be a continuous data then in order to do a univariate analysis we check the distribution of the data i.e, we check if the data is normally distributed or not and if the data is said to be a categorical data then we would check the bar plot of that data..In order to perform a Bivariate Analysis (The study of two data at an instance) we would have to plot the scatter plot in order to check the relationship in between them.

- **Modelling using Logistic Regression, KNN and Random Forest**

The modelling was done by using Logistic Regression , Random Forest and KNN and at the end it was ensemble by using VotingClassifier.The Following are the algorithms that were used for performing the predictions

Logistic Regression:-

Classification techniques are an essential part of machine learning and data mining applications. Approximately 70% of problems in Data Science are classification problems. There are lots of classification problems that are available, but the logistics regression is common and is a useful regression method for solving the binary classification problem. Another category of classification is Multinomial classification, which handles the issues where multiple classes are present in the target variable. For example, IRIS dataset a very famous example of multi-class classification. Other examples are classifying article/blog/document category.

Logistic Regression can be used for various classification problems such as spam detection. Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

In this tutorial, you will learn the following things in Logistic Regression:

- Introduction to Logistic Regression
- Linear Regression Vs. Logistic Regression
- Maximum Likelihood Estimation Vs. Ordinary Least Square Method
- How do Logistic Regression works?
- Model building in Scikit-learn
- Model Evaluation using Confusion Matrix.
- Advantages and Disadvantages of Logistic Regression

Logistic Regression:-

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Linear Regression Equation:

Where, y is dependent variable and x1, x2 ... and Xn are explanatory variables.

Sigmoid Function:

Apply Sigmoid function on linear regression:

Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.
- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

Linear Regression Vs. Logistic Regression

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.

Maximum Likelihood Estimation Vs. Least Square Method

The MLE is a "likelihood" maximization method, while OLS is a distance-minimizing approximation method. Maximizing the likelihood function determines the parameters that are most likely to produce the observed data. From a statistical point of view, MLE sets the mean and variance as parameters in determining the specific parametric values for a given model. This set of parameters can be used for predicting the data needed in a normal distribution.

Ordinary Least squares estimates are computed by fitting a regression line on given data points that has the minimum sum of the squared deviations (least square error). Both are used to estimate the parameters of a linear regression model. MLE assumes a joint probability mass function, while OLS doesn't require any stochastic assumptions for minimizing distance.

Sigmoid Function

The sigmoid function, also called logistic function gives an ‘S’ shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The outputcannotFor example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that patient will suffer from cancer.

Types of Logistic Regression

Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

Model building in Scikit-learn

Let's build the diabetes prediction model.

Here, you are going to predict diabetes using Logistic Regression Classifier.
Let's first load the required Pima Indian Diabetes dataset using the pandas' read CSV function. You can download data from the following link: https://www.kaggle.com/uciml/pima-indians-diabetes-database

Loading Data

```
#import pandas

import pandas as pd

col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

# load dataset

pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)
```

pima.head()

Selecting Feature

Here, you need to divide the given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
#split dataset in features and target variable

feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
```

```
X = pima[feature_cols] # Features

y = pima.label # Target variable
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split dataset by using function train_test_split(). You need to pass 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
# split X and y into training and testing sets

from sklearn.cross_validation import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

```
/home/admin/.local/lib/python3.5/site-packages/sklearn/cross_validation.py:41:
DeprecationWarning: This module was deprecated in version 0.18 in favor of the
model_selection module into which all the refactored classes and functions are moved.
Also, note that the interface of the new CV iterators is different from that of this
module. This module will be removed in 0.20.

"This module will be removed in 0.20.", DeprecationWarning)
```

Here, the Dataset is broken into two parts in a ratio of 75:25. It means 75% data will be used for model training and 25% for model testing.

Model Development and Prediction

First, import the Logistic Regression module and create a Logistic Regression classifier object using LogisticRegression() function.

Then, fit your model on the train set using fit() and perform prediction on the test set using predict().

```
# import the class

from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)

logreg = LogisticRegression()

# fit the model with data

logreg.fit(X_train,y_train)
y_pred=logreg.predict(X_test)
```

Model Evaluation using Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.

```
# import the metrics class

from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

cnf_matrix

array([[119, 11],

[ 26, 36]])
```

Here, you can see the confusion matrix in the form of the array object. The dimension of this matrix is 2*2 because this model is binary classification. You have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, 119 and 36 are actual predictions, and 26 and 11 are incorrect predictions.

Visualizing Confusion Matrix using Heatmap

Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.

Here, you will visualize the confusion matrix using Heatmap.

```
# import required modules

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

class_names=[0,1] # name of classes

fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))

plt.xticks(tick_marks, class_names)

plt.yticks(tick_marks, class_names)

# create heatmap

sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set_label_position("top")

plt.tight_layout()

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

Text(0.5,257.44,'Predicted label')
```

Confusion Matrix Evaluation Metrics

Let's evaluate the model using model evaluation metrics such as accuracy, precision, and recall.

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

print("Precision:",metrics.precision_score(y_test, y_pred))

print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.8072916666666666

Precision: 0.7659574468085106

Recall: 0.5806451612903226
```

Well, you got a classification rate of 80%, considered as good accuracy.

Precision: Precision is about being precise, i.e., how accurate your model is. In other words, you can say, when a model makes a prediction, how often it is correct. In your prediction case, when your Logistic Regression model predicted patients are going to suffer from diabetes, that patients have 76% of the time.

Recall: If there are patients who have diabetes in the test set and your Logistic Regression model can identify it 58% of the time.

ROC Curve

Receiver Operating Characteristic(ROC) curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity.

```
y_pred_proba = logreg.predict_proba(X_test)[::,1]

fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)

auc = metrics.roc_auc_score(y_test, y_pred_proba)

plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)

plt.show()
```

AUC score for the case is 0.86. AUC score 1 represents perfect classifier, and 0.5 represents a worthless classifier

The Random Forests Algorithm

Let's understand the algorithm in layman's terms. Suppose you want to go on a trip and you would like to travel to a place which you will enjoy.

So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends.

Let's suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip.

In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far.

The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forests algorithm.

It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

How does the algorithm work?

It works in four steps:

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.
4. Select the prediction result with the most votes as the final prediction.

Advantages:

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages:

- Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.

- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Finding important features

Random forests also offers a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

This score will help you choose the most important features and drop the least important ones for model building.

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

Random Forests vs Decision Trees

- Random forests is a set of multiple decision trees.
- Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- Decision trees are computationally faster.
- Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

Building a Classifier using Scikit-learn

You will be building a model on the iris flower dataset, which is a very famous classification set. It comprises the sepal length, sepal width, petal length, petal width, and type of flowers. There are three species or classes: setosa, versicolor, and virginia. You will build a model to classify the type of flower. The dataset is available in the scikit-learn library or you can download it from the UCI Machine Learning Repository.

Start by importing the datasets library from scikit-learn, and load the iris dataset with load_iris().

```
#Import scikit-learn dataset library

from sklearn import datasets

#Load dataset

iris = datasets.load_iris()
```

You can print the target and feature names, to make sure you have the right dataset, as such:

```
# print the label species(setosa, versicolor,virginica)
```

```
print(iris.target_names)

# print the names of the four features

print(iris.feature_names)

['setosa' 'versicolor' 'virginica']

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

It's a good idea to always explore your data a bit, so you know what you're working with. Here, you can see the first five rows of the dataset are printed, as well as the target variable for the whole dataset.

```
# print the iris data (top 5 records)

print(iris.data[0:5])

# print the iris labels (0:setosa, 1:versicolor, 2:virginica)

print(iris.target)

[[ 5.1 3.5 1.4 0.2]

[ 4.9 3. 1.4 0.2]

[ 4.7 3.2 1.3 0.2]

[ 4.6 3.1 1.5 0.2]

[ 5. 3.6 1.4 0.2]]
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

2 2]
```

Here, you can create a DataFrame of the iris dataset the following way.

```
# Creating a DataFrame of given iris dataset.

import pandas as pd

data=pd.DataFrame({

'sepal length':iris.data[:,0],

'sepal width':iris.data[:,1],

'petal length':iris.data[:,2],

'petal width':iris.data[:,3],

'species':iris.target

})

data.head()
```

petal length

petal width

sepal length

sepal width

species

0

1.4

0.2

5.1

3.5

0

1

1.4

0.2

4.9

3.0

0

2

1.3

0.2

4.7

3.2

0

3

1.5

0.2

4.6

3.1

0

4

1.4

0.2

5.0

3.6

0

First, you separate the columns into dependent and independent variables (or features and labels). Then you split those variables into a training and test set.

```
# Import train_test_split function

from sklearn.model_selection import train_test_split

X=data[['sepal length', 'sepal width', 'petal length', 'petal width']] # Features

y=data['species'] # Labels

# Split dataset into training set and test set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
```

After splitting, you will train the model on the training set and perform predictions on the test set.

```
#Import Random Forest Model

from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier

clf=RandomForestClassifier(n_estimators=100)
```

#Train the model using the training sets y_pred=clf.predict(X_test)

clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)

After training, check the accuracy using actual and predicted values.

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

('Accuracy:', 0.93333333333333335)

You can also make a prediction for a single item, for example:

- sepal length = 3
- sepal width = 5
- petal length = 4
- petal width = 2

Now you can predict which type of flower it is.

clf.predict([[3, 5, 4, 2]])

array([2])

Here, 2 indicates the flower type Virginica.

Finding Important Features in Scikit-learn

Here, you are finding important features or selecting features in the IRIS dataset. In scikit-learn, you can perform this task in the following steps:

- First, you need to create a random forests model.
- Second, use the feature importance variable to see feature importance scores.
- Third, visualize these scores using the seaborn library.

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier

clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)

clf.fit(X_train,y_train)

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',

max_depth=None, max_features='auto', max_leaf_nodes=None,

min_impurity_decrease=0.0, min_impurity_split=None,

min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,

oob_score=False, random_state=None, verbose=0,

warm_start=False)

import pandas as pd

feature_imp =
pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=
False)

feature_imp

petal width (cm) 0.458607

petal length (cm) 0.413859

sepal length (cm) 0.103600

sepal width (cm) 0.023933

dtype: float64
```

You can also visualize the feature importance. Visualizations are easy to understand and interpretable.

For visualization, you can use a combination of matplotlib and seaborn. Because seaborn is built on top of matplotlib, it offers a number of customized themes and provides additional plot types. Matplotlib is a superset of seaborn and both are equally important for good visualizations.

```
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

# Creating a bar plot

sns.barplot(x=feature_imp, y=feature_imp.index)

# Add labels to your graph

plt.xlabel('Feature Importance Score')

plt.ylabel('Features')

plt.title("Visualizing Important Features")

plt.legend()

plt.show()
```

Generating the Model on Selected Features

Here, you can remove the "sepal width" feature because it has very low importance, and select the 3 remaining features.

```
# Import train_test_split function

from sklearn.cross_validation import train_test_split

# Split dataset into features and labels

X=data[['petal length', 'petal width','sepal length']] # Removed feature "sepal length"

y=data['species']

# Split dataset into training set and test set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70,
random_state=5) # 70% training and 30% test
```

After spliting, you will generate a model on the selected training set features, perform predictions on the selected test set features, and compare actual and predicted values.

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier

clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)

clf.fit(X_train,y_train)
```

```
# prediction on test set

y_pred=clf.predict(X_test)

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

('Accuracy:', 0.9523809523809523)
```

You can see that after removing the least important features (sepal length), the accuracy increased. This is because you removed misleading data and noise, resulting in an increased accuracy. A lesser amount of features also reduces the training time.

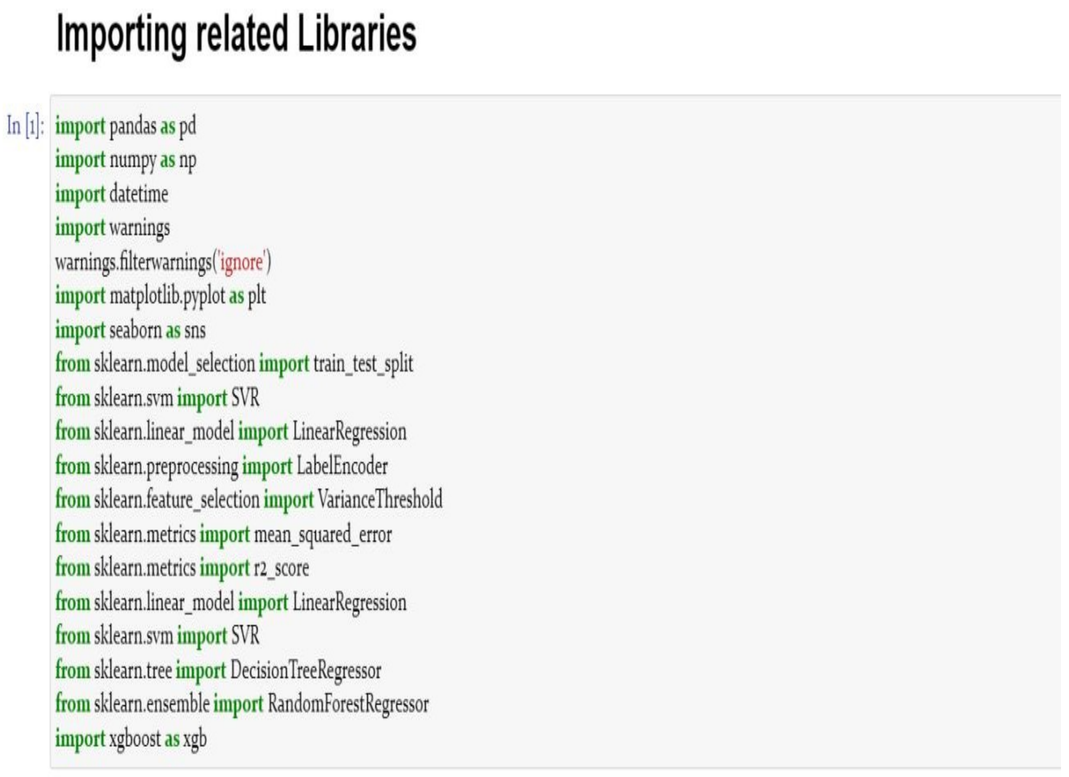
- **Performing Prediction**

The Churn was successfully predicted by using the using Logistic Regression , KNNClassifier and Random Forest.

- **Visualization in between Actual and predicted Values**

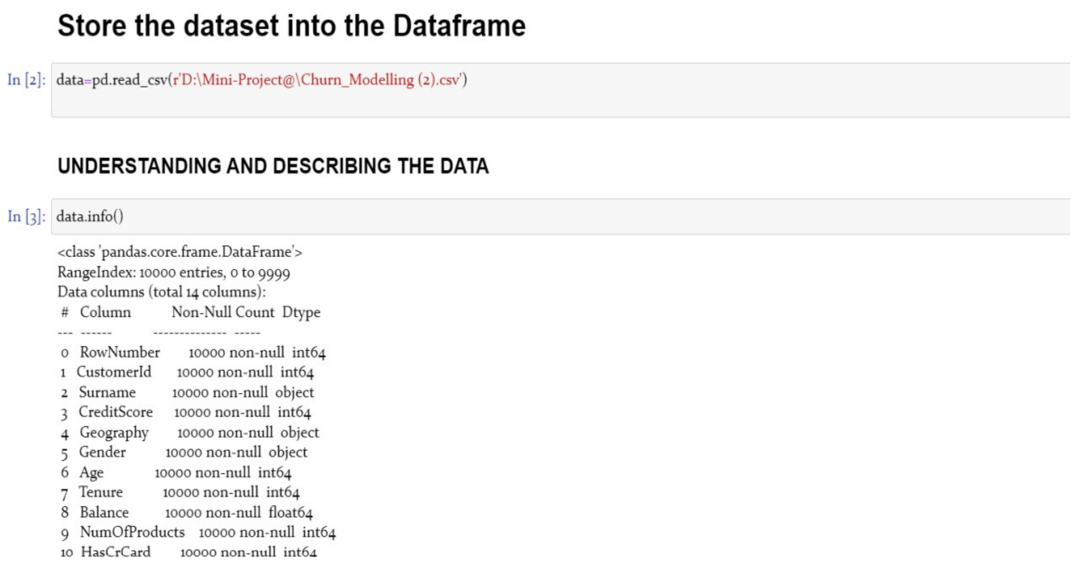
The visualization in between Actual Churn Prediction VS the Predicted Churn Value through Logistic Regression Model.

8. CODE AND OUTPUT



Importing related Libraries

```
In [1]: import pandas as pd
        import numpy as np
        import datetime
        import warnings
        warnings.filterwarnings('ignore')
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.svm import SVR
        from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import LabelEncoder
        from sklearn.feature_selection import VarianceThreshold
        from sklearn.metrics import mean_squared_error
        from sklearn.metrics import r2_score
        from sklearn.linear_model import LinearRegression
        from sklearn.svm import SVR
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import RandomForestRegressor
        import xgboost as xgb
```



Store the dataset into the Dataframe

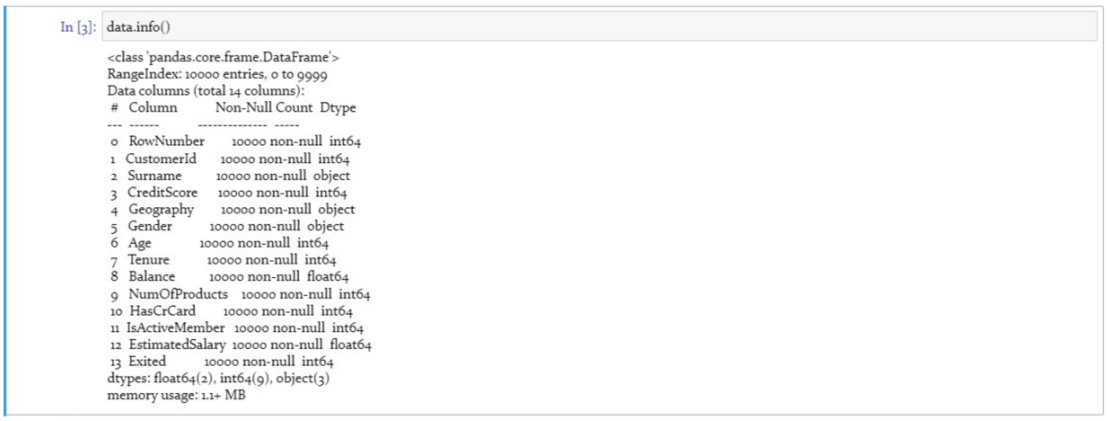
```
In [2]: data=pd.read_csv(r'D:\Mini-Project@\Churn_Modelling (2).csv')
```

UNDERSTANDING AND DESCRIBING THE DATA

```
In [3]: data.info()

<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
```

UNDERSTANDING AND DESCRIBING THE DATA



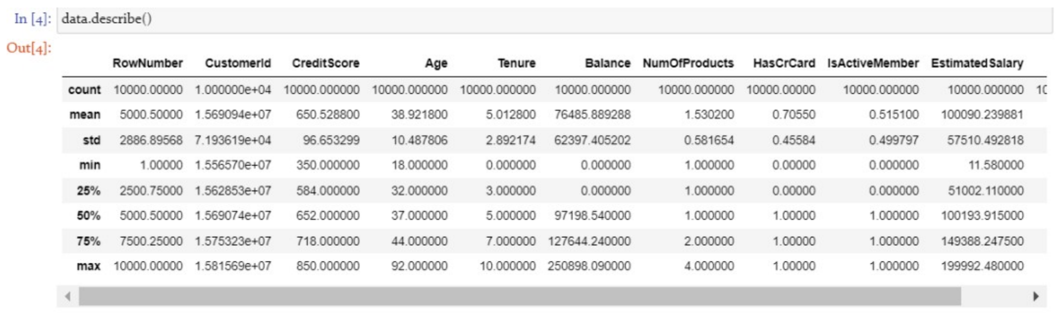
In [3]:
```
data.info()
```

```
<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
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [4]:
```
data.describe()
```

Out[4]:

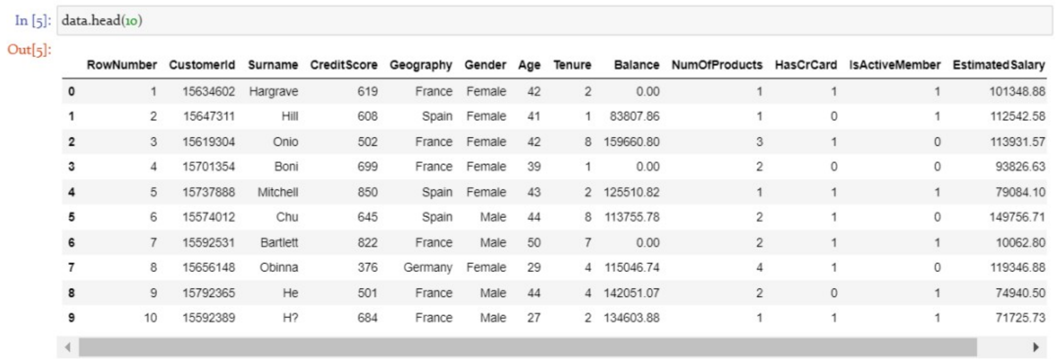
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	1C
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	



In [4]:
```
data.describe()
```

Out[4]:

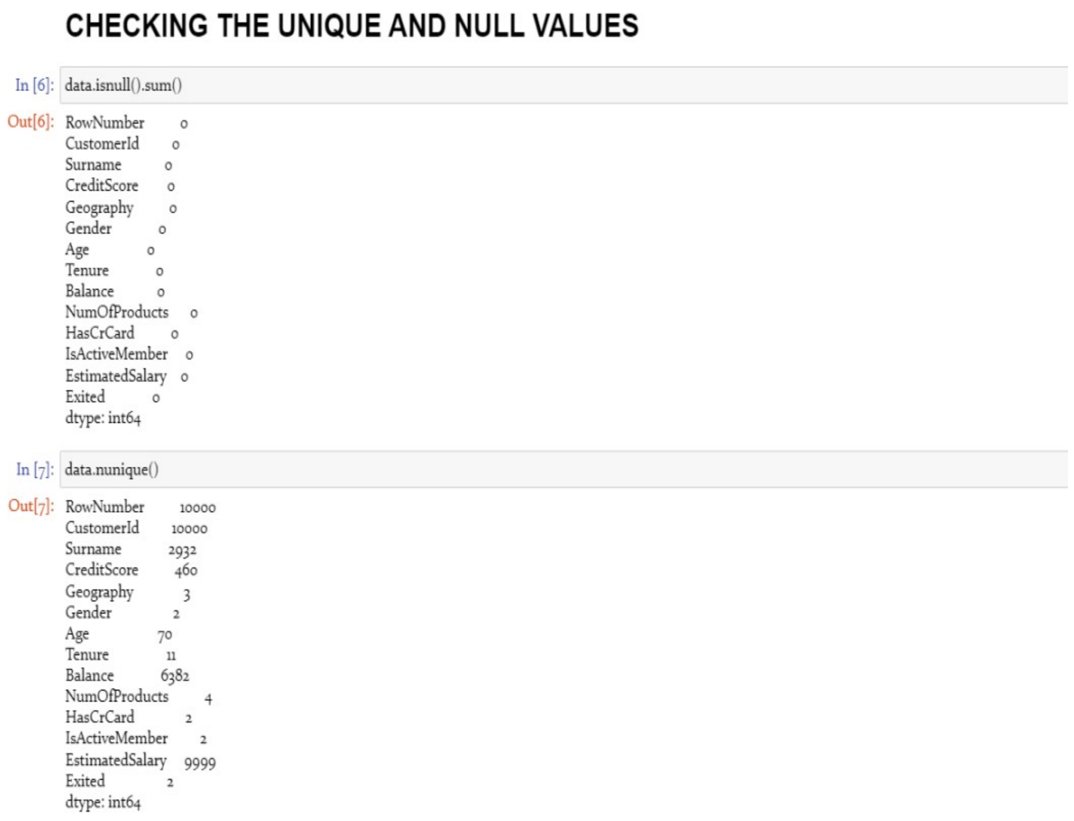
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	1C
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	



In [5]:
```
data.head(10)
```

Out[5]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73



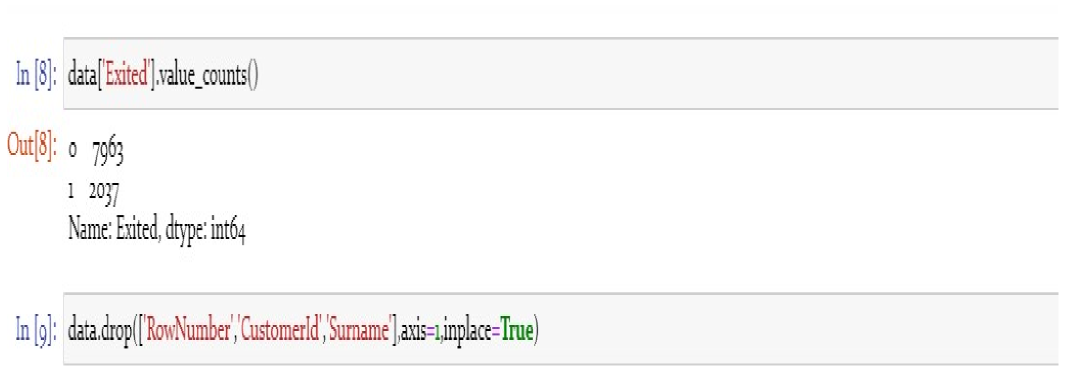
CHECKING THE UNIQUE AND NULL VALUES

```
In [6]: data.isnull().sum()

Out[6]: 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 [7]: data.nunique()

Out[7]: RowNumber        10000
        CustomerId       10000
        Surname           2932
        CreditScore        460
        Geography            3
        Gender               2
        Age                 70
        Tenure              11
        Balance           6382
        NumOfProducts        4
        HasCrCard            2
        IsActiveMember       2
        EstimatedSalary   9999
        Exited               2
        dtype: int64
```



```
In [8]: data['Exited'].value_counts()

Out[8]: 0    7963
        1    2037
        Name: Exited, dtype: int64

In [9]: data.drop(['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
```

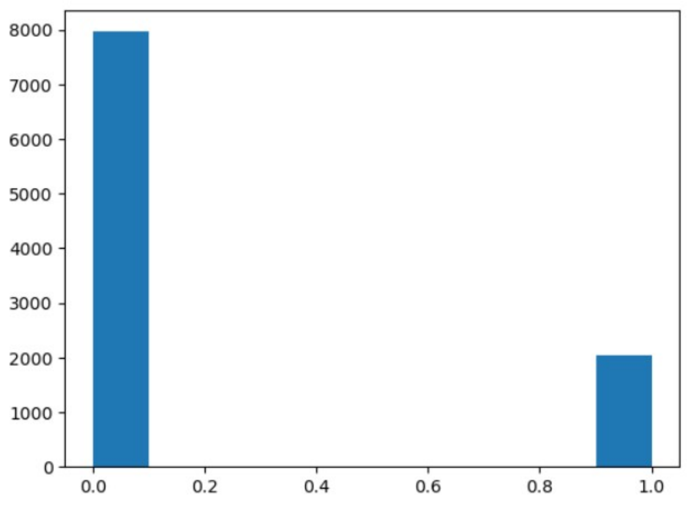
Exploratory Data Analysis:

Here our main interest is to get an understanding as to how the given attributes relate too the 'Exit' status

EDA ANALYSIS

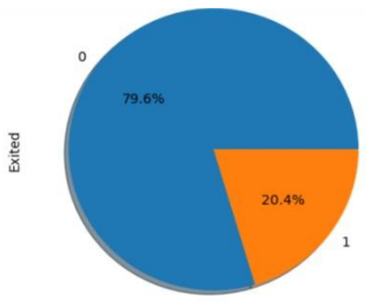
In [10]:
```
plt.hist(data['Exited'])
```

Out[10]:
```
(array([7963.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
        2037.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



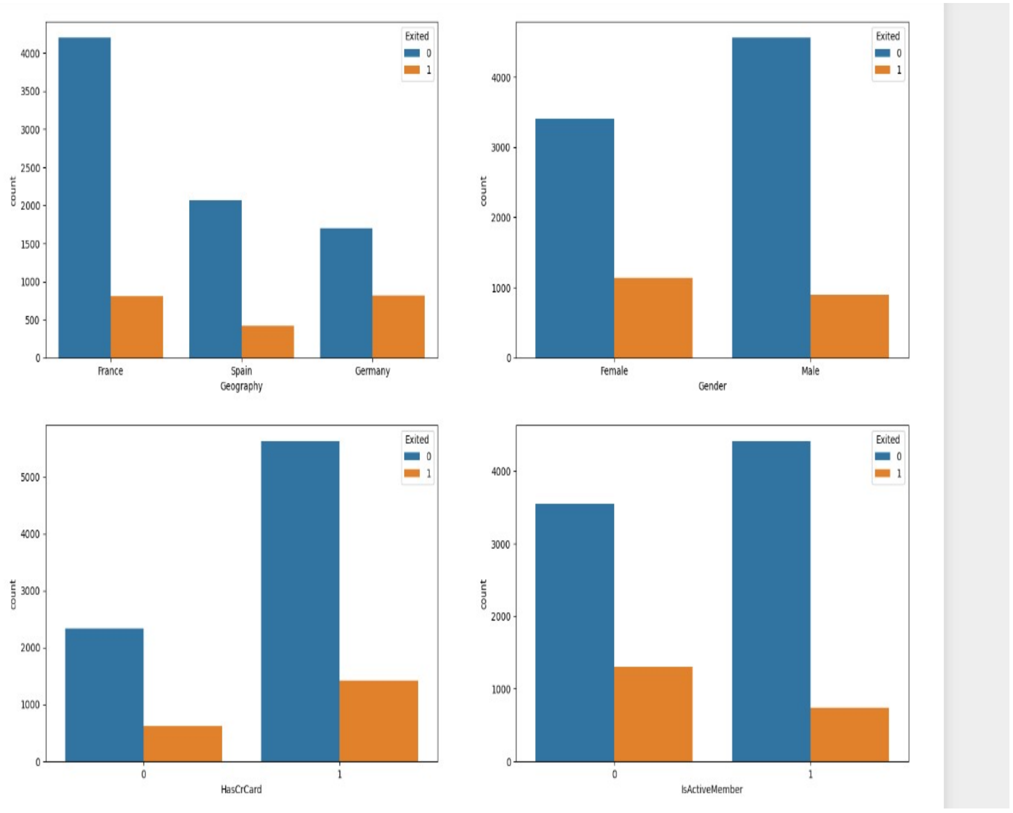
In [11]:
```
data["Exited"].value_counts().plot.pie(title = "Years of staying in the city", autopct = "%1.1f%%", shadow = True)
```

Out[11]:
```
<AxesSubplot:title={'center':'Years of staying in the city'}, ylabel='Exited'>
```



In [12]:
```
fig, axarr = plt.subplots(2, 2, figsize=(20, 12))
sns.countplot(x='Geography', hue = 'Exited',data = data, ax=axarr[0][0])
sns.countplot(x='Gender', hue = 'Exited',data = data, ax=axarr[0][1])
sns.countplot(x='HasCrCard', hue = 'Exited',data = data, ax=axarr[1][0])
sns.countplot(x='IsActiveMember', hue = 'Exited',data = data, ax=axarr[1][1])
```

Out[12]:
```
<AxesSubplot:xlabel='IsActiveMember', ylabel='count'>
```

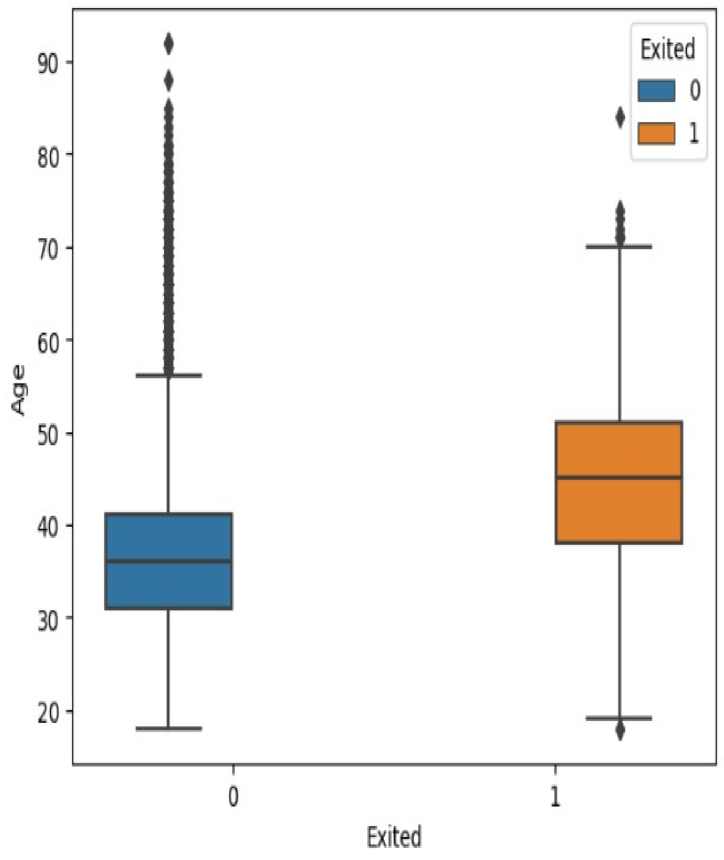


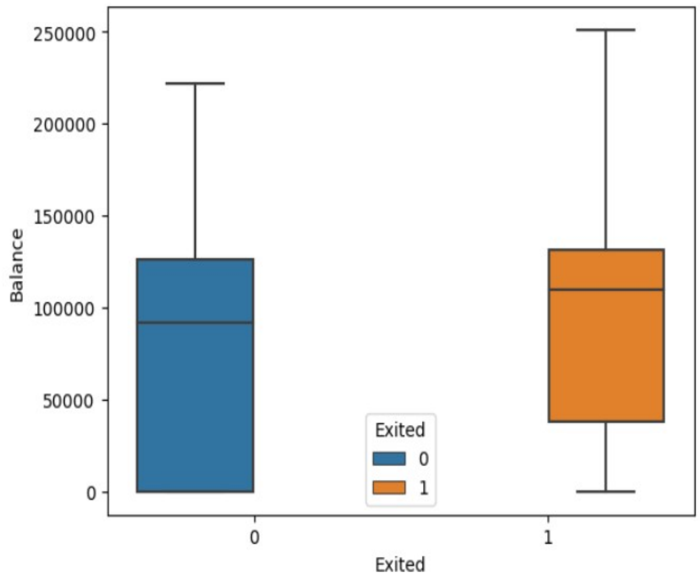
note the following:

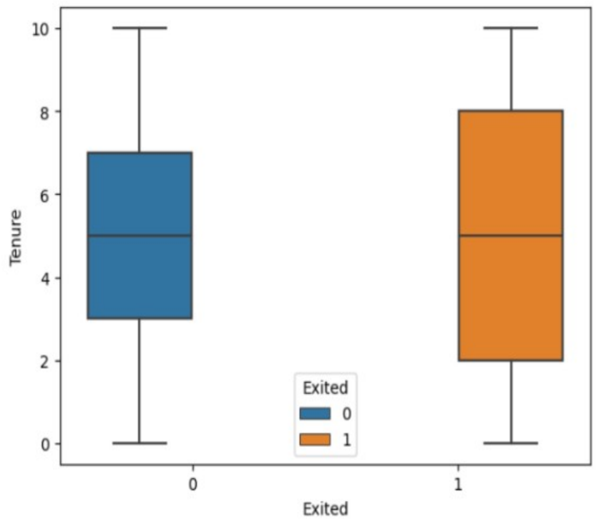
- Majority of the data is from persons from France. However, the proportion of churned customers is with inversely related to the population of customers alluding to the bank possibly having a problem (maybe not enough customer service resources allocated) in the areas where it has fewer clients.
- The proportion of female customers churning is also greater than that of male customers
- Interestingly, majority of the customers that churned are those with credit cards. Given that majority of the customers have credit cards could prove this to be just a coincidence.
- Unsurprisingly the inactive members have a greater churn. Worryingly is that the overall proportion of inactive mebers is quite high suggesting that the bank may need a program implemented to turn this group to active customers as this will definately have a positive impact on th

```
In [13]: sns.boxplot(y='Age',x = 'Exited', hue = 'Exited',data = data)
```

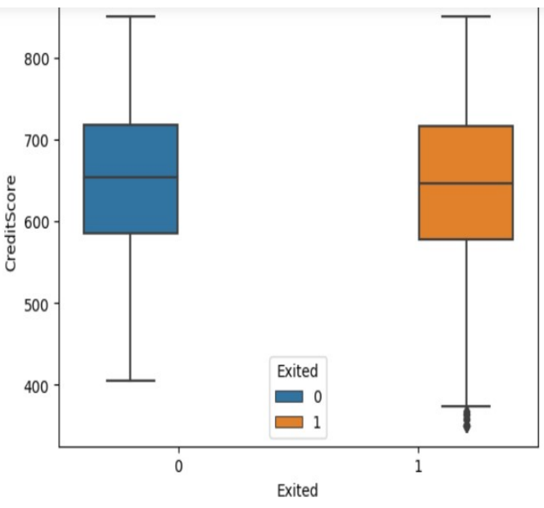
```
Out[13]: <AxesSubplot:xlabel='Exited', ylabel='Age'>
```



In [14]:
```
sns.boxplot(y='Balance',x = 'Exited', hue = 'Exited',data = data)
```
Out[14]:
```
<AxesSubplot:xlabel='Exited', ylabel='Balance'>
```


In [15]:
```
sns.boxplot(y='Tenure',x = 'Exited', hue = 'Exited',data = data)
```
Out[15]:
```
<AxesSubplot:xlabel='Exited', ylabel='Tenure'>
```


In [16]:
```
sns.boxplot(y='CreditScore',x = 'Exited', hue = 'Exited',data = data)
```
Out[16]:
```
<AxesSubplot:xlabel='Exited', ylabel='CreditScore'>
```



note the following:

- There is no significant difference in the credit score distribution between retained and churned customers.
- The older customers are churning at more than the younger ones alluding to a difference in service preference in the age categories. The bank may need to review their target market or review the strategy for retention between the different age groups
- With regard to the tenure, the clients on either extreme end (spent little time with the bank or a lot of time with the bank) are more likely to churn compared to those that are of average tenure.
- Worryingly, the bank is losing customers with significant bank balances which is likely to hit their available capital for lending.
- Neither the product nor the salary has a significant effect on the likelihood to churn.

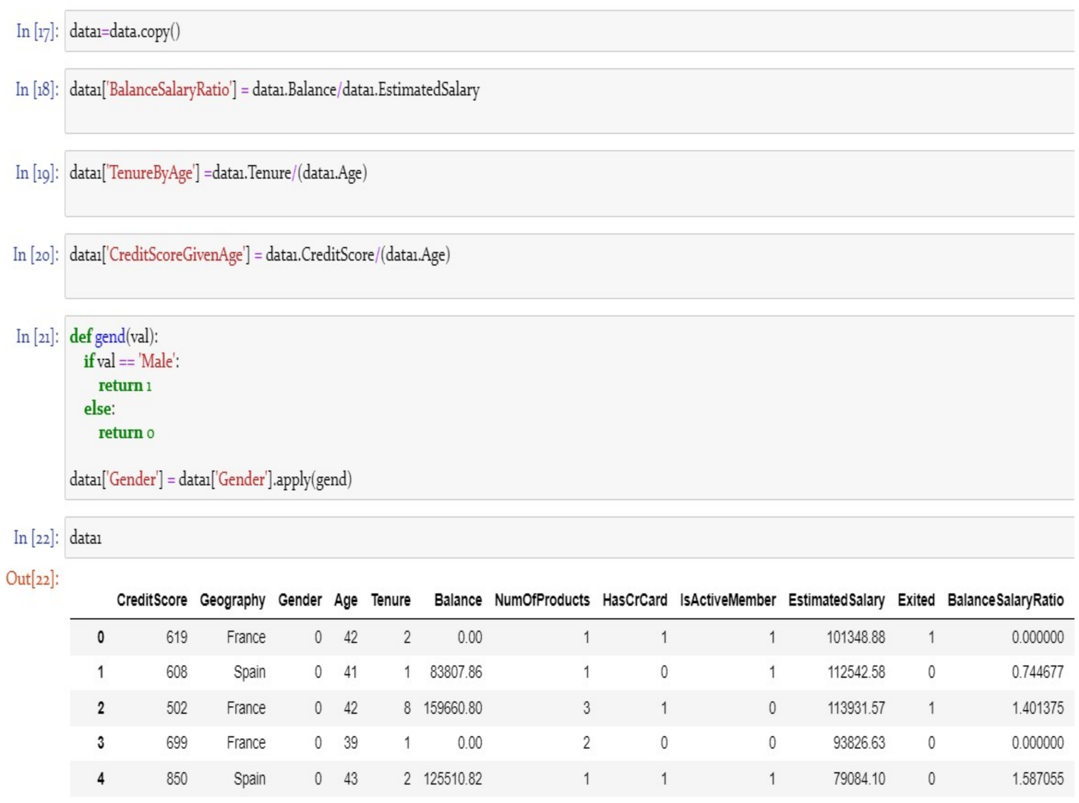
Feature engineering

We seek to add features that are likely to have an impact on the probability of churning. We first split the train and test sets

Feature engineering

We seek to add features that are likely to have an impact on the probability of churning. We first split the train and test sets

FEATURE ENGINEERING



In [17]:
```
data1=data.copy()
```

In [18]:
```
data1['BalanceSalaryRatio'] = data1.Balance/data1.EstimatedSalary
```

In [19]:
```
data1['TenureByAge'] =data1.Tenure/(data1.Age)
```

In [20]:
```
data1['CreditScoreGivenAge'] = data1.CreditScore/(data1.Age)
```

In [21]:
```
def gend(val):
    if val == 'Male':
        return 1
    else:
        return 0

data1['Gender'] = data1['Gender'].apply(gend)
```

In [22]:
```
data1
```

Out[22]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	BalanceSalaryRatio
0	619	France	0	42	2	0.00	1	1	1	101348.88	1	0.000000
1	608	Spain	0	41	1	83807.86	1	0	1	112542.58	0	0.744677
2	502	France	0	42	8	159660.80	3	1	0	113931.57	1	1.401375
3	699	France	0	39	1	0.00	2	0	0	93826.63	0	0.000000
4	850	Spain	0	43	2	125510.82	1	1	1	79084.10	0	1.587055

FEATURE SCALING

```
In [23]: from sklearn.preprocessing import MinMaxScaler,minmax_scale
         data1['Age'] = minmax_scale(data1['Age'])
         data1['EstimatedSalary'] = minmax_scale(data1['EstimatedSalary'])
         data1['Balance'] = minmax_scale(data1['Balance'])
         data1['CreditScore'] = minmax_scale(data1['CreditScore'])
         data1['BalanceSalaryRatio'] = minmax_scale(data1['BalanceSalaryRatio'])
         data1['TenureByAge'] = minmax_scale(data1['TenureByAge'])
         data1['CreditScoreGivenAge'] = minmax_scale(data1['CreditScoreGivenAge'])
```

```
In [24]: data1['Geography'].value_counts()
```

```
Out[24]: France     5014
         Germany    2509
         Spain      2477
         Name: Geography, dtype: int64
```

ONE HOT ENCODNG

```
In [25]: df_geo = data1['Geography']
         df_geo = pd.get_dummies(df_geo,drop_first=True)
         data1.drop(['Geography'],axis=1,inplace=True)
         df_final = pd.concat([data1,df_geo],axis=1)
```

```
In [26]: df_final.head(10)
```

Out[26]:

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	BalanceSalaryRatio	TenureByAge
0	0.538	0	0.324324	2	0.000000	1	1	1	0.506735	1	0.000000	0.085714
1	0.516	0	0.310811	1	0.334031	1	0	1	0.562709	0	0.000070	0.043902
2	0.304	0	0.324324	8	0.636357	3	1	0	0.569654	1	0.000132	0.342857
3	0.698	0	0.283784	1	0.000000	2	0	0	0.469120	0	0.000000	0.046154

```
In [27]: X = df_final.drop('Exited',axis=1)
         y = df_final['Exited']
```

APPLYING SMOTE

```
In [28]: from imblearn.over_sampling import SMOTE
         X_res,y_res = SMOTE().fit_resample(X,y)
```

SPLITTING OF DATASET

```
In [29]: X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.30, random_state=42)
```

CHOOSING BEST MODEL

```
In [30]: from sklearn.linear_model import LogisticRegression
         LR = LogisticRegression(random_state=0)
         LR.fit(X_train,y_train)

Out[30]: LogisticRegression(random_state=0)
```

EVALUATION OF MODEL

```
In [31]: from sklearn.metrics import classification_report,confusion_matrix
         pred1 = LR.predict(X_test)
         print(classification_report(y_test,pred1))
         print()
         print(confusion_matrix(y_test,pred1))

                   precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

           0       0.73      0.71      0.72      2426
           1       0.71      0.73      0.72      2352

    accuracy                           0.72      4778
   macro avg       0.72      0.72      0.72      4778
weighted avg       0.72      0.72      0.72      4778


[[1714  712]
 [ 637 1715]]
```

```
In [32]: from sklearn.naive_bayes import GaussianNB
         NB = GaussianNB()
         NB.fit(X_train,y_train)

Out[32]: GaussianNB()
```

```
In [33]: from sklearn.metrics import classification_report,confusion_matrix
         pred2 = NB.predict(X_test)
         print(classification_report(y_test,pred2))
         print()
         print(confusion_matrix(y_test,pred2))
```

```
              precision    recall  f1-score   support

           0       0.59      0.96      0.73      2426
           1       0.87      0.31      0.46      2352

    accuracy                           0.64      4778
   macro avg       0.73      0.63      0.59      4778
weighted avg       0.73      0.64      0.59      4778


[[2317  109]
 [1624  728]]
```

```
In [34]: from sklearn.ensemble import RandomForestClassifier
         forest = RandomForestClassifier(n_estimators=100)
         forest.fit(X_train,y_train)

Out[34]: RandomForestClassifier()
```

```
In [35]: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
         pred = forest.predict(X_test)
         print(classification_report(y_test,pred))
         print()
         print(confusion_matrix(y_test,pred))
         print(accuracy_score(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.86      0.87      2426
           1       0.86      0.88      0.87      2352

    accuracy                           0.87      4778
   macro avg       0.87      0.87      0.87      4778
weighted avg       0.87      0.87      0.87      4778


[[2094  332]
 [ 289 2063]]
0.8700293009627459
```

THE BEST MODEL IS RANDOMFOREST FOR THIS CLASSIFICATION DATASET WHICH GIVES ACCURACY AROUND 87%

Conclusion

The precision of the model on previousy unseen test data is slightly higher with regard to predicting 1's i.e. those customers that churn. However, in as much as the model has a high accuracy, it still misses about half of those who end up churning. This could be imprved by providing retraining the model with more data over time while in the meantime working with the model to save the 41% that would have churned

References

1. Ngai, E.W.; Xiu, L.; Chau, D.C. Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Syst. Appl.* **2009**, *36*, 2592–2602. [Google Scholar] [CrossRef]
2. Bahari, T.F.; Elayidom, M.S. An efficient CRM-data mining framework for the prediction of customer behaviour. *Procedia Comput. Sci.* **2015**, *46*, 725–731. [Google Scholar] [CrossRef][Green Version]
3. Ranjan, J.; Bhatnagar, V. Critical success factors for implementing CRM using data mining. *J. Knowl. Manag. Pract.* **2008**, *1*, 7. [Google Scholar] [CrossRef]