**Project Title**

**Intelligent Customer Retention: Using Machine Learning for**

**Enhanced Prediction of Telecom Customer Churn**

**Project Description**

Customer churn is often referred to as customer attrition, or customer defection which is the rate at which the customers are lost. Customer churn is a major problem and one of the most important concerns for large companies. Due to the direct effect on the revenues of the companies, especially in the telecom field, companies are seeking to develop means to predict potential customer to churn. Looking at churn, different reasons trigger customers to terminate their contracts, for example better price offers, more interesting packages, bad service experiences or change of customers’ personal situations.

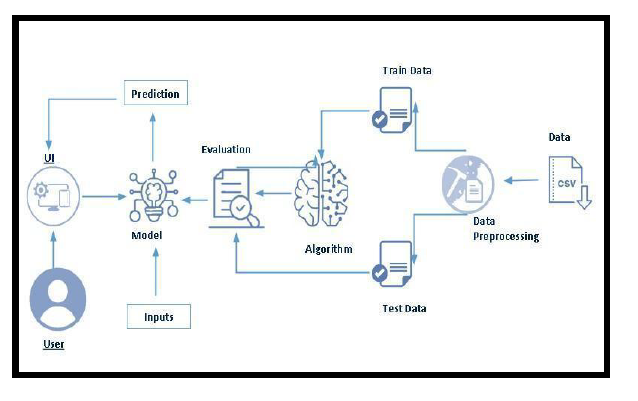
Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behaviour of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.

Telecommunication industry always suffers from a very high churn rates when one industry offers a better plan than the previous there is a high possibility of the customer churning from the present due to a better plan in such a scenario it is very difficult to avoid losses but through prediction we can keep it to a minimal level.

Telecom companies often use customer churn as a key business metrics to predict the number of

customers that will leave a telecom service provider. A machine learning model can be used to identity the probable churn customers and then makes the necessary business decisions.

**Technical Architecture:**



**Project Flow:**

● User interacts with the UI to enter the input.

● Entered input is analysed by the model which is integrated.

● Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

● Define Problem / Problem Understanding

○ Specify the business problem

○ Business requirements

○ Literature Survey

○ Social or Business Impact.

● Data Collection & Preparation

○ Collect the dataset

○ Data Preparation

● Exploratory Data Analysis

○ Descriptive statistical

○ Visual Analysis

● Model Building

○ Training the model in multiple algorithms

○ Testing the model

● Performance Testing & Hyperparameter Tuning

○ Testing model with multiple evaluation metrics

○ Comparing model accuracy before & after applying hyperparameter tuning

● Model Deployment

○ Save the best model

○ Integrate with Web Framework

● Project Demonstration & Documentation

○ Record explanation Video for project end to end solution

○ Project Documentation-Step by step project development procedure

**Project Structure**

Create a Project folder which contains files as shown below

A python file called app.py for server side scipting.

● We need the model which is saved and the saved model in this content is **churn.pkl**

**●** Templates folder which contains base.HTML file, index.HTML file, predyes.HTML , predno.HTML

file.

**●** Static folder which contains css folder which contains main.css , js folder which contains global.js ,

images folder and vendor folder.

**Milestone 1: Define Problem / Problem Understanding**

**Activity 1: Specify the business problem**

Telecom companies often use customer churn as a key business metrics to predict the number of

customers that will leave a telecom service provider. A machine learning model can be used to identity the probable churn customers and then makes the necessary business decisions.

**Activity 2: Business requirements**

The business requirements for a machine learning model to predict whether the customer will churn or not on customer information, to minimise the number of false positives (customer that predicted as loyal but churn) and false negatives (customer predicted to be churn which could have stayed loyal). Provide an explanation for the model's decision, for better decision making in order to gain more profitability.

**Activity 3: Literature Survey (Student Will Write)**

As the data is increasing daily due to digitization in the banking sector, people want to apply for loans through the internet. Machine Learning (ML), as a typical method for information investigation, has gotten more consideration increasingly. Individuals of various businesses are utilising ML calculations to take care of the issues dependent on their industry information. Telecom companies often use customer churn as a key business metrics to predict the number of customers that will leave a telecom service provider. A machine learning model can be used to identity the probable churn customers and then makes the necessary business decisions.

**Activity 4: Social or Business Impact.**

Social Impact:- Proposed model can help improve the overall customer experience and

service quality. Companies can also make better decisions about how to retain their customers.

Business Model/ Impact :- This product can generate revenue using a product based model, where the system can be sold as a product to the telecom companies. This product can also be used for subscription based model.

**Milestone 2: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

**Activity 1: Collect the dataset**

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI

repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please

refer to the link given below to download the dataset.

Link: https://www.kaggle.com/shrutimechlearn/churn-modelling

As the dataset is downloaded. Let us read and understand the data properly with the help

of some visualisation techniques and some analysing techniques.

**Note:** There are a number of techniques for understanding the data. But here we have

used some of it. In an additional way, you can use multiple techniques.

**Activity 1.1: Importing the libraries**

Import the necessary libraries as shown in the following table.

|  |
| --- |
| #import necessary libraries  import pandas as pd  import numpy as np  import pickle  import matplotlib.pyplot as plt  %matplotlib inline  import seaborn as sns  import sklearn  from sklearn.preprocessing import LabelEncoder, OneHotEncoder  from sklearn.linear\_model import LogisticRegression  from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import RandomForestClassifier  from sklearn.neighbors import KNeighborsClassifier  from sklearn.svm import SVC  from sklearn.model\_selection import RandomizedSearchCV  #import imblearn  #from imblearn.over\_sampling import SMOTE  #from sklearn.model\_selection import train\_test\_split  #from sklearn.preprocessing import StandardScaler  #from sklearn.metrics import accuracy\_score,classification\_report,confusion\_matrix,f1\_score |

**Activity 1.2: Read the Dataset**

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset

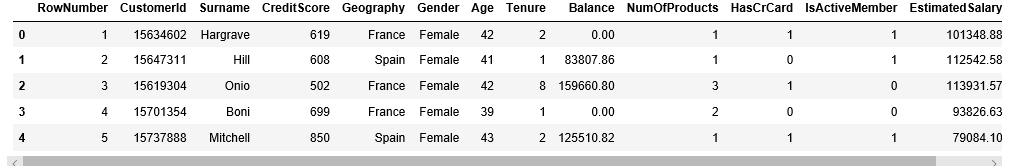
with the help of pandas.

In pandas we have a function called read\_csv() to read the dataset. As a parameter we

have to give the directory of the csv file.

data=pd.read\_csv("D:\\NMDS\\Churn\_Modelling.csv")

data.head()



**Activity 2: Data Preparation**

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might

have so much randomness so we need to clean the dataset properly in order to fetch good

results. This activity includes the following steps.

● Handling missing values

● Handling categorical data

● Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine

learning. Depending on the condition of your dataset, you may or may not have to go

through all these steps.

**Activity 2.1: Handling missing values**

● Let’s find the shape of our dataset first. To find the shape of our data, the df.shape

method is used. To find the data type, df.info() function is used.

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

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

#data.TotalCharges=pd.to\_numeric(data.TotalCharges,error='coerce')

**data.isnull().any()**

RowNumber False

CustomerId False

Surname False

CreditScore False

Geography False

Gender False

Age False

Tenure False

Balance False

NumOfProducts False

HasCrCard False

IsActiveMember False

EstimatedSalary False

Exited False

dtype: bool

From the above code of analysis, we can infer that column TotalCharges is having the missing values, we need to treat them in a required way.

**data.isnull().sum()**

RowNumber 0

CustomerId 0

Surname 0

CreditScore 0

Geography 0

Gender 0

Age 0

Tenure 0

Balance 0

NumOfProducts 0

HasCrCard 0

IsActiveMember 0

EstimatedSalary 0

Exited 0

dtype: int64

**Activity 2.2: Handling Categorical Values**

As we can see our dataset has categorical data we must convert the categorical data to

integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques.

There are several techniques but in our project we are using manual encoding with the

help of list comprehension.

**Label Encoding**.

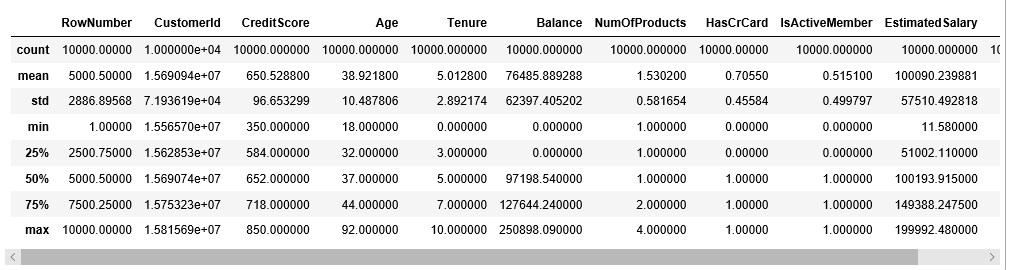
Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

**Milestone 3: Exploratory Data Analysis**

**Activity 1: Descriptive statistical**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

**data.describe()**

****

**Activity 2: Visual analysis**

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

**Activity 2.1: Univariate analysis**

In simple words, univariate analysis understands the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

● The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

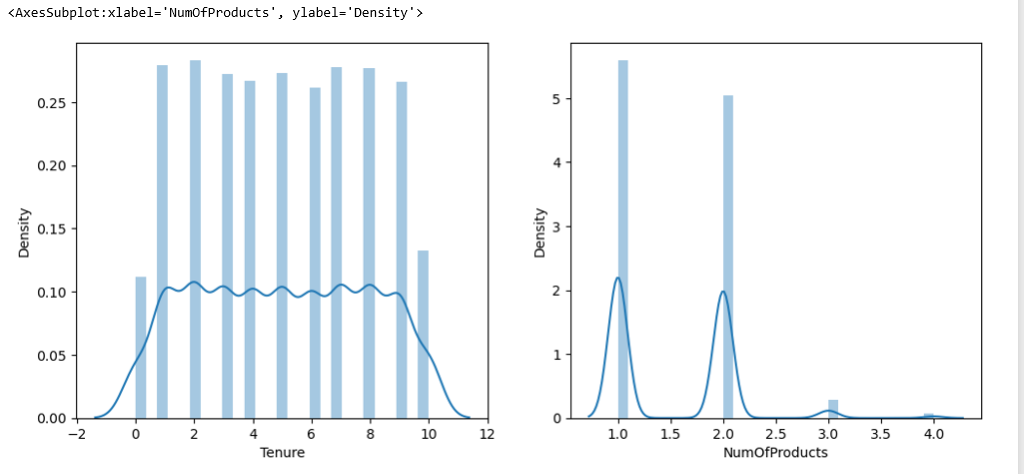
**plt.figure(figsize=(12,5))**

**plt.subplot(1,2,1)**

**sns.distplot(data['Tenure'])**

**plt.subplot(1,2,2)**

**sns.distplot(data['NumOfProducts'])**

****

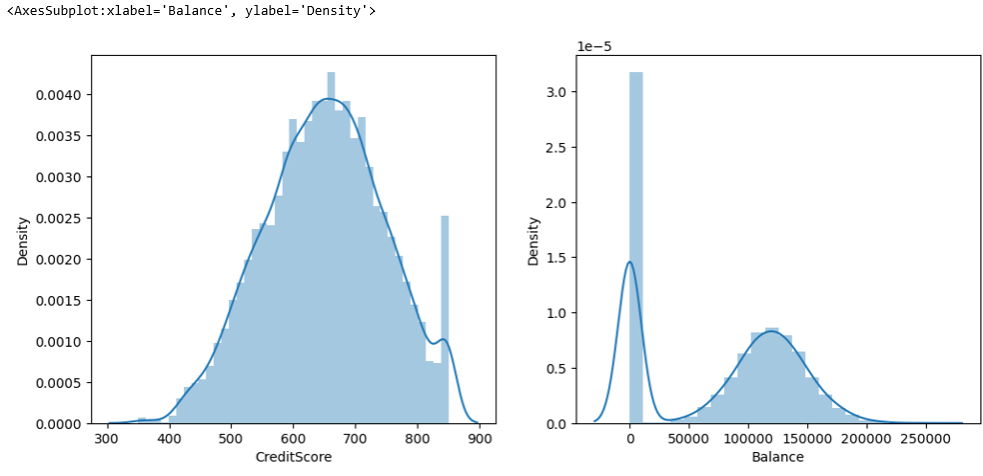
**plt.figure(figsize=(12,5))**

**plt.subplot(1,2,1)**

**sns.distplot(data['CreditScore'])**

**plt.subplot(1,2,2)**

**sns.distplot(data['Balance'])**

****

In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.

● From the plot we came to know, Applicants income is skewed towards left side,

where as credit history is categorical with 1.0 and 0.0

**Countplot :-**

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

From the graph we can infer that , gender and education is a categorical variables with 2 categories , from gender column we can infer that 0-category is having more weightage than category-1,while education with 0,it means no education is a underclass when compared with category -1, which means educated .

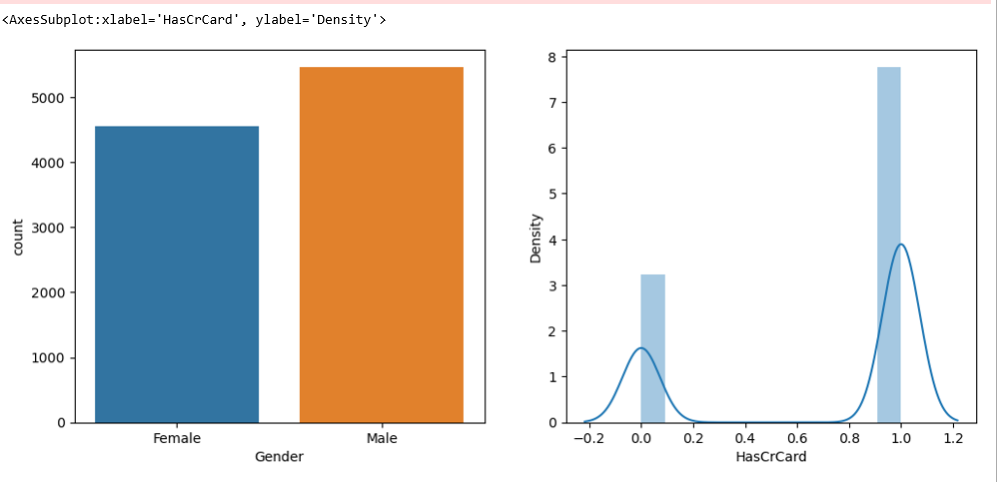
**plt.figure(figsize=(12,5))**

**plt.subplot(1,2,1)**

**sns.countplot(data['Gender'])**

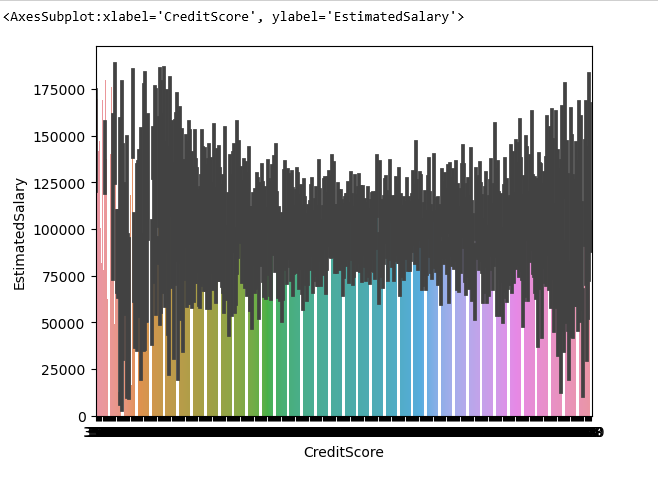
**plt.subplot(1,2,2)**

**sns.distplot(data['HasCrCard'])**

****

**Activity 2.2: Bivariate analysis**

**sns.barplot(x="CreditScore",y="EstimatedSalary",data=data)**

****

From the above graph we can infer the analysis such as

● Segmenting the gender column and married column based on bar graphs

● Segmenting the Education and Self-employed based on bar graphs ,for drawing insights such as educated people are employed.

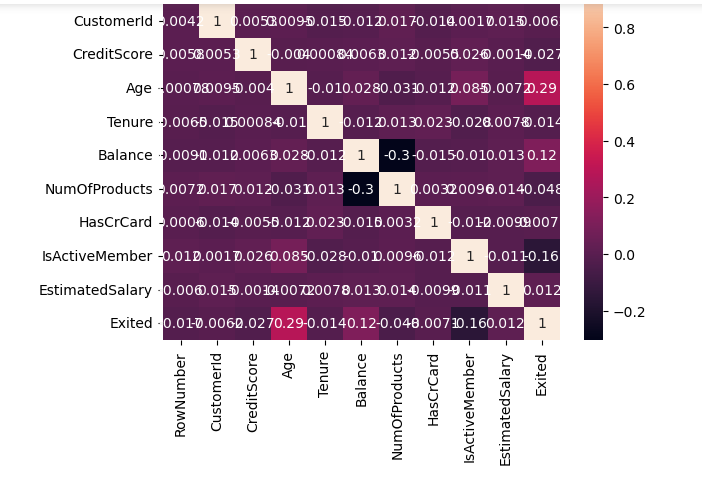
● Loan amount term based on the property area of a person holding

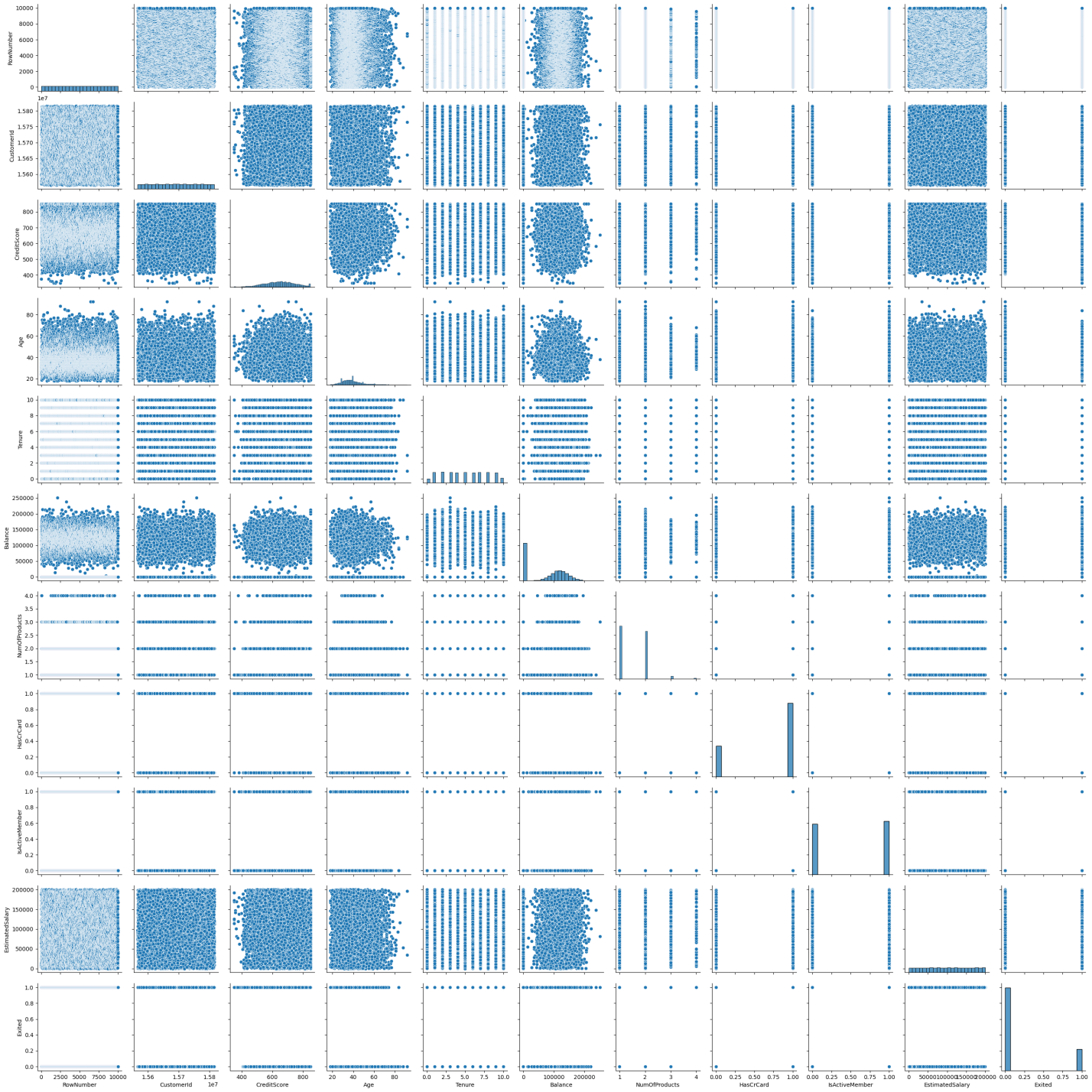
**Activity 2.3: Multivariate analysis**

In simple words, multivariate analysis is to find the relation between multiple

features. Here we have used a swarm plot from the seaborn package.

**sns.heatmap(data.corr(),annot=True)**





From the above graph we are plotting the relationship between the Gender, applicants

income and loan status of the person.

Now, the code would be normalising the data by scaling it to have a similar range of

values, and then splitting that data into a training set and a test set for training the model

and testing its performance, respectively.

**Splitting data into train and test**

Now let’s split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target

variable. And on y target variable is passed. For splitting training and testing data we are

using the train\_test\_split() function from sklearn. As parameters, we are passing x, y,

test\_size, random\_state.

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_resamble,y\_resamble,test\_size=0.2,random\_state=0)

**Scaling the Data**

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction.

Models such as KNN, Logistic regression need scaled data, as they follow distance based

method and Gradient Descent concept.

from sklearn.preprocessing import standarscaler

sc= standardscaler()

x\_train=sc.fit\_transform(x\_train)

x\_test=sc.fit\_transform(x\_test)

We will perform scaling only on the input values.Once the dataset is scaled, it will be

converted into an array and we need to convert it back to a dataframe.

**Milestone 4: Model Building**

**Activity 1: Training the model in multiple algorithms**

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

**Activity 1.2: Logistic Regression Model**

Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

#importing and building the Decision tree model

def logreg(x\_train,x\_test,y\_train,y\_test):

lr=LogisticRegression(random\_state=0)

lr.fit(x\_train,y\_train)

y\_lr\_tr=lr.predict(x\_train)

print(accuracy\_score(y\_lr\_tr,y\_))

ypred\_lr=lr.predict(x\_test)

print(accuracy\_score(ypred\_lr,y\_test))

print("\*\*\*Logistic Regression\*\*\*")

print("confusion\_Matrix")

print(confusion\_matrix(y\_test,ypred\_lr))

print("classification Report")

print(classification\_report(y\_test,ypred\_lr))

#printing the train accuracy and test accuracy respectively

logreg(x\_train,x\_test,y\_train,y\_test)

Activity 1.2: Decision tree model

A function named decisionTree is created and train and test data are passed as the

parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training

data is passed to the model with the .fit() function. Test data is predicted with .predict()

function and saved in a new variable. For evaluating the model, a confusion matrix and

classification report is done.

#importing and building the Decision tree model

def decisionTree(x\_train,x\_test,y\_train,y\_test):

dtc=DecisionTreeClassifier(criterion="entropy",random\_state=0)

dec.fit(x\_train,y\_train)

y\_dt\_tr=dtc.predict(x\_train)

print(accuracy\_score(y\_dt\_tr,y\_train))

ypred\_dt=dtc.predict(x\_test)

print(accuracy\_score(ypred\_dt,y\_test))

print("\*\*\*Decision Tree\*\*\*")

print("confusion\_matrix")

print(confusion\_matrix(y\_test,ypred\_dt))

print("Classification Report")

print(classification\_report(y\_test,ypred\_dt))

#printing the train accuracy and test accuracy respecftively

decisionTree(x\_train,x\_test,y\_train,y\_test)