**Naan Mudhalvan**

**Data Analytics using IBM Cognos**

Phase 5 : Project Documentation and Submission

**Project Title :** Customer Churn Prediction

**Project Abstract :** Customer churn poses a significant challenge for businesses across industries, leading to revenue loss and decreased profitability. This project aims to develop an effective customer churn prediction model for [Your Company's Name] using advanced machine learning techniques. By leveraging historical customer data, including demographic information, purchase history, and customer interactions, our model employs predictive analytics to identify customers at risk of churning.

**Project Objectives :**

To develop a robust and precise machine learning model that can accurately predict customer churn. This entails leveraging historical customer data encompassing demographics, transaction history, and interactions to create a predictive tool.

**Steps involved and done :**

**Importing necessary libraries :**

Libraries that are required for developing the project are imported

Step 1

**Data Collection and preprocessing:**

Collection of data from reliable source and preprocessing of data

**Finding the most accurate model and predicting individual customer churn :**

Find out the most accurate model by comparing their accuracies, and at last, predict the churn probability of an individual customer.

**Utilizing various Machine Learning algorithms to predict customer churn :**

Feed the training data to various ML classification models, observe the accuracy in their outputs.

**Visualization :**

Visualizing customer churn through various tools like Bar charts and plots.

**Code :**

The development of the project has been done in three phases. We have included the Code that we developed in order to achieve the desired result below. The dataset used is Telco customer churn dataset with 7043 Entries and 21 attributes.

**PART – 1 : Importing necessary libraries and Data Preprocessing :**

**#Import necessary libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.offline as po

import plotly.graph\_objs as go

**#Import Customer Churn Dataset**

churn\_dataset = pd.read\_csv("Telco-Customer-Churn.csv")

print(churn\_dataset.shape)

**#Convert String values (Yes and No) of Churn column to 1 and 0**

churn\_dataset.loc[churn\_dataset.Churn == 'No','Churn'] = 0

churn\_dataset.loc[churn\_dataset.Churn == 'Yes','Churn'] = 1

**#Convert 'No Internet Service' to 'No' for the below mentioned columns**

cols = ['OnlineBackup','StreamingMovies','DeviceProtection','TechSupport','OnlineSecurity','StreamingTV']

for i in cols :

churn\_dataset[i] = churn\_dataset[i].replace({'No internet service' : 'No'})

**#Replace all the spaces with null values**

churn\_dataset['TotalCharges'] = churn\_dataset['TotalCharges'].replace(" ",np.nan)

**#Drop null values of 'Total Charges' feature**

churn\_dataset = churn\_dataset[churn\_dataset['TotalCharges'].notnull()]

churn\_dataset = churn\_dataset.reset\_index()[churn\_dataset.columns]

**#Convert 'Total Charges' column values to float data type**

churn\_dataset['TotalCharges'] = churn\_dataset['TotalCharges'].astype(float)

print(churn\_dataset['Churn'].value\_counts().values)

**PART – 2 : Visualizing Customer Churn under various categories**

**#Visualize Total Customer Churn**

plot\_by\_churn\_labels = churn\_dataset["Churn"].value\_counts().keys().tolist()

plot\_by\_churn\_values = churn\_dataset["Churn"].value\_counts().values.tolist()

plot\_data = [

go.Pie(labels = plot\_by\_churn\_labels,

values = plot\_by\_churn\_values,

marker = dict(colors=['Teal','Grey'],

line=dict(color="white",

width=1.5)),

rotation=90,

hoverinfo="label+value+text",

hole= .6)

]

plot\_layout = go.Layout(dict(title = "Customer Churn",

plot\_bgcolor="rgb(243,243,243)",

paper\_bgcolor="rgb(243,243,243)",))

fig=go.Figure(data=plot\_data,layout=plot\_layout)

po.iplot(fig)

**#Visualize churn rate by Gender :**

plot\_by\_gender = churn\_dataset.groupby('gender').Churn.mean().reset\_index()

plot\_data = [

go.Bar(

x = plot\_by\_gender['gender'],

y = plot\_by\_gender['Churn'],

width = [0.3,0.3],

marker = dict(

color = ['orange','green'])

)

]

plot\_layout = go.Layout(

xaxis = {"type" : "category"},

yaxis = {"title" : "Churn Rate"},

title = 'Churn Rate by Gender',

plot\_bgcolor = 'rgb(243,243,243)',

paper\_bgcolor = 'rgb(243,243,243)',

)

fig = go.Figure(data = plot\_data,layout=plot\_layout)

po.iplot(fig)

**# Visualize churn rate by Tech Support**

plot\_by\_techsupport = churn\_dataset.groupby('TechSupport').Churn.mean().reset\_index()

plot\_data = [

go.Bar(

x = plot\_by\_techsupport['TechSupport'],

y = plot\_by\_techsupport['Churn'],

width = [0.3, 0.3, 0.3],

marker = dict(

color = ['orange','green', 'teal'])

)

]

plot\_layout = go.Layout(

xaxis = {"type": "category"},

yaxis = {"title" : "Churn Rate"},

title = 'Churn Rate by Tech Support',

plot\_bgcolor = "rgb(243,243,243)",

paper\_bgcolor = 'rgb(243,243,243)'

)

fig = go.Figure(data = plot\_data, layout = plot\_layout)

po.iplot(fig)

**# Visualize Churn Rate by Internet Services**

plot\_by\_internet\_service **=** churn\_dataset**.**groupby('InternetService')**.**Churn**.**mean()**.**reset\_index()

plot\_data **=** [

go**.**Bar(

x**=**plot\_by\_internet\_service['InternetService'],

y**=**plot\_by\_internet\_service['Churn'],

width **=** [0.3, 0.3, 0.3],

marker**=**dict(

color**=**['orange', 'green', 'teal'])

)

]

plot\_layout **=** go**.**Layout(

xaxis**=**{"type": "category"},

yaxis**=**{"title": "Churn Rate"},

title**=**'Churn Rate by Internet Service',

plot\_bgcolor **=** 'rgb(243,243,243)',

paper\_bgcolor **=** 'rgb(243,243,243)',

)

fig **=** go**.**Figure(data**=**plot\_data, layout**=**plot\_layout)

po**.**iplot(fig)

**# Visualize Churn Rate by Contract Duration**

plot\_by\_contract **=** churn\_dataset**.**groupby('Contract')**.**Churn**.**mean()**.**reset\_index()

plot\_data **=** [

go**.**Bar(

x**=**plot\_by\_contract['Contract'],

y**=**plot\_by\_contract['Churn'],

width **=** [0.3, 0.3,0.3],

marker**=**dict(

color**=**['orange', 'green','teal'])

)

]

plot\_layout **=** go**.**Layout(

xaxis**=**{"type": "category"},

yaxis**=**{"title": "Churn Rate"},

title**=**'Churn Rate by Contract Duration',

plot\_bgcolor **=** 'rgb(243,243,243)',

paper\_bgcolor **=** 'rgb(243,243,243)',

)

fig **=** go**.**Figure(data**=**plot\_data, layout**=**plot\_layout)

po**.**iplot(fig)

**# Visualize Relation between Tenure & Churn rate**

plot\_by\_tenure **=** churn\_dataset**.**groupby('tenure')**.**Churn**.**mean()**.**reset\_index()

plot\_data **=** [

go**.**Scatter(

x**=**plot\_by\_tenure['tenure'],

y**=**plot\_by\_tenure['Churn'],

mode**=**'markers',

name**=**'Low',

marker**=** dict(size**=** 5,

line**=** dict(width**=**0.8),

color**=** 'green'

),

)

]

plot\_layout **=** go**.**Layout(

yaxis**=** {'title': "Churn Rate"},

xaxis**=** {'title': "Tenure"},

title**=**'Relation between Tenure & Churn rate',

plot\_bgcolor **=** "rgb(243,243,243)",

paper\_bgcolor **=** "rgb(243,243,243)",

)

fig **=** go**.**Figure(data**=**plot\_data, layout**=**plot\_layout)

po**.**iplot(fig)

**PART – 3 : Machine Learning Algorithms to predict customer churn**

**#Perform One Hot Encoding using get\_dummies method**

churn\_dataset **=** pd**.**get\_dummies(churn\_dataset, columns **=** ['Contract','Dependents','DeviceProtection','gender',

'InternetService','MultipleLines','OnlineBackup',

'OnlineSecurity','PaperlessBilling','Partner',

'PaymentMethod','PhoneService','SeniorCitizen',

'StreamingMovies','StreamingTV','TechSupport'],

drop\_first**=True**)

**#Perform Feature Scaling and One Hot Encoding**

**from** sklearn.preprocessing **import** StandardScaler

**#PerformFeature Scaling on 'tenure', 'MonthlyCharges', 'TotalCharges' in order to bring them on same scale**

standardScaler **=** StandardScaler()

columns\_for\_ft\_scaling **=** ['tenure', 'MonthlyCharges', 'TotalCharges']

**#Apply the feature scaling operation on dataset using fit\_transform() method**

churn\_dataset[columns\_for\_ft\_scaling] **=** standardScaler**.**fit\_transform(churn\_dataset[columns\_for\_ft\_scaling])

**# See subset of values**

churn\_dataset**.**head()

**#Number of columns increased and have suffixes attached, as a result of get\_dummies method.**

churn\_dataset**.**columns

**#Create Feature variable X and Target variable y**

y **=** churn\_dataset['Churn']

X **=** churn\_dataset**.**drop(['Churn','customerID'], axis **=** 1)

**#Split the data into training set (70%) and test set (30%)**

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

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.30, random\_state **=** 50)

**# Machine Learning classification model libraries**

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.svm **import** SVC

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn **import** metrics

**#Fit the logistic Regression Model**

logmodel **=** LogisticRegression(random\_state**=**50)

logmodel**.**fit(X\_train,y\_train)

**#Predict the value for new, unseen data**

pred **=** logmodel**.**predict(X\_test)

**# Find Accuracy using accuracy\_score method**

logmodel\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, pred) **\*** 100, 2)

**#Fit the Support Vector Machine Model**

svcmodel **=** SVC(kernel**=**'linear', random\_state**=**50, probability**=True**)

svcmodel**.**fit(X\_train,y\_train)

**#Predict the value for new, unseen data**

svc\_pred **=** svcmodel**.**predict(X\_test)

**# Find Accuracy using accuracy\_score method**

svc\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, svc\_pred) **\*** 100, 2)

**#Fit the K-Nearest Neighbor Model**

**from** sklearn.neighbors **import** KNeighborsClassifier

knnmodel **=** KNeighborsClassifier(n\_neighbors**=**5, metric**=**'minkowski', p**=**2) *#p=2 represents Euclidean distance, p=1 represents Manhattan Distance*

knnmodel**.**fit(X\_train, y\_train)

**#Predict the value for new, unseen data**

knn\_pred **=** knnmodel**.**predict(X\_test)

**# Find Accuracy using accuracy\_score method**

knn\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, knn\_pred) **\*** 100, 2)

**#Fit the Decision Tree Classification Model**

**from** sklearn.tree **import** DecisionTreeClassifier

dtmodel **=** DecisionTreeClassifier(criterion **=** "gini", random\_state **=** 50)

dtmodel**.**fit(X\_train, y\_train)

**#Predict the value for new, unseen data**

dt\_pred **=** dtmodel**.**predict(X\_test)

**# Find Accuracy using accuracy\_score method**

dt\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, dt\_pred) **\*** 100, 2)

**#Fit the Random Forest Classification Model**

**from** sklearn.ensemble **import** RandomForestClassifier

rfmodel **=** RandomForestClassifier(n\_estimators **=** 100, criterion **=** 'entropy', random\_state **=** 0)

rfmodel**.**fit(X\_train, y\_train)

**#Predict the value for new, unseen data**

rf\_pred **=** rfmodel**.**predict(X\_test)

**# Find Accuracy using accuracy\_score method**

rf\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, rf\_pred) **\*** 100, 2)

**# Compare Several models according to their Accuracies**

Model\_Comparison **=** pd**.**DataFrame({

'Model': ['Logistic Regression', 'Support Vector Machine', 'K-Nearest Neighbor',

'Decision Tree', 'Random Forest'],

'Score': [logmodel\_accuracy, svc\_accuracy, knn\_accuracy,

dt\_accuracy, rf\_accuracy]})

Model\_Comparison\_df **=** Model\_Comparison**.**sort\_values(by**=**'Score', ascending**=False**)

Model\_Comparison\_df **=** Model\_Comparison\_df**.**set\_index('Score')

Model\_Comparison\_df**.**reset\_index()

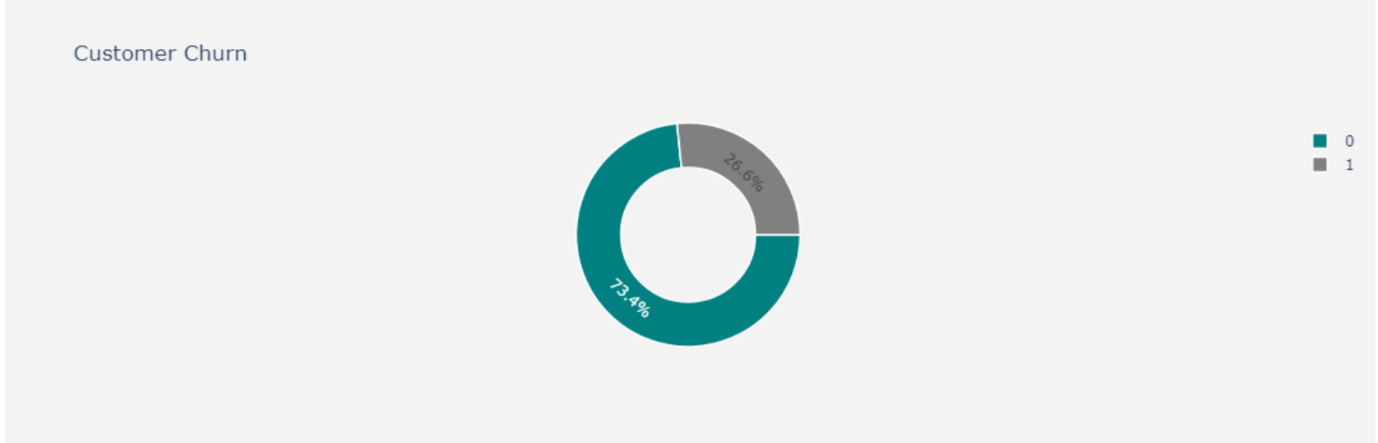
**# Predict the probability of Churn of each customer**

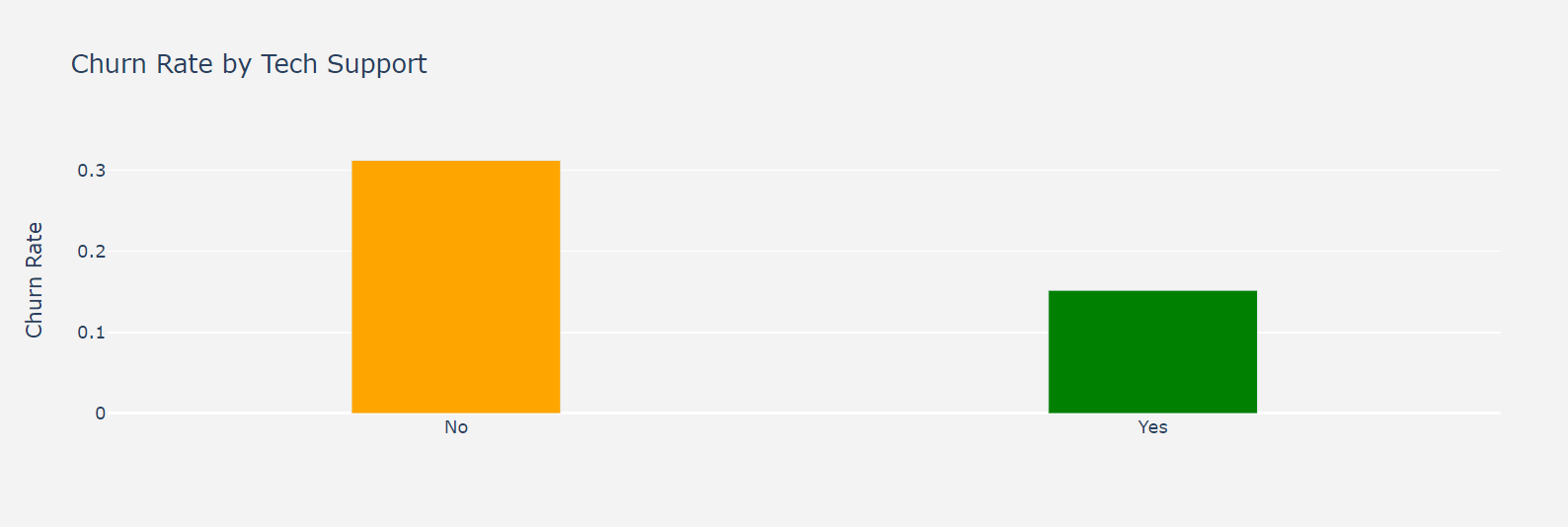
churn\_dataset['Probability\_of\_Churn'] **=** logmodel**.**predict\_proba(churn\_dataset[X\_test**.**columns])[:,1]

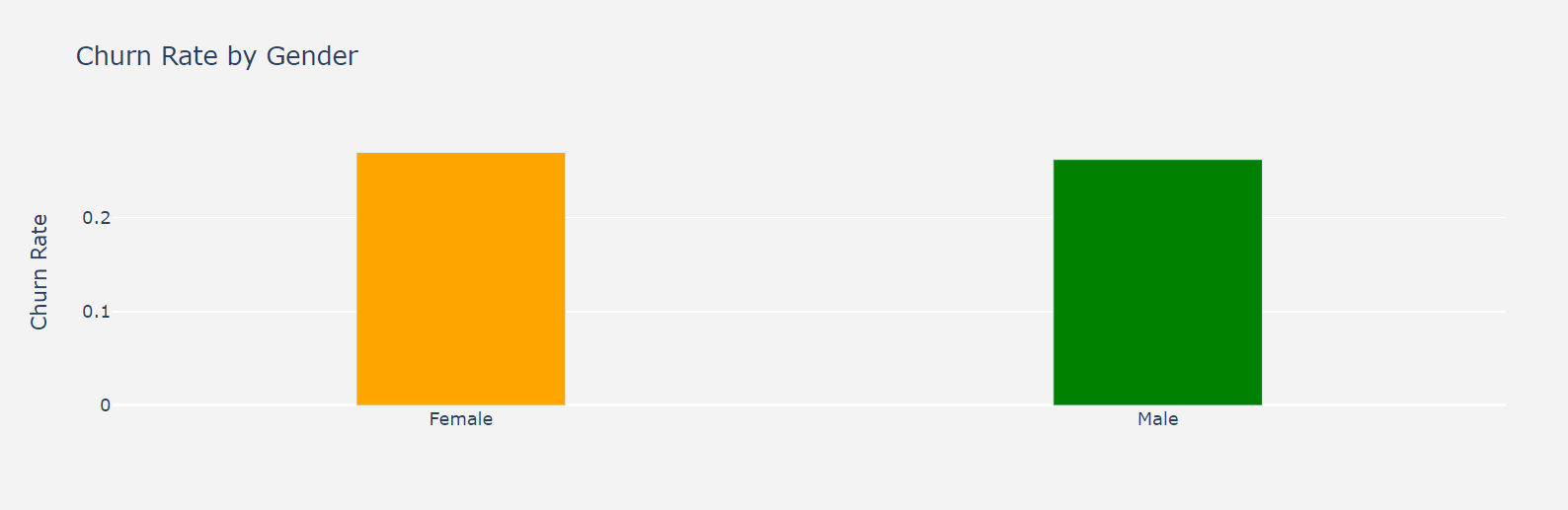
**# Create a Dataframe showcasing probability of Churn of each customer**

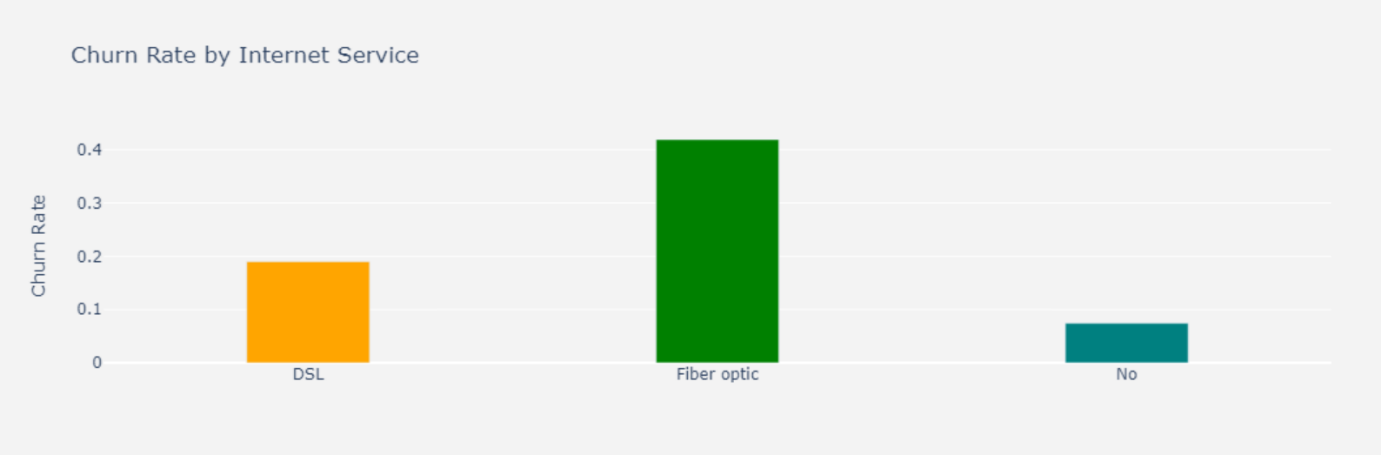
churn\_dataset[['customerID','Probability\_of\_Churn']]**.**head()

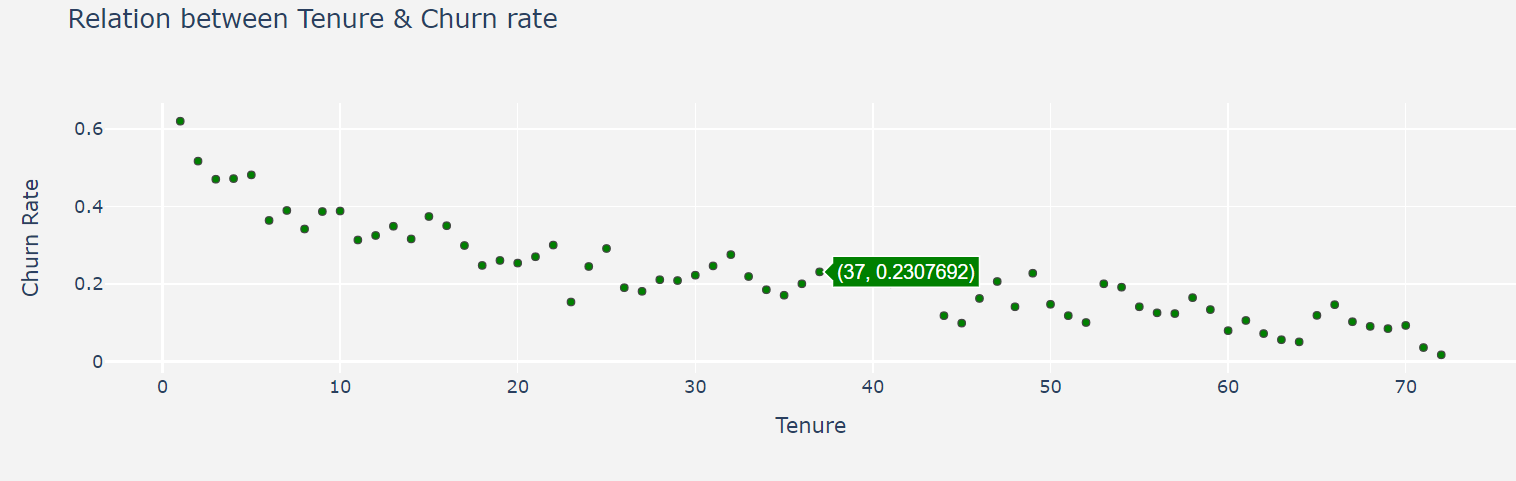
**Output :**

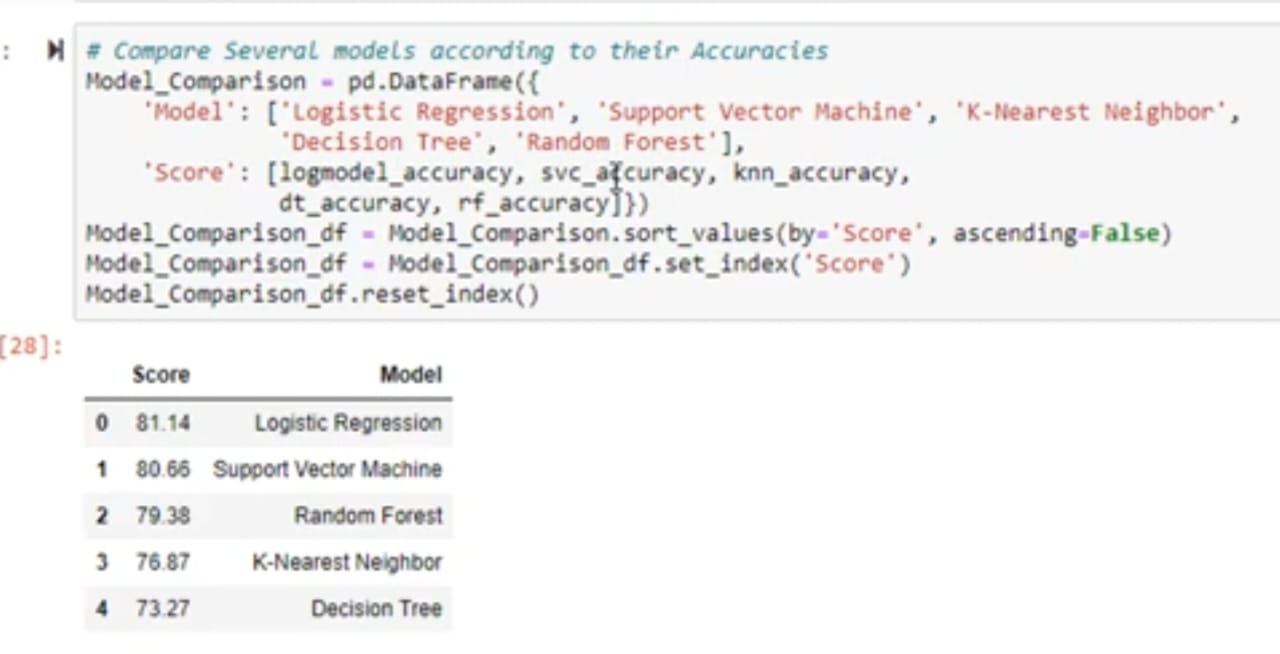
****

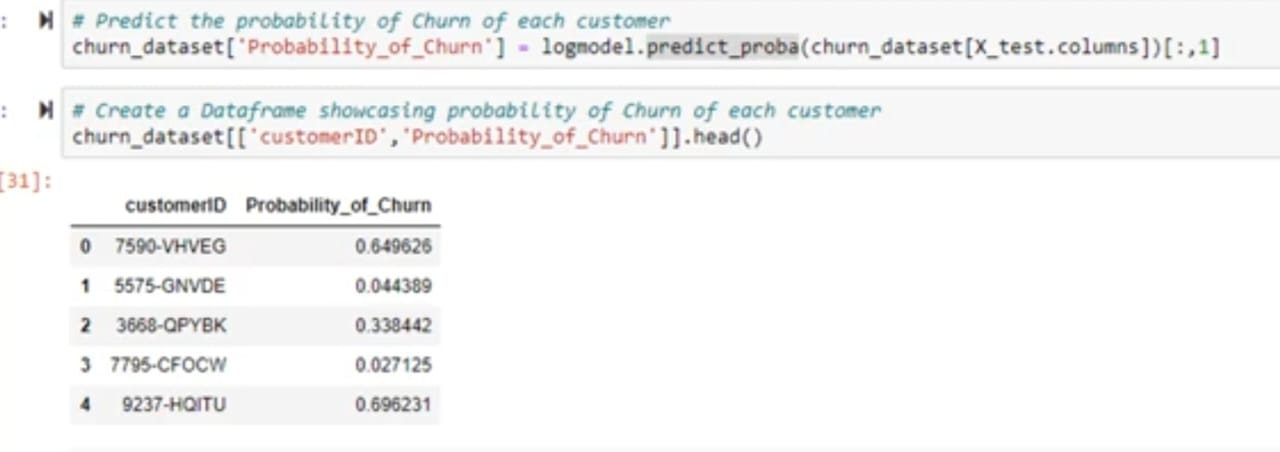


****

****

****

****

****

**Conclusion :**

Thus we have built a Customer Churn prediction model, using which we can get the analysis of customers churning out Category-wise, and that predicts the individual probability whether a particular customer will churn out or not. It also compares the accuracy at which the various machine learning classification algorithms predict the output, which helps us to find the best out of the group.

**Phase – 5 Submission done by :**

Logeshwari. R

810021205051