Project Report

On

Binary Classification with a Bank Churn

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

by

Ms. R. ASHRITHA– 22WH1A6626

Ms. M. SAATVIKA– 22WH1A6628

Ms. M. NITHYASRI– 22WH1A6630

Ms. B. RISHITA – 22WH1A6631

Under the esteemed guidance of

Ms. A Naga Kalyani

Assistant Professor, CSE(AI&ML)

VISHNU
UNIVERSAL LEARNING
BVRITH
Estd. 2012

BVRIT HYDERABAD College of Engineering for Women

(UGC Autonomous Institution | Approved by AICTE | Affiliated to JNTUH)

(NAAC Accredited - A Grade | NBA Accredited B.Tech. (EEE, ECE, CSE and IT)

Bachupally, Hyderabad – 500090

2024-25

Department of Computer Science & Engineering

(Artificial Intelligence & Machine Learning)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with A Grade

Bachupally, Hyderabad – 500090

2024-25

CERTIFICATE

This is to certify that the major project entitled **"Binary Classification with a Bank Churn"** is a Bonafide work carried out by **Ms. R. Ashritha (22WH1A6626), Ms. M. Saatvika (22WH1A6628), Ms. M. Nithya Sri (22WH1A6630), and Ms. B. Rishita (22WH1A6631)** in partial fulfilment of the requirements for the award of **B.Tech degree in Computer Science & Engineering (AI & ML)** at **BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to **Jawaharlal Nehru Technological University Hyderabad, Hyderabad**, under my guidance and supervision. The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Supervisor	**Head of the Department**
Ms. A Naga Kalyani	**Dr. B. Lakshmi Praveena**
Assistant Professor	**HOD & Professor**
Dept of CSE(AI&ML)	**Dept of CSE(AI&ML)**

External Examiner

DECLARATION

We hereby declare that the work presented in this project entitled **"Binary classification with a bank churn"** submitted towards completion of Project work in III Year of B.Tech of CSE(AI&ML) at **BVRIT HYDERABAD College of Engineering for Women,** Hyderabad is an authentic record of our original work carried out under the guidance of **Ms. A Naga Kalyani, Assistant Professor, Department of CSE(AI&ML).**

Sign with Date:

R. Ashritha

(22wh1a6626)

Sign with Date:

M. Saatvika

(22wh1a6628)

Sign with Date:

M. NithyaSri

(22wh1a6630)

Sign with Date:

B. Rishita

(22wh1a6631)

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women**, for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. B. Lakshmi Praveena, Head of the Department, Department of CSE(AI&ML), BVRIT HYDERABAD College of Engineering for Women,** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide**, Ms. A Naga Kalyani, Assistant Professor, CSE(AI&ML), BVRIT HYDERABAD College of Engineering for Women,** for her constant guidance and encouragement throughout the project.

Finally, we would like to thank our Major Project Coordinator, all Faculty and Staff of CSE(AI&ML) department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents and Friends** for giving moral strength and constant encouragement.

R. Ashritha (22wh1a6626)

M. Saatvika (22wh1a6628)

M. NithyaSri (22wh1a6630)

B. Rishita (22wh1a6631)

ABSTRACT

The project **"Binary Classification with a Bank Churn"** aims to predict customer churn, a critical issue for banks striving to retain existing customers and enhance profitability. Churn refers to customers discontinuing their relationship with the bank, and addressing it effectively requires identifying at-risk customers early. This project uses a machine learning approach to classify customers into two categories—those likely to churn and those likely to stay. The dataset includes features such as demographics, account activity, and transaction history, which were pre-processed for quality and readiness through steps like handling missing values, encoding categorical features, and normalizing data.

A **Logistic Regression model** was employed for this task, leveraging its efficiency and interpretability in binary classification problems. Exploratory Data Analysis (EDA) was crucial in uncovering patterns that guided feature engineering, while the model's probabilistic nature helped assess churn likelihood effectively. The model's performance was evaluated using metrics like accuracy, precision, which confirmed its ability to identify potential churners accurately. This project highlights the utility of Logistic Regression in solving real-world business challenges, providing banks with a scalable and actionable solution to minimize churn and improve customer retention strategies.

PROBLEM STATEMENT

Customer churn, where customers discontinue their relationship with a bank, is a major challenge that impacts profitability and customer retention efforts. Retaining existing customers is often more cost-effective than acquiring new ones, making early identification of at-risk customers critical. Churn prediction is complex due to the diverse factors influencing customer behaviour, such as transaction patterns, account activity, and demographics. Accurate predictions enable banks to implement targeted retention strategies, reducing churn and maximizing profitability.

This project aims to develop a machine learning-based solution to classify customers into two categories: churners and non-churners. By analysing historical data and extracting meaningful features, the project focuses on building a scalable, interpretable model suitable for real-world applications. The solution seeks to bridge the gap between raw data and actionable insights, empowering banks to improve retention strategies and optimize their operational decision-making processes.

DATA SET

https://github.com/31Rishita/Bank-Customer-Churn-Prediction/blob/main/Churn_Modelling.csv

SOURCE CODE

1.# For data wrangling

```
import numpy as np
import pandas as pd

# For visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
pd.options.display.max_rows = None
pd.options.display.max_columns = None

df = pd.read_csv('/content/Churn_Modelling.csv', delimiter=',')
df.shape
```

OUTPUT

(10000, 14)

2.# Check columns list and missing values

```
df.isnull().sum()
```

OUTPUT

	0
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

3.# Get unique count for each variable

df.nunique()

OUTPUT

Out[156]:

	0
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

4.df = df.drop(["RowNumber", "CustomerId", "Surname"], axis = 1)

df.head()

OUTPUT

Out[158]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

5. df.dtypes

OUTPUT

Out[159]:

	0
CreditScore	int64
Geography	object
Gender	object
Age	int64
Tenure	int64
Balance	float64
NumOfProducts	int64
HasCrCard	int64
IsActiveMember	int64
EstimatedSalary	float64
Exited	int64

6. labels = 'Exited', 'Retained'

sizes = [df.Exited[df['Exited']==1].count(), df.Exited[df['Exited']==0].count()]

explode = (0, 0.1)

fig1, ax1 = plt.subplots(figsize=(10, 8))

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

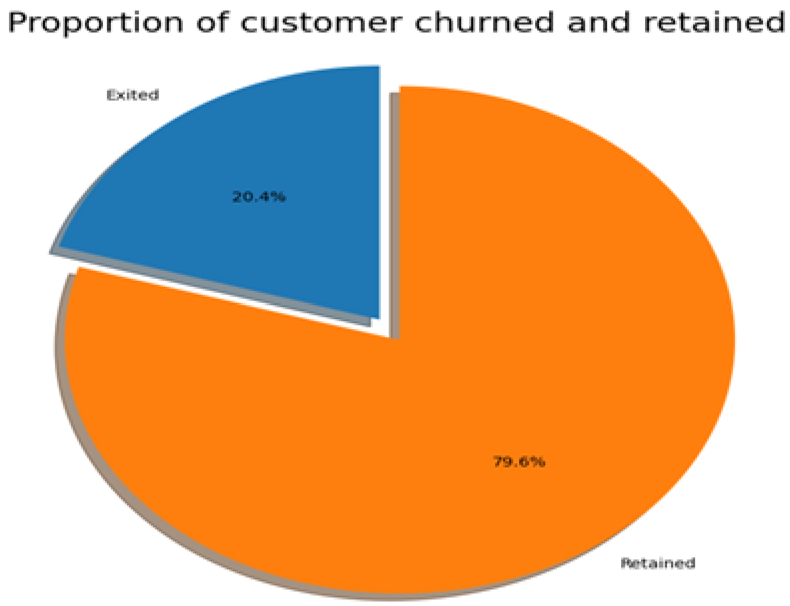
    shadow=True, startangle=90)

ax1.axis('equal')

plt.title("Proportion of customer churned and retained", size = 20)

plt.show()

OUTPUT



7. # We first review the 'Status' relation with categorical variables

fig, axarr = plt.subplots(2, 2, figsize=(20, 12))

sns.countplot(x='Geography', hue = 'Exited',data = df, ax=axarr[0][0])

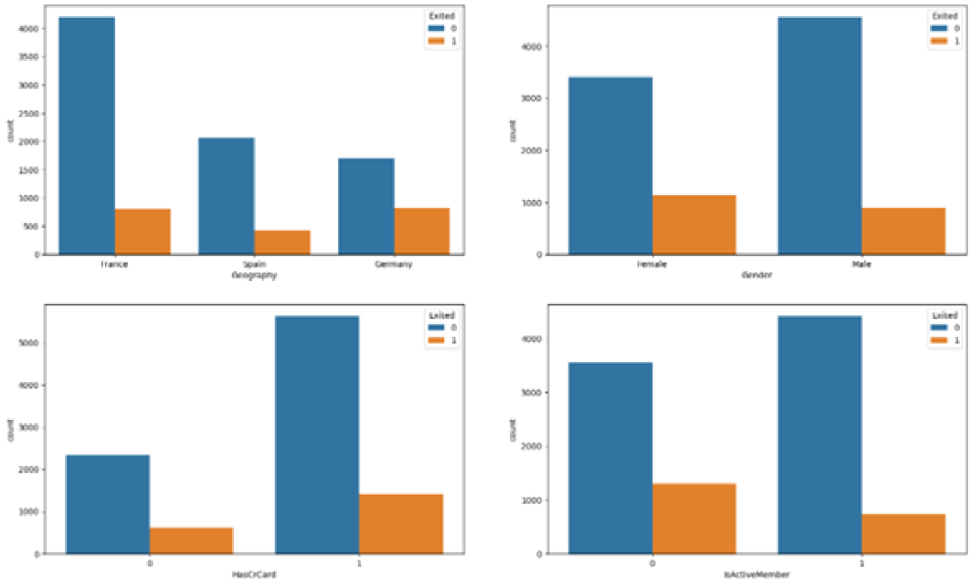
sns.countplot(x='Gender', hue = 'Exited',data = df, ax=axarr[0][1])

sns.countplot(x='HasCrCard', hue = 'Exited',data = df, ax=axarr[1][0])

sns.countplot(x='IsActiveMember', hue = 'Exited',data = df, ax=axarr[1][1])

OUTPUT

```
Out[161]:
<Axes: xlabel='IsActiveMember', ylabel='count'>
```

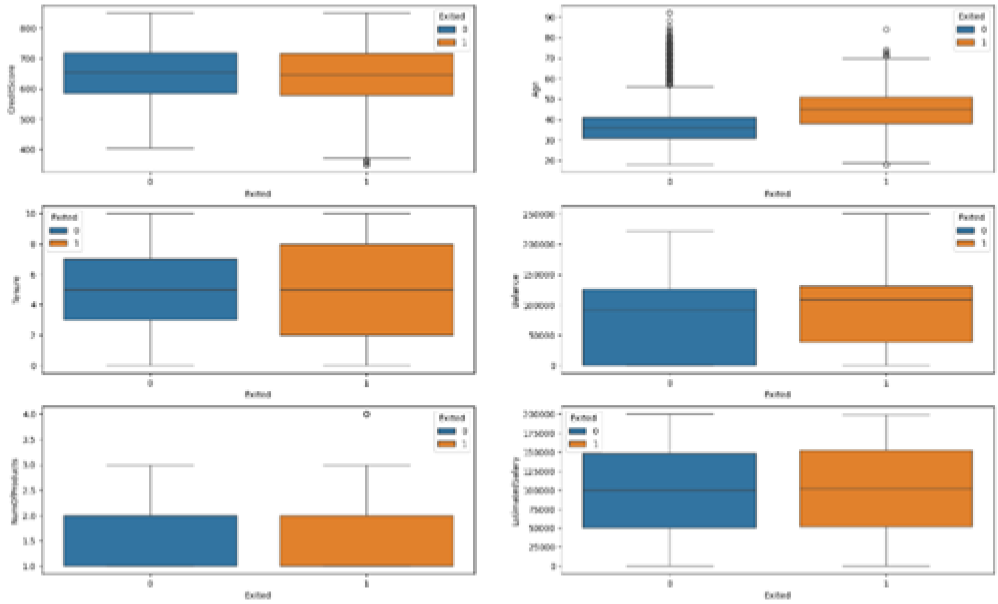


8.# Relations based on the continuous data attributes

```
fig, axarr = plt.subplots(3, 2, figsize=(20, 12))
sns.boxplot(y='CreditScore',x = 'Exited', hue = 'Exited',data = df, ax=axarr[0][0])
sns.boxplot(y='Age',x = 'Exited', hue = 'Exited',data = df , ax=axarr[0][1])
sns.boxplot(y='Tenure',x = 'Exited', hue = 'Exited',data = df, ax=axarr[1][0])
sns.boxplot(y='Balance',x = 'Exited', hue = 'Exited',data = df, ax=axarr[1][1])
sns.boxplot(y='NumOfProducts',x = 'Exited', hue = 'Exited',data = df,
ax=axarr[2][0])
sns.boxplot(y='EstimatedSalary',x = 'Exited', hue = 'Exited',data = df,
ax=axarr[2][1])
```

OUTPUT

<Axes: xlabel='Exited', ylabel='EstimatedSalary'>



9.# Split Train, test data

```
df_train = df.sample(frac=0.8,random_state=200)

df_test = df.drop(df_train.index)

print(len(df_train))

print(len(df_test))
```

OUTPUT

8000

2000

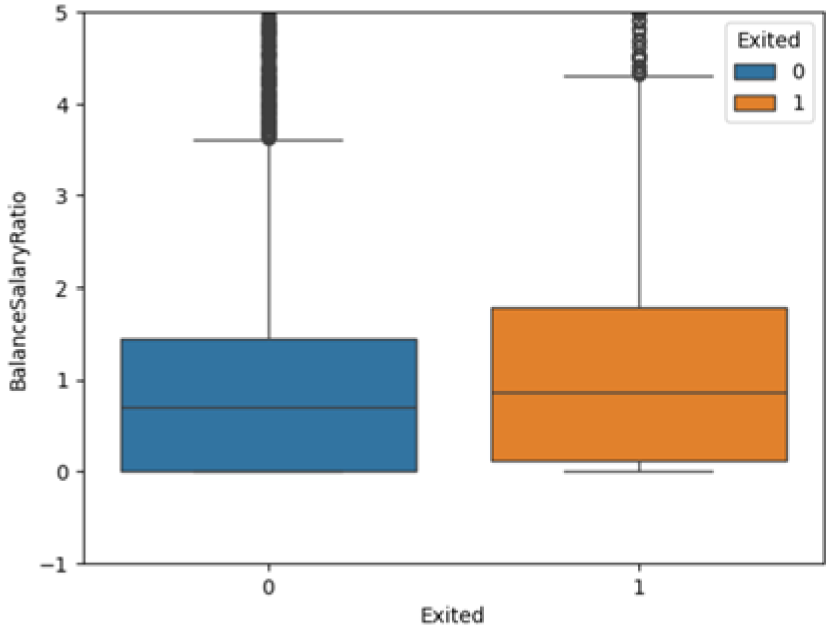
10.
```
df_train['BalanceSalaryRatio'] = df_train.Balance/df_train.EstimatedSalary

sns.boxplot(y='BalanceSalaryRatio',x = 'Exited', hue = 'Exited',data = df_train)

plt.ylim(-1, 5)
```

OUTPUT

```
Out[164]:
(-1.0, 5.0)
```



11.# Given that tenure is a 'function' of age, we introduce a variable aiming to standardize tenure over age:

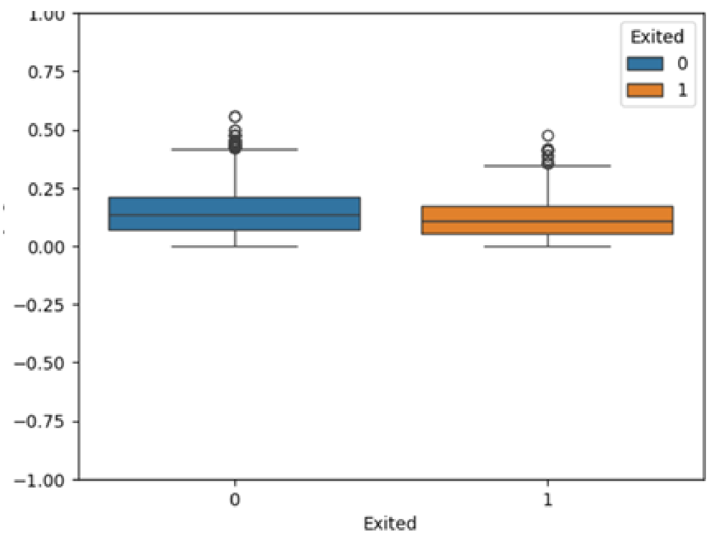
df_train['TenureByAge'] = df_train.Tenure/(df_train.Age)

sns.boxplot(y='TenureByAge',x = 'Exited', hue = 'Exited',data = df_train)

plt.ylim(-1, 1)

plt.show()

OUTPUT



12. '''Lastly we introduce a variable to capture credit score given age to take into account credit behaviour visavis adult life

:-)'''

df_train['CreditScoreGivenAge'] = df_train.CreditScore/(df_train.Age)

 # Resulting Data Frame

df_train.head()

OUTPUT

Out[167]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsAc
8159	461	Spain	Female	25	6	0.00	2	1	
6332	619	France	Female	35	4	90413.12	1	1	
8895	699	France	Female	40	8	122038.34	1	1	
5351	558	Germany	Male	41	2	124227.14	1	1	
4314	638	France	Male	34	5	133501.36	1	0	

13. # Arrange columns by data type for easier manipulation

continuous_vars = ['CreditScore', 'Age', 'Tenure', 'Balance','NumOfProducts', 'EstimatedSalary', 'BalanceSalaryRatio', 'TenureByAge','CreditScoreGivenAge']

cat_vars = ['HasCrCard', 'IsActiveMember','Geography', 'Gender']

df_train = df_train[['Exited'] + continuous_vars + cat_vars]

df_train.head()

OUTPUT

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGive
8159	0	461	25	6	0.00	2	15306.29	0.000000	0.240000	18.44
6332	0	619	35	4	90413.12	1	20555.21	4.398550	0.114286	17.68
8895	0	699	40	8	122038.34	1	102085.35	1.195454	0.200000	17.47
5351	0	558	41	2	124227.14	1	111184.67	1.117305	0.048780	13.60
4314	0	638	34	5	133501.36	1	155643.04	0.857741	0.147059	18.76

14.'''For the one hot variables, we change 0 to -1 so that the models can capture a negative relation

where the attribute in inapplicable instead of 0'''

df_train.loc[df_train.HasCrCard == 0, 'HasCrCard'] = -1

df_train.loc[df_train.IsActiveMember == 0, 'IsActiveMember'] = -1

df_train.head()

OUTPUT

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGive
8159	0	461	25	6	0.00	2	15306.29	0.000000	0.240000	18.44
6332	0	619	35	4	90413.12	1	20555.21	4.398550	0.114286	17.68
8895	0	699	40	8	122038.34	1	102085.35	1.195454	0.200000	17.47
5351	0	558	41	2	124227.14	1	111184.67	1.117305	0.048780	13.60
4314	0	638	34	5	133501.36	1	155643.04	0.857741	0.147059	18.76

15. # One hot encode the categorical variables

```
lst = ['Geography', 'Gender']
remove = list()
for i in lst:
    if (df_train[i].dtype == str or df_train[i].dtype == object):  # Use `str` and `object`
        for j in df_train[i].unique():
            df_train[i + '_' + j] = np.where(df_train[i] == j, 1, -1)
    remove.append(i)
df_train = df_train.drop(remove, axis=1)
df_train.head()
```

OUTPUT

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGive
8159	0	461	25	6	0.00	2	15306.29	0.000000	0.240000	18.44
6332	0	619	35	4	90413.12	1	20555.21	4.398550	0.114286	17.68
8895	0	699	40	8	122038.34	1	102085.35	1.195454	0.200000	17.47
5351	0	558	41	2	124227.14	1	111184.67	1.117305	0.048780	13.60
4314	0	638	34	5	133501.36	1	155643.04	0.857741	0.147059	18.76

16. # minMax scaling the continuous variables

minVec = df_train[continuous_vars].min().copy()

maxVec = df_train[continuous_vars].max().copy()

df_train[continuous_vars] = (df_train[continuous_vars]-minVec)/(maxVec-minVec)

df_train.head()

OUTPUT

Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary	BalanceSalaryRatio	TenureByAge	CreditScoreGivenA
0	0.222	0.094595	0.6	0.000000	0.333333	0.076118	0.000000	0.432000	0.3231
0	0.538	0.229730	0.4	0.360358	0.000000	0.102376	0.003317	0.205714	0.3052
0	0.698	0.297297	0.8	0.486406	0.000000	0.510225	0.000901	0.360000	0.3001
0	0.416	0.310811	0.2	0.495130	0.000000	0.555744	0.000843	0.087805	0.2082
0	0.576	0.216216	0.5	0.532094	0.000000	0.778145	0.000647	0.264706	0.3308

17. # data prep pipeline for test data

def DfPrepPipeline(df_predict,df_train_Cols,minVec,maxVec):

 # Add new features

df_predict['BalanceSalaryRatio'] = df_predict.Balance/df_predict.EstimatedSalary

df_predict['TenureByAge'] = df_predict.Tenure/(df_predict.Age - 18)

df_predict['CreditScoreGivenAge'] = df_predict.CreditScore/(df_predict.Age - 18)

 # Reorder the columns

```
continuous_vars = ['CreditScore','Age','Tenure','Balance','NumOfProducts','EstimatedSalary','BalanceSalaryRatio',
                'TenureByAge','CreditScoreGivenAge']
cat_vars = ['HasCrCard','IsActiveMember',"Geography", "Gender"]
df_predict = df_predict[['Exited'] + continuous_vars + cat_vars]
# Change the 0 in categorical variables to -1
df_predict.loc[df_predict.HasCrCard == 0, 'HasCrCard'] = -1
df_predict.loc[df_predict.IsActiveMember == 0, 'IsActiveMember'] = -1
# One hot encode the categorical variables
lst = ["Geography", "Gender"]
remove = list()
for i in lst:
    for j in df_predict[i].unique():
        df_predict[i+'_'+j] = np.where(df_predict[i] == j,1,-1)
    remove.append(i)
df_predict = df_predict.drop(remove, axis=1)

# Ensure that all one hot encoded variables that appear in the train data appear in the subsequent data
L = list(set(df_train_Cols) - set(df_predict.columns))
for l in L:
    df_predict[str(l)] = -1
# MinMax scaling coontinuous variables based on min and max from the train data
df_predict[continuous_vars] = (df_predict[continuous_vars]-minVec)/(maxVec-minVec)
# Ensure that The variables are ordered in the same way as was ordered in the train set
```

```python
    df_predict = df_predict[df_train_Cols]
    return df_predict

# Support functions
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
# Fit models
from sklearn.linear_model import LogisticRegression
# Scoring functions
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

# Function to give best model score and parameters
def best_model(model):
    print(model.best_score_)
    print(model.best_params_)
    print(model.best_estimator_)
def get_auc_scores(y_actual, method,method2):
    auc_score = roc_auc_score(y_actual, method);
    fpr_df, tpr_df, _ = roc_curve(y_actual, method2);
    return (auc_score, fpr_df, tpr_df)

# One-Hot Encoding categorical variables
df_train_encoded = pd.get_dummies(df_train, drop_first=True)
```

 # Fit primal logistic regression

```
log_primal = LogisticRegression(C=100, class_weight=None, dual=False,
fit_intercept=True, intercept_scaling=1,
                    max_iter=250, multi_class='auto', n_jobs=None,
penalty='l2', random_state=None,
                    solver='lbfgs', tol=1e-05, verbose=0, warm_start=False)
log_primal.fit(df_train_encoded.loc[:, df_train_encoded.columns != 'Exited'],
df_train_encoded.Exited)
```

OUTPUT

```
LogisticRegression(C=100, max_iter=250, multi_class='auto', tol=1e-05)
```

18.# One-Hot Encoding categorical variables

```
df_train_encoded = pd.get_dummies(df_train, drop_first=True)
```

 # Apply Polynomial Features (degree 2)

```
poly2 = PolynomialFeatures(degree=2)
df_train_pol2 = poly2.fit_transform(df_train_encoded.loc[:,
df_train_encoded.columns != 'Exited'])
```

 # Fit logistic regression with polynomial features

```
log_pol2 = LogisticRegression(
    C=10,
    class_weight=None,
    dual=False,
    fit_intercept=True,
    intercept_scaling=1,
    max_iter=300,
    multi_class='auto',  # Changed from 'warn' to 'auto'
    n_jobs=None,
    penalty='l2',
```

```
    random_state=None,
    solver='liblinear',
    tol=0.0001,
    verbose=0,
    warm_start=False
)
log_pol2.fit(df_train_pol2, df_train_encoded.Exited)
```

OUTPUT

```
176… LogisticRegression(C=10, max_iter=300, multi_class='auto', solver='liblinear')
```

19. # Apply One-Hot Encoding to the training data

```
df_train_encoded = pd.get_dummies(df_train, drop_first=True)
# Apply PolynomialFeatures to the encoded training data
poly2 = PolynomialFeatures(degree=2)
df_train_pol2 = poly2.fit_transform(df_train_encoded.loc[:,
df_train_encoded.columns != 'Exited'])
# Fit logistic regression on the transformed training data
log_primal.fit(df_train_pol2, df_train_encoded.Exited)
# Now, apply the same transformations to the training data for prediction
df_train_encoded_pred = pd.get_dummies(df_train, drop_first=True)
df_train_pol2_pred = poly2.transform(df_train_encoded_pred.loc[:,
df_train_encoded_pred.columns != 'Exited'])
# Now, predict using the trained model
predictions = log_primal.predict(df_train_pol2_pred)
# Print classification report
print(classification_report(df_train.Exited, predictions))
```

OUTPUT

```
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      6353
           1       0.76      0.47      0.58      1647

    accuracy                           0.86      8000
   macro avg       0.82      0.72      0.75      8000
weighted avg       0.85      0.86      0.85      8000
```

20. print(classification_report(df_train.Exited, log_pol2.predict(df_train_pol2)))

OUTPUT

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      6353
           1       0.77      0.46      0.57      1647

    accuracy                           0.86      8000
   macro avg       0.82      0.71      0.75      8000
weighted avg       0.85      0.86      0.85      8000
```

21. # Apply PolynomialFeatures with different degrees

poly1 = PolynomialFeatures(degree=1) *# For primal*

poly2 = PolynomialFeatures(degree=2) *# For polynomial kernel*

X_pol1 = poly1.fit_transform(X) *# Apply transformation for primal*

X_pol2 = poly2.fit_transform(X) *# Apply transformation for polynomial kernel*

 # Fit logistic regression models on different feature transformations

log_primal.fit(X_pol1, y)

log_pol2.fit(X_pol2, y)

 # Get AUC scores for each model with the corresponding transformed data

auc_log_primal, fpr_log_primal, tpr_log_primal = get_auc_scores(y, log_primal.predict(X_pol1), log_primal.predict_proba(X_pol1)[:, 1])

print(f"AUC for Logistic Regression with Primal Features (Degree 1): {auc_log_primal}") auc_log_pol2, fpr_log_pol2, tpr_log_pol2 = get_auc_scores(y, log_pol2.predict(X_pol2), log_pol2.predict_proba(X_pol2)[:, 1])

print(f"AUC for Logistic Regression with Polynomial Features (Degree 2): {auc_log_pol2}")

OUTPUT

```
AUC for Logistic Regression with Primal Features (Degree 1): 0.778152799603876
AUC for Logistic Regression with Polynomial Features (Degree 2): 0.8703684111584858
```

22. plt.figure(figsize = (12,6), linewidth= 1)

plt.plot(fpr_log_primal, tpr_log_primal, label = 'log primal Score: ' + str(round(auc_log_primal, 5)))

plt.plot(fpr_log_pol2, tpr_log_pol2, label = 'log pol2 score: ' + str(round(auc_log_pol2, 5)))

plt.plot([0,1], [0,1], 'k--', label = 'Random: 0.5')

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

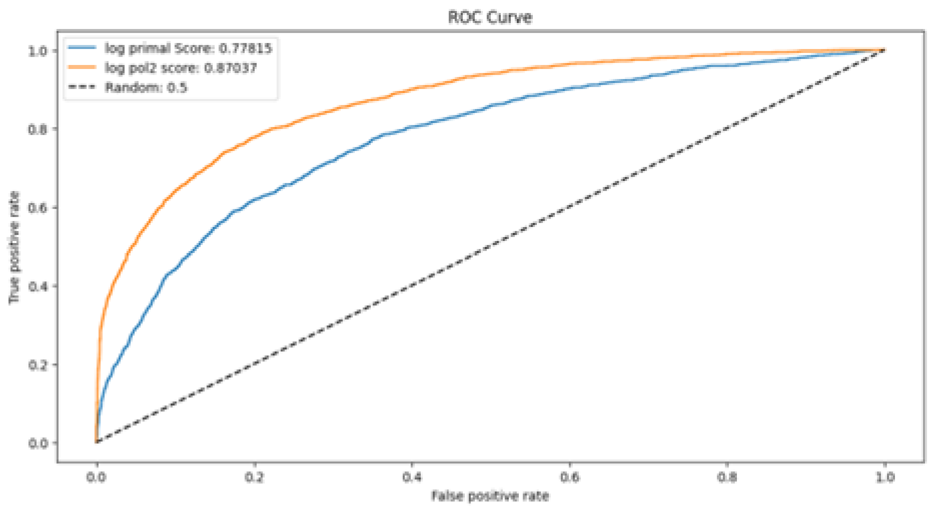
plt.title('ROC Curve')

plt.legend(loc='best')

#plt.savefig('roc_results_ratios.png')

plt.show()

OUTPUT



23. # Make the data transformation for test data

```
df_test = DfPrepPipeline(df_test,df_train.columns,minVec,maxVec)
df_test = df_test.mask(np.isinf(df_test))
df_test = df_test.dropna()
df_test.shape
```

OUTPUT

```
(1996, 17)
```

GITHUB LINK

https://github.com/31Rishita/Bank-Customer-Churn-Prediction