Bank Churn Prediction: An In-Depth Analysis to Improve Customer Retention

INTRODUCTION:

In the current highly driven banking sector, banks are more concerned about customer attrition. A bank's success can be greatly impacted by its capacity to predict client attrition and implement proactive efforts to keep customers. We will delve further into the subject of bank churn prediction in this blog article, going over important ideas, strategies, and best practices.

The dataset and original code can be accessed through.....

The Challenge:

One of the main issues facing financial institutions in the current banking climate is the ongoing battle with client attrition, or "churn." With customer preferences constantly shifting in this highly competitive market, a forward-thinking strategy to client retention is crucial. In light of this need, developing a clever and accurate churn prediction model becomes essential. This model requires the use of modern data analytics and machine learning techniques in order to identify patterns and identify client behaviours that may indicate impending churn. Banks can effectively address this issue and boost customer satisfaction and client relationships by proactively implementing tailored retention initiatives. In the

end, this will protect the institution's stability and ongoing growth from fierce competition.

Bank Churn Prediction using popular classification algorithms.

The Data:

The datasets describe over 10,000 data points, which includes features such RowNumber, CustomerId, Surname CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts,, HasCrCard, IsActiveMember, EstimatedSalary, Exited.

Let's now begin this task by importing the required dataset and the Python libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

```
data=pd.read_csv('/content/Churn_Modelling.csv')
```

```
data.head()
```

wNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
1	15634602	Hargrave	619	France	Female	42	2	0.0(
2	15647311	Hill	608	Spain	Female	41	1	83807.8(
3	15619304	Onio	502	France	Female	42	8	159660.8(
4	15701354	Boni	699	France	Female	39	1	0.0(
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8(

Let's now examine the details of each column within the dataset:

```
[ ]  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
```

Lets’s now examine the training and testing data :

8000 rows *14 columns (80% of the entire dataset) are in the training group, while 2000 rows*14 columns (20%) are in the testing group.

```
Train set: 8000 rows x 14 columns
Test set: 2000 rows x 14 columns
```

EDA/Data Preprocessing

With the dataset in hand, we can perform some EDA magic on it:

1.Checking Null Values:

```
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
```

We can infer that there are no null values.

2. Identify numerical and categorical features:

```
data.dtypes
```

```
RowNumber          int64
CustomerId         int64
Surname           object
CreditScore        int64
Geography         object
Gender            object
Age                int64
Tenure             int64
Balance          float64
NumOfProducts      int64
HasCrCard          int64
IsActiveMember     int64
EstimatedSalary  float64
Exited             int64
dtype: object
```

From here we can infer that there are 11 numerical values and 3 ctagorical values.

3.Drooping unnecessary columns:

data=data.drop(['RowNumber','CustomerId','Surname'],axis=1)

```
data=data.drop(['RowNumber','CustomerId','Surname'],axis=1)

data.head()
```

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

This data is being dropped since they donot play major role in the dataset.

4.Encoding Categorical Data:

train_data_ = pd.get_dummies(data, columns=['Gender’, ‘Geography'])

```
train_data_ = pd.get_dummies(data, columns=['Gender','Geography'])
train_data_.head()
```

	Gender_Female	Gender_Male	Geography_France	Geography_Germany	Geography_Spai
	1	0	1	0	
	1	0	0	0	
	1	0	1	0	
	1	0	1	0	
	1	0	0	0	

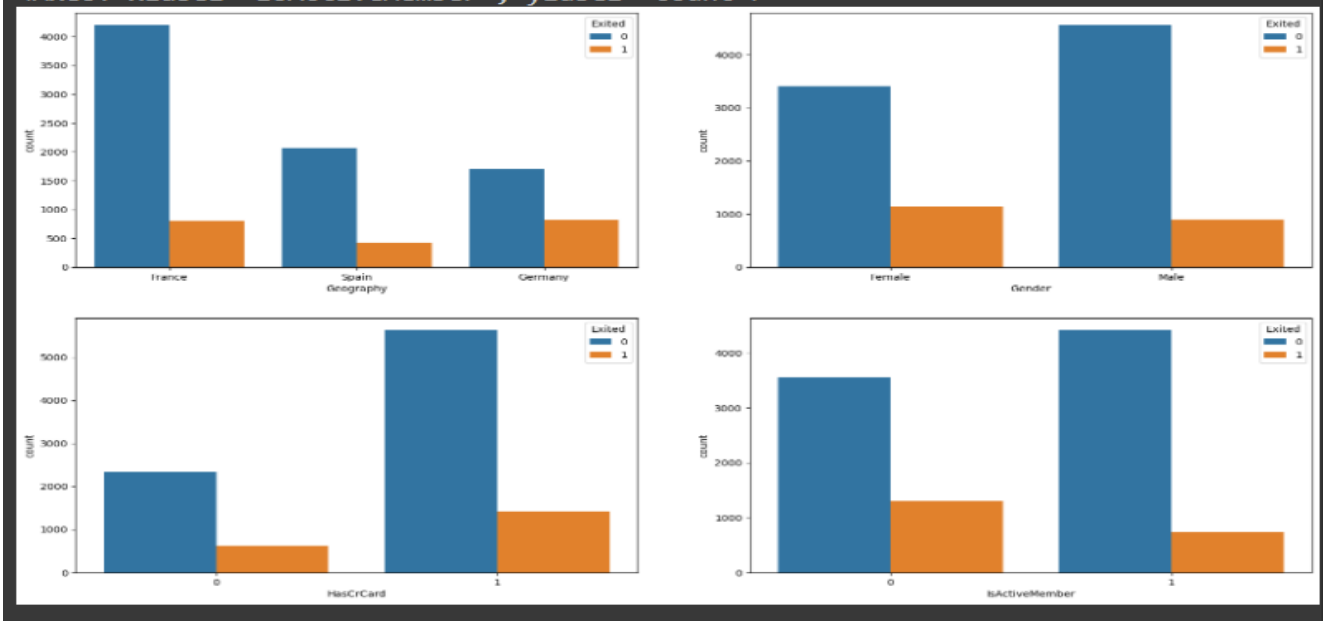
Categorical is converted to numerical form so it would be easy and efficient for data handling in further processes.

VISUALIZATION:

1.PLOTTING CATAGORICAL DATA:

```
#plotting for catagorical
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])
```

```
<Axes: xlabel='IsActiveMember', ylabel='count'>
```

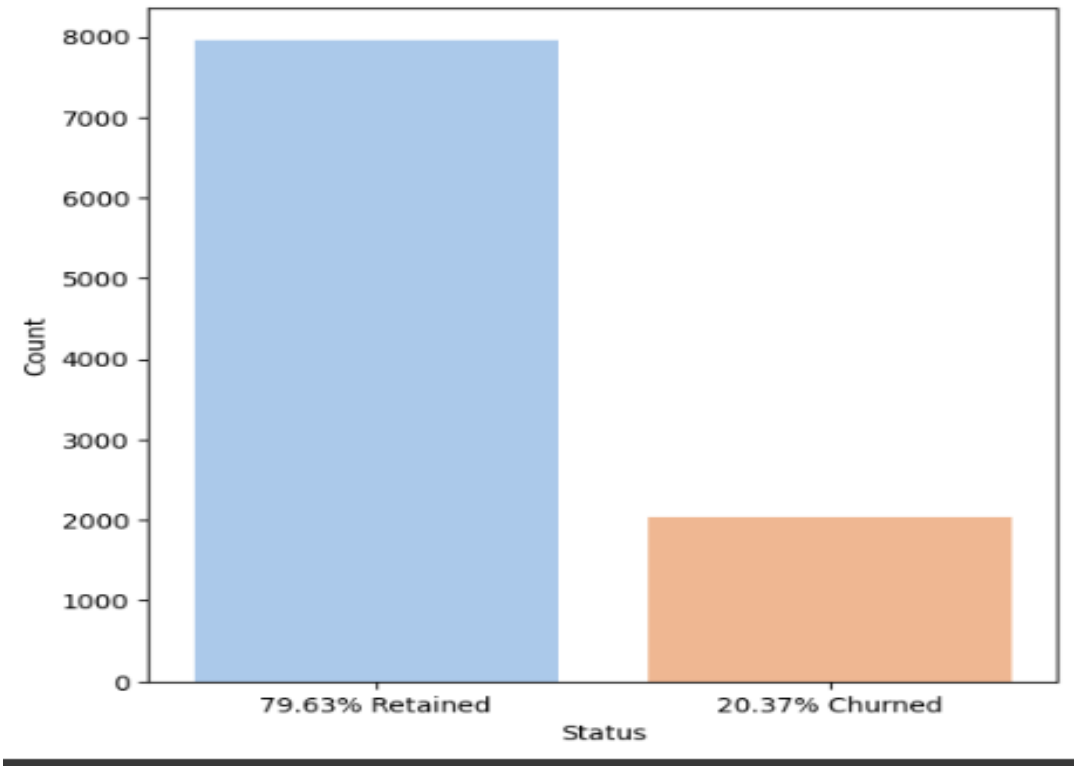


The bank has customers in France, Spain, and Germany, predominantly male, with a majority having a credit card, but nearly 50% are not active.

2.Finding total number of customers churned or retained:

The 'Churned' class is a minority in the dataset because the vast majority (80%) of clients are retained.

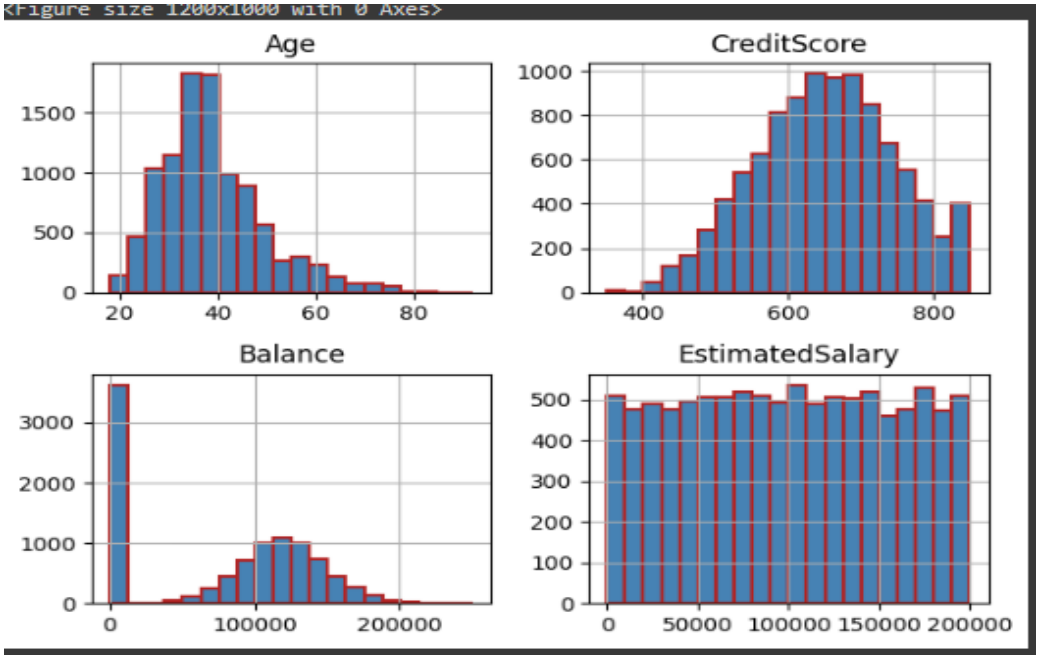
```
Fig, ax = plt.subplots(figsize=(6, 6))

sns.countplot(x='Exited', data=data, palette='pastel' ,ax=ax)

total_samples = len(data)

churn_percentage = (data['Exited'].sum() / total_samples) * 100

retained_percentage = 100 - churn_percentage

ax.set_xticklabels(['{:.2f}%      Retained'.format(retained_percentage), '{:.2f}% Churned'.format(churn_percentage)])

ax.set_xlabel('Status')

ax.set_ylabel('Count')

plt.show()
```



The dataset is imbalanced because a large majority (80%) of clients are retained, making the 'Churned' class a minority.

3.Plotting all Numeric Variables:

```
numeric = data[['Age', 'CreditScore', 'Balance', 'EstimatedSalary']]
plt.figure(figsize=(12, 10))
numeric.hist(bins=20, color='steelblue', edgecolor='firebrick', linewidth=1.5, layout=(2, 2))
plt.tight_layout()
plt.show()
```

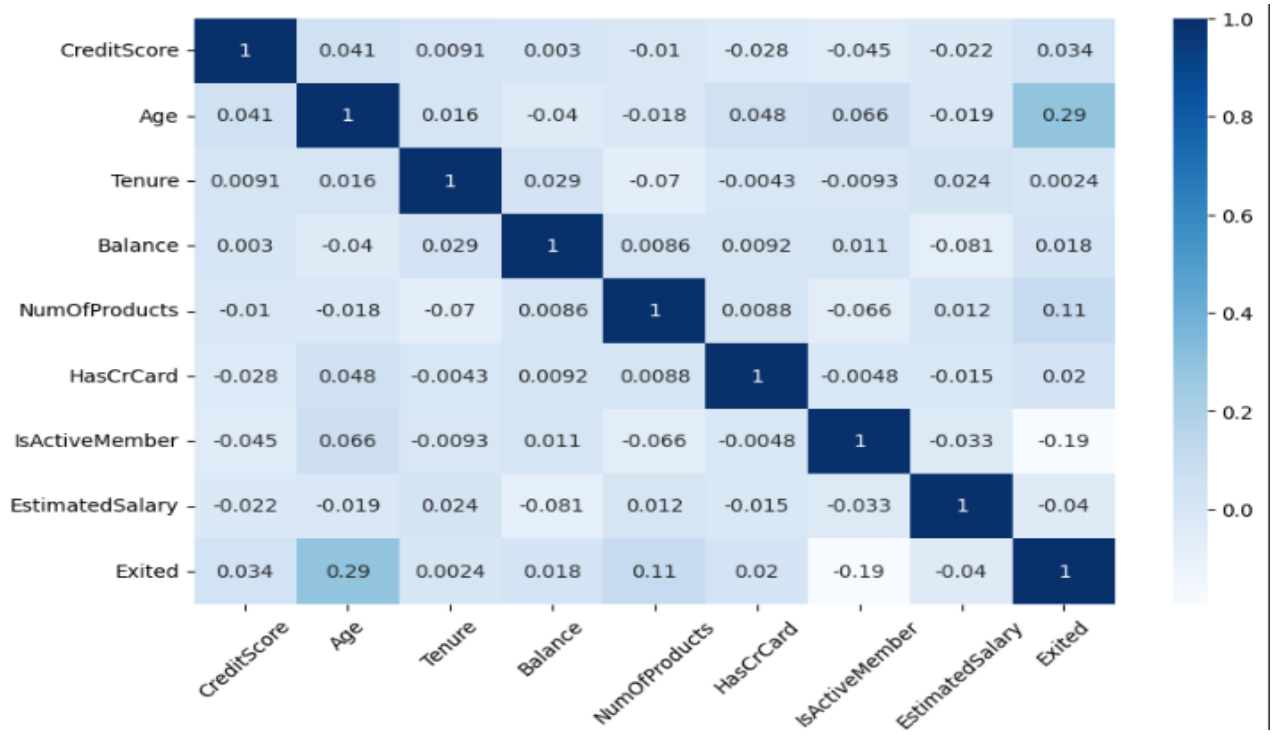


The dataset shows a slightly tail-heavy age distribution, with most clients having a credit score above 600. The balances follow a bell curve, and salaries are fairly evenly distributed.

4. Correlation:

Draw correlation heatmaps for numerical features, as categorical variables require a more complex method.

```
plt.figure(figsize=(10, 6))

sns.heatmap(correlation_matrix, annot=True, cmap='Blues',
xticklabels=correlation_matrix.columns,
yticklabels=correlation_matrix.columns)

plt.xticks(rotation=45)

plt.show()
```



There is no significant intercorrelation between our features.

5. Using ML algorithm for training:

We have used multiple algorithms for training purposes like Gaussian Navies Bayes, Logistic Regression, KNN Classifier, etc.

Among all the algorithms KNN performs best on the validation data with an accuracy score of 0.832 %.

Comparison of Performance Metrics:

	Model	Accuracy	Precision	Recall	F1-score
0	KNN	0.832	0.60	0.43	0.50
1	Linear Regression	0.811	0.55	0.20	0.29
2	Gaussian Naive Bayes	0.820	0.56	0.38	0.45

Conclusion

Banks can use bank churn prediction as a vital tool to predict customer behavior and take proactive steps to keep customers. Banks can establish enduring customer relationships and develop successful retention strategies by employing advanced analytics and data-driven insights.

Best practices for churn prediction, such as frequent updates, model retraining, performance monitoring, and model validation on unseen data, should be implemented by banks. Accurate interpretation requires cooperation between banking specialists, business analysts.