DESCRIPTION

The problem statement is divided into two parts:

1. Attrited Customer / Churned Customer	2. Existing customers
To find Patterns to predicts why they left the services of banks and to predict which customers are at risk of churning.	By identifying patterns and trends in our data, we can create personalized offers and services that customers, ultimately elevating their satisfaction and potentially attract new customers.
The data-driven approach enables the bank to better understand the factors influencing customer departures and take proactive steps to mitigate such occurrences are tailored to the unique needs of our existing.	To improving customer interactions and offering more relevant services by analyzing credit card usage and financial attributes. This approach increases customer satisfaction and loyalty.

→ **Attrited Customer / Churned Customer**

Definition of Attrited Customer / Churned Customer: These terms refer to customers who have terminated their relationship with the bank and are no longer availing its services.

Proposed Solution: To address the issue of customer attrition, we aim to identify patterns within the dataset of churned customers. By doing so, we can predict which customers are at risk of churning. The dataset contains several key attributes, including:

- geographical location
- Gender
- Age
- Tenure
- account balance
- credit score
- membership status
- estimated salary.

Significance of Identifying Patterns: Analyzing these patterns can provide valuable insights into the reasons for customer attrition. Furthermore, it allows us to develop strategies to reduce attrition rates and enhance customer satisfaction through improved personalized banking services. This data-driven approach enables the bank to better understand the factors influencing customer departures and take proactive steps to mitigate such occurrences.

→ **Data-Driven Solutions to Enhance the Banking Experience for Existing Customers :** Our strategy involves a thorough analysis of customer data with the goal of enhancing the banking experience for our existing customers. We will delve into various attributes within our dataset, including:

- Age

- Gender
- Credit Limit
- Average Utilization Ratio
- Income Category
- Credit Card Category
- Marital Status
- Education Level
- Months on Book/Tenure
- Active Status/Months of Inactivity

Let's discuss each solution in more detail:

1.Customer Segmentation based on Age, Gender, and Credit Limit:

Proposed Solution: We will start by analyzing customer behaviour in relation to their age, gender, and credit limit. The primary objective here is to identify patterns in how these factors affect the utilization of banking products. We aim to determine which products or services customers are more likely to use based on these attributes.

Significance of Identifying Patterns: Understanding the relationship between age, gender, and credit limit and how it impacts customer behaviour is crucial. It helps us create a more personalised banking experience by identifying which products and services customers are most likely to engage with. This, in turn, allows us to offer targeted promotions and coupons for the products that align with their preferences. By doing so, we can significantly boost customer engagement and overall satisfaction.

2.Marital Status and Utilization Change:

Proposed Solution: The second part of our solution focuses on studying how marital status impacts customer utilization of banking products. This analysis will delve into how customers' average utilization changes during significant life events, such as marriage or divorce, and how these changes influence their product choices.

Significance of Identifying Patterns: The significance of analyzing how marital status influences customer utilization is substantial. We can provide more tailored services based on customers' marital status, thereby increasing their satisfaction. By adapting our offerings to their specific needs, we foster stronger customer loyalty and improve their overall banking experience. This is a pivotal step in enhancing our relationships with our valued customers.

3.Customer Credit Card Analysis:

Proposed Solution: Our initial focus will be on examining the types of credit cards held by customers. This analysis encompasses factors such as card category (e.g., gold, platinum, premium), the benefits associated with these cards, and how customers use them.

Significance of Identifying Patterns: Understanding how customers use different credit cards is vital. This analysis helps us identify the features they value the most and how effectively they leverage their card benefits. By comprehending their usage patterns, we can enhance our service offerings to better suit their preferences.

4.Income, Credit Limit, and Average Utilization Analysis:

Proposed Solution: The next step involves exploring the relationship between a customer's income, credit limit, and their average utilization of credit. By understanding how these factors interrelate, we can offer more tailored financial products and services that align with individual financial situations.

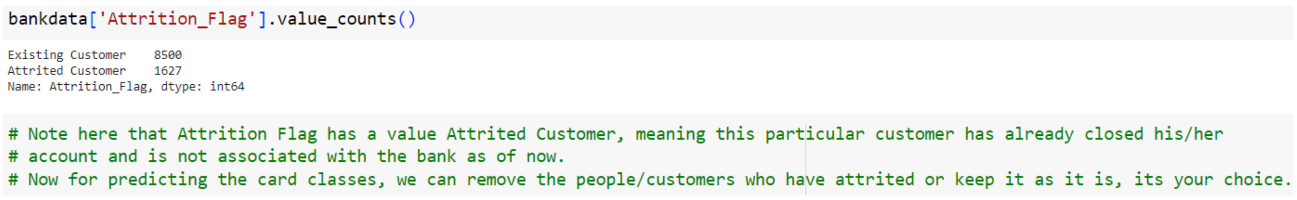
Significance of Identifying Patterns: This analysis holds significance as it allows us to tailor financial products to match each customer's unique financial situation. By identifying how income, credit limit, and credit utilization interplay, we can ensure the services offered are better aligned with their needs.

5.Tenure and Activity Status Monitoring:

Proposed Solution: Lastly, we will closely monitor customers' tenure with the bank and their activity status, assessing how frequently they utilize our banking services. This information will guide us in rewarding loyal and active customers with personalized services, thereby improving their overall experience.

Significance of Identifying Patterns: Tracking customer tenure and activity status is crucial for recognizing and rewarding loyal and active customers. By personalizing our services based on this information, we enhance customer satisfaction and loyalty, ultimately improving their banking experience.

Data Analysis

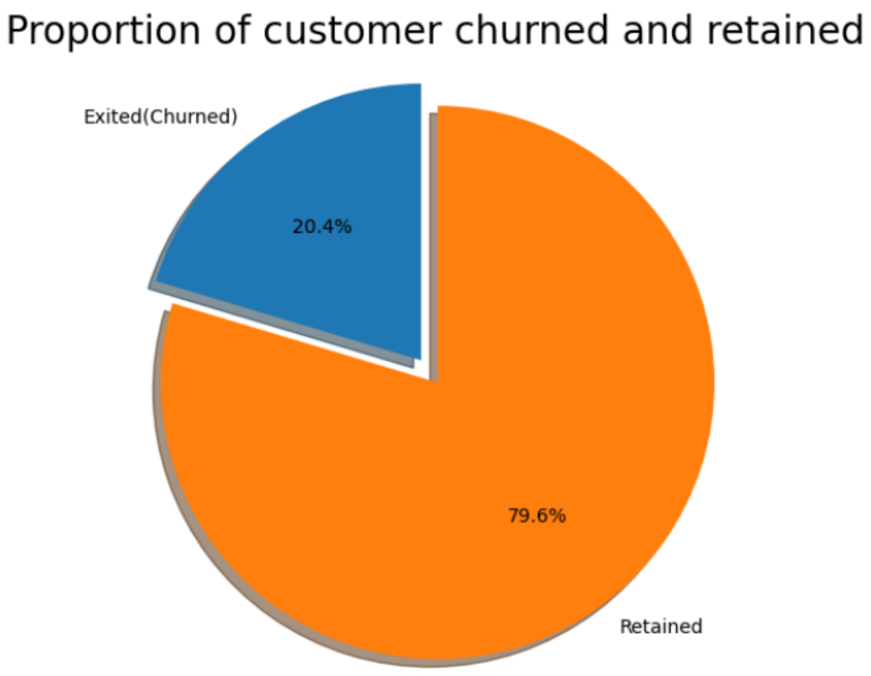


```
bankdata['Attrition_Flag'].value_counts()
```

```
Existing Customer    8500
Attrited Customer    1627
Name: Attrition_Flag, dtype: int64
```

```
# Note here that Attrition Flag has a value Attrited Customer, meaning this particular customer has already closed his/her
# account and is not associated with the bank as of now.
# Now for predicting the card classes, we can remove the people/customers who have attrited or keep it as it is, its your choice.
```

```
#These pie chart beautifully show us the percentage of people churned/left
#Thus we can focus our analysis of the 20.4% who left to find why they left
#Also predict if any other customer from the rest 79.6% are showing the same pattern
```



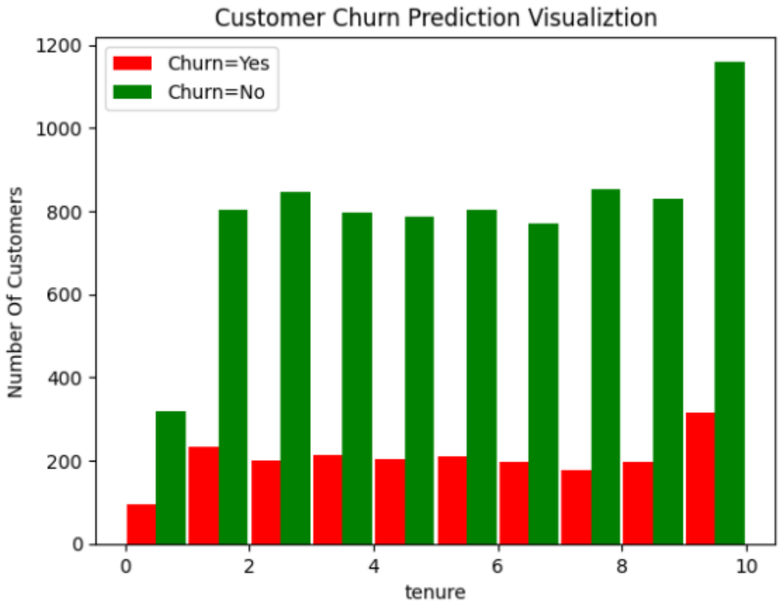
```
#The data tell us about the count of all unique values in the particular coloum
# Important
 # -- only 3 geography location i.e France , Spain & Germany
 # -- Dataset is vast for age as there are total 70 unique age
 # -- Exited value is 1 is Churned/left and 0 if not.
 # -- Similarly Active member value is 1 if active otherwise 0.
```

```
CreditScore         460
Geography             3
Gender                2
Age                  70
Tenure               11
Balance            6382
NumOfProducts         4
HasCrCard             2
IsActiveMember        2
EstimatedSalary    9999
Exited                2
dtype: int64
```

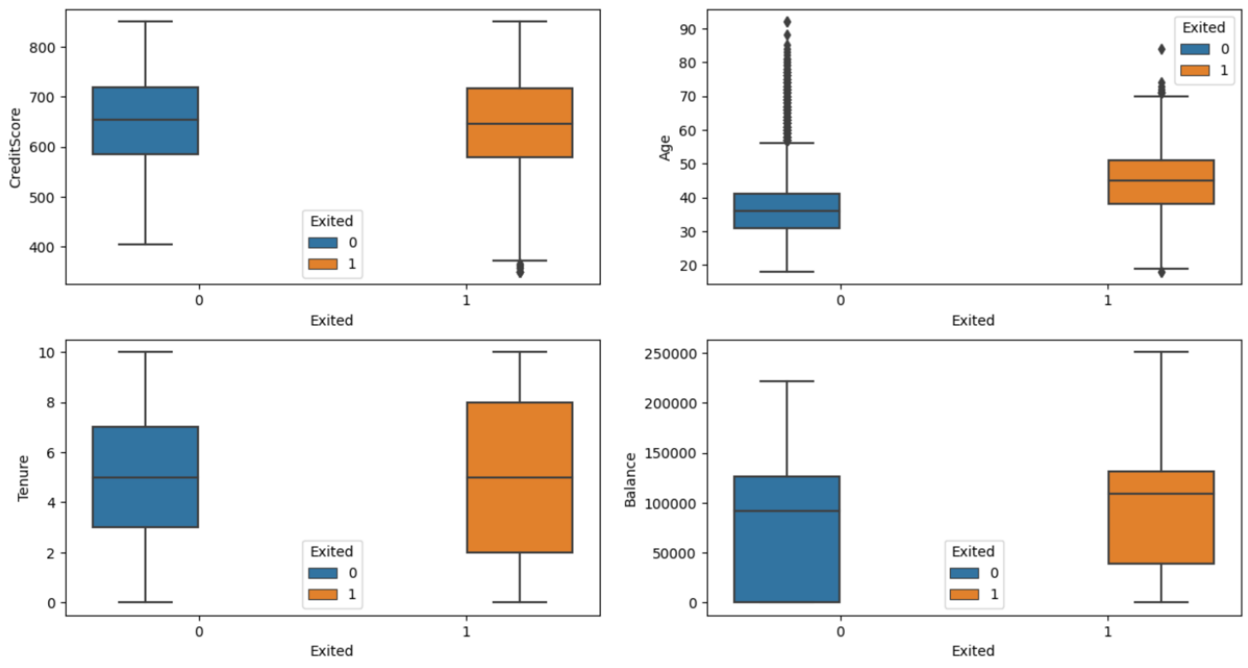
<u>1. Attrited Customer / Churned Customer</u>

```
#It depicts the ratio of customer with tenure and Churn
# and the data show us that the ratio of new customer leaving is high which is common
# but the Churn ratio of old customers is also high
```

```
<matplotlib.legend.Legend at 0x7c656f58e9b0>
```

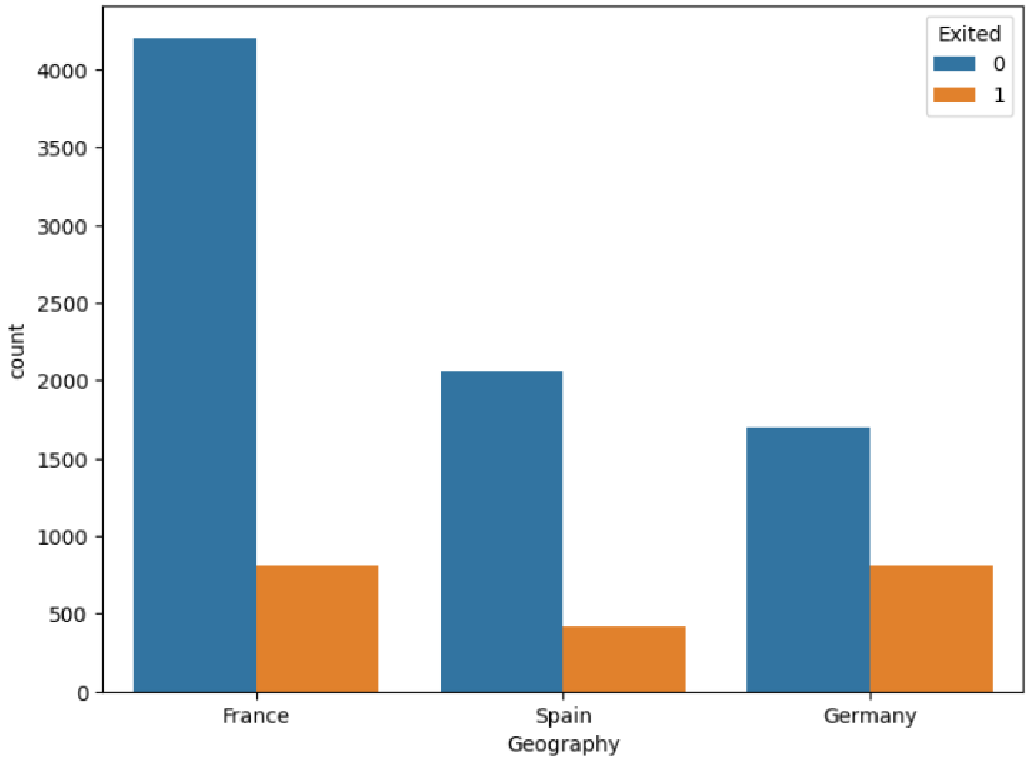


```
#It can be said that only AGE and Tenure coloum has significant difference and odd data to start our analysis
# As people with more tenure or has been with bank more is leaving rather than new one
```



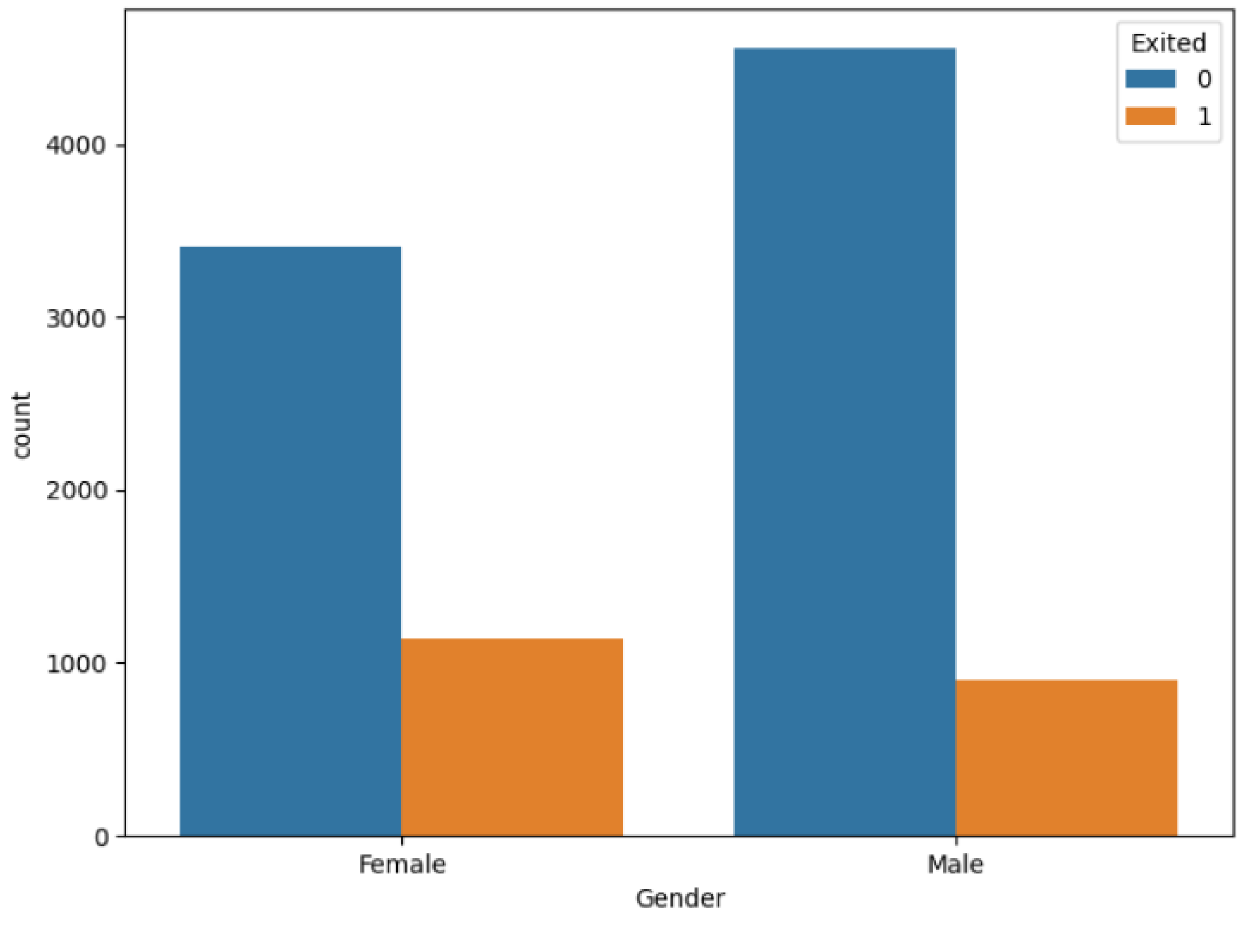
```
## 1 geography graph
    # It shows us that france has max population the data around 4000 and
    # churn value around 600 thus has min churn ratio

    # While germany has the higest churn ratio with min population around 1500
    # and churn value around 1000
```

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



```
## 2 Gender
# As found the churn/left ration in female is more than male
# Population of females are also less thn males with around 3500
# and that of males near 5000
```

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



2. Existing customers

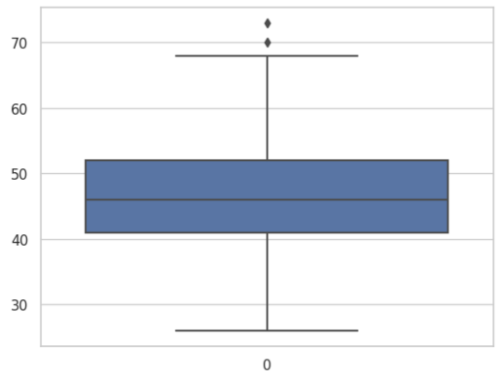
```
# So we will be needing to extract data from income_category columns and other categorical columns, since as you can see the data
# is in string format and the range is giving, not an exact value which can be feeded into the model directly.
```

```
bankdata.columns
```

```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio'],
      dtype='object')
```

```
# Understanding the data.

sb.set_theme(style="whitegrid")
sb.boxplot(bankdata['Customer_Age'])
```

```
<Axes: >
```



```
# So most of the customers are somewhere near 45 age, mean being 45 with some outliers that are depicted by the dots on the top
# side at age 70 and maybe 75 but there are not many outliers thus customer age is a decent coloum.
```

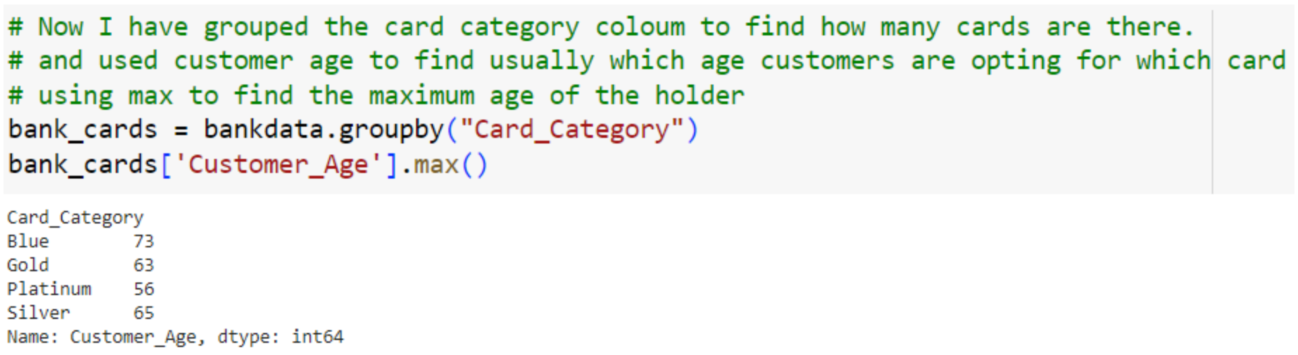
```
#So by using groupby on GENDER M & F then use aggeregate to find MEAN and COUNT of CREDIT LIMIT
bankdata[['Gender','Credit_Limit']].groupby('Gender').agg(['mean','count'])

# Thus found the CREDIT LIMIT of Females are significantly less then the Males
```

	Credit_Limit	
	mean	count
Gender		
F	5023.854274	5358
M	12685.674963	4769

```
#So by using groupby on GENDER M & F then use aggeregate to find MEAN and COUNT of Avg_Utilization_Ratio
bankdata[['Gender','Avg_Utilization_Ratio']].groupby('Gender').agg(['mean','count'])

# Average Utilization Ratio: it's how much you currently owe divided by your credit limit. It is generally expressed as a percent.
# For example, if you have a total of $10,000 in credit available on two credit cards, and a balance of $5,000 on one, your credit
# utilization rate is 50% — you're using half of the total credit you have available.
# A low credit utilization rate shows you're
# using less of your available credit.
```

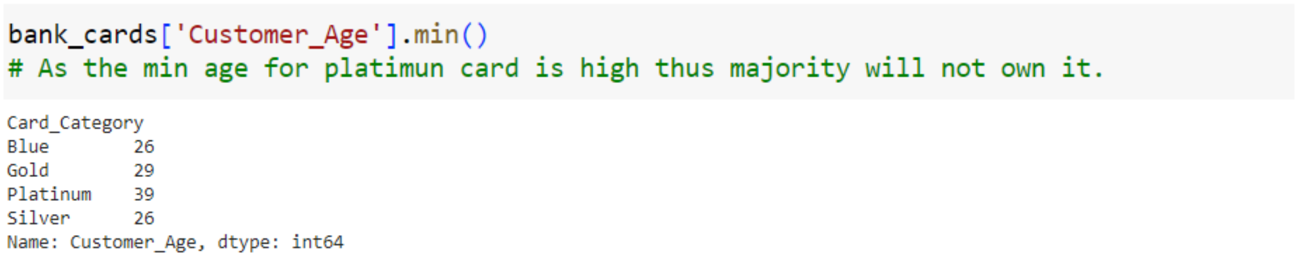
	Avg_Utilization_Ratio	
	mean	count
Gender		
F	0.341957	5358
M	0.199548	4769

```
#As expected the avg utilization of females are more than males but there credit limit is low as well.
```



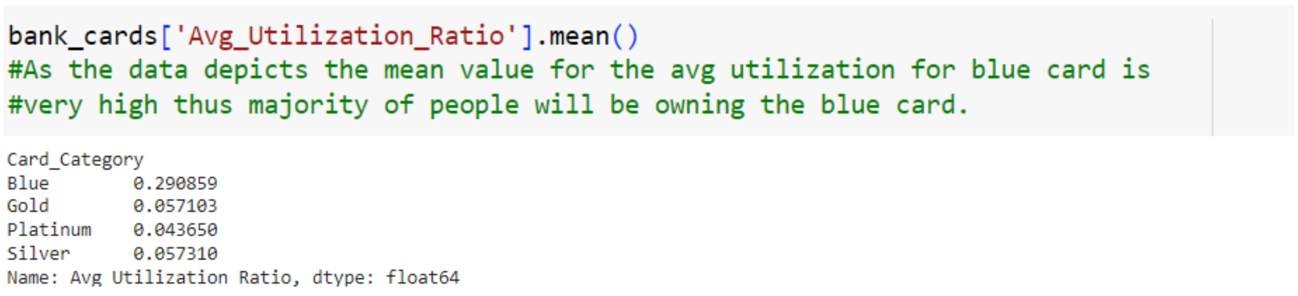
```
# Now I have grouped the card category coloum to find how many cards are there.
# and used customer age to find usually which age customers are opting for which card
# using max to find the maximum age of the holder
bank_cards = bankdata.groupby("Card_Category")
bank_cards['Customer_Age'].max()
```

```
Card_Category
Blue        73
Gold        63
Platinum    56
Silver      65
Name: Customer_Age, dtype: int64
```



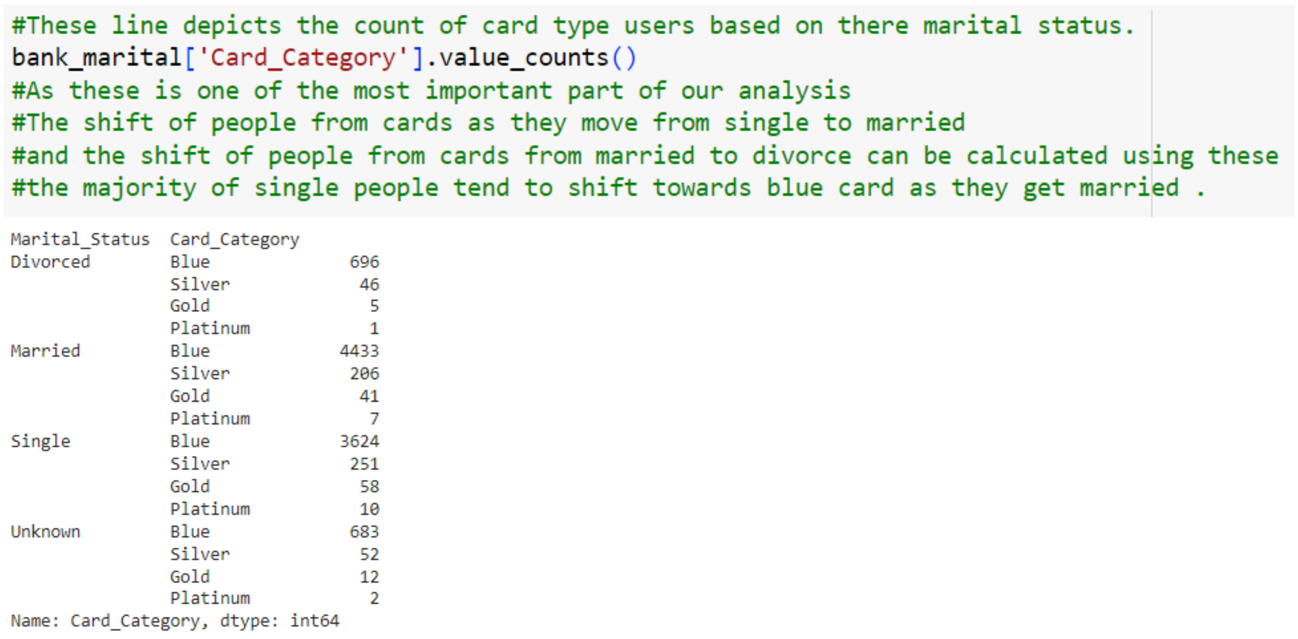
```
bank_cards['Customer_Age'].min()
# As the min age for platimun card is high thus majority will not own it.
```

```
Card_Category
Blue        26
Gold        29
Platinum    39
Silver      26
Name: Customer_Age, dtype: int64
```



```
bank_cards['Avg_Utilization_Ratio'].mean()
#As the data depicts the mean value for the avg utilization for blue card is
#very high thus majority of people will be owning the blue card.
```

```
Card_Category
Blue        0.290859
Gold        0.057103
Platinum    0.043650
Silver      0.057310
Name: Avg_Utilization_Ratio, dtype: float64
```



```
#These line depicts the count of card type users based on there marital status.
bank_marital['Card_Category'].value_counts()
#As these is one of the most important part of our analysis
#The shift of people from cards as they move from single to married
#and the shift of people from cards from married to divorce can be calculated using these
#the majority of single people tend to shift towards blue card as they get married .
```

```
Marital_Status  Card_Category
Divorced        Blue              696
                Silver             46
                Gold                5
                Platinum            1
Married         Blue             4433
                Silver            206
                Gold               41
                Platinum            7
Single          Blue             3624
                Silver            251
                Gold               58
                Platinum           10
Unknown         Blue              683
                Silver             52
                Gold               12
                Platinum            2
Name: Card_Category, dtype: int64
```

Model Training

→ Cleaning the Data

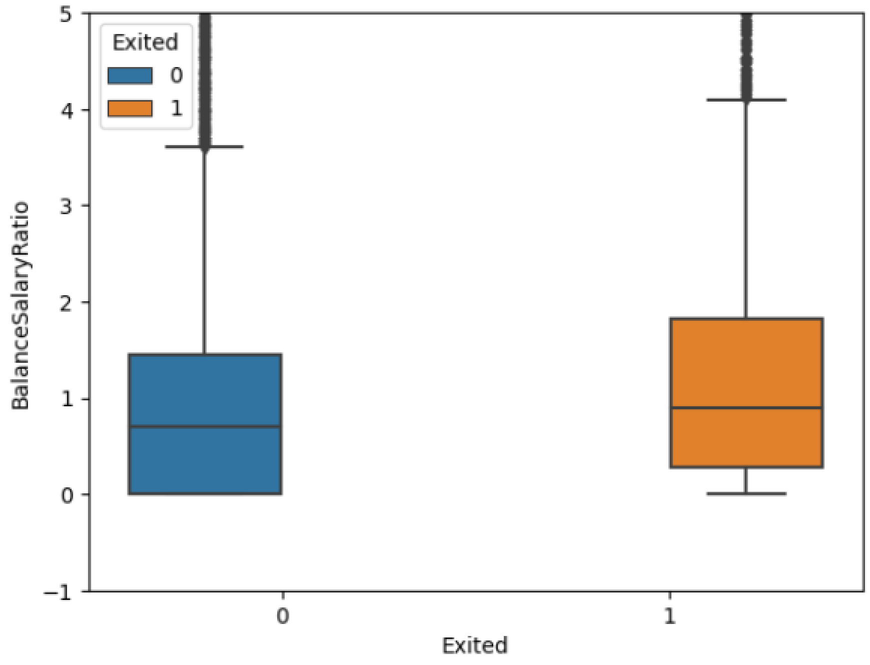
1. Attrited Customer / Churned Customer

```
## Feature Engineering
```

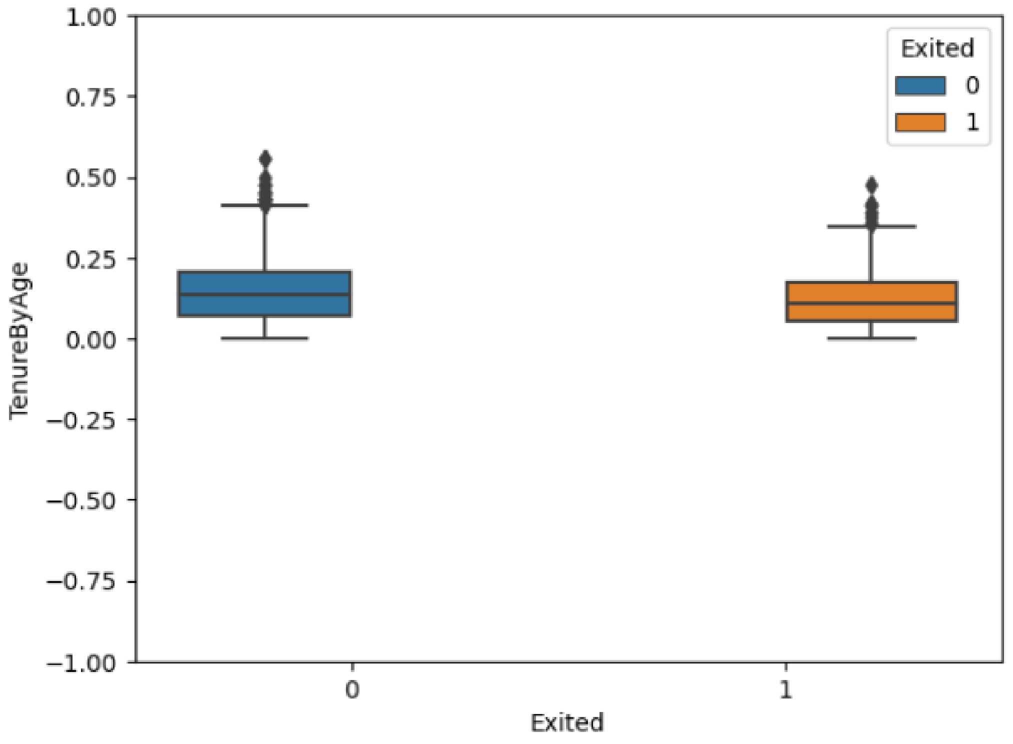
```
### Making a new column BalanceSalaryRatio
### As the Data with very Large variables can decrease the model accuracy
```

```
df['BalanceSalaryRatio'] = df.Balance/df.EstimatedSalary
sb.boxplot(y='BalanceSalaryRatio',x = 'Exited', hue = 'Exited',data = df)
plt.ylim(-1, 5)
```

```
(-1.0, 5.0)
```



```
df['TenureByAge'] = df.Tenure/(df.Age)
sb.boxplot(y='TenureByAge',x = 'Exited', hue = 'Exited',data = df)
plt.ylim(-1, 1)
plt.show()
```



```
#### Printing the categorical variables
```

```
def print_unique_col_values(df):
       for column in df:
            if df[column].dtypes=='object':
                print(f'{column}: {df[column].unique()}')
```

```
print_unique_col_values(df)
```

```
Geography: ['France' 'Spain' 'Germany']
Gender: ['Female' 'Male']
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	BalanceSalaryRatio	TenureByAge	Geography_France	Geography_Germany	Geography_Spain
0	619	0	42	2	0.00	1	1	1	101348.88	1	0.000000	0.047619	1	0	0
1	608	0	41	1	83807.86	1	0	1	112542.58	0	0.744677	0.024390	0	0	1
2	502	0	42	8	159660.80	3	1	0	113931.57	1	1.401375	0.190476	1	0	0
3	699	0	39	1	0.00	2	0	0	93826.63	0	0.000000	0.025641	1	0	0
4	850	0	43	2	125510.82	1	1	1	79084.10	0	1.587055	0.046512	0	0	1
5	645	1	44	8	113755.78	2	1	0	149756.71	1	0.759604	0.181818	0	0	1
6	822	1	50	7	0.00	2	1	1	10062.80	0	0.000000	0.140000	1	0	0

```
### Label Encoding
# As Caategorical Data cannot be passed on to our model , it must be converted to Numerical Data
```

```
df['Gender'].replace({'Male': 1,'Female': 0},inplace=True)
```

```
### One Hot Encoding method
```

```
#it will separate the coloums with 3 geographical location present
df1 = pd.get_dummies(data=df, columns=['Geography'])
df1.head(7)
```

	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	BalanceSalaryRatio	TenureByAge	Geography_France	Geography_Germany	Geography_Spain
0	0.538	0	0.324324	0.2	0.000000	0.000000	1	1	0.506735	1	0.000000	0.047619	1	0	0
1	0.516	0	0.310811	0.1	0.334031	0.000000	0	1	0.562709	0	0.744677	0.024390	0	0	1
2	0.304	0	0.324324	0.8	0.636357	0.666667	1	0	0.569654	1	1.401375	0.190476	1	0	0
3	0.698	0	0.283784	0.1	0.000000	0.333333	0	0	0.469120	0	0.000000	0.025641	1	0	0
4	1.000	0	0.337838	0.2	0.500246	0.000000	1	1	0.395400	0	1.587055	0.046512	0	0	1
5	0.590	1	0.351351	0.8	0.453394	0.333333	1	0	0.748797	1	0.759604	0.181818	0	0	1
6	0.944	1	0.432432	0.7	0.000000	0.333333	1	1	0.050261	0	0.000000	0.140000	1	0	0

2. Existing customers

```
bankdata['Gender'].value_counts()
```

```
F    5358
M    4769
Name: Gender, dtype: int64
```

```
bankdata['Education_Level'].value_counts()
```

```
Graduate         3128
High School      2013
Unknown          1519
Uneducated       1487
College          1013
Post-Graduate     516
Doctorate         451
Name: Education_Level, dtype: int64
```

```
bankdata['Marital_Status'].value_counts()
```

```
Married     4687
Single      3943
Unknown      749
Divorced     748
Name: Marital_Status, dtype: int64
```

```
# Converting Categorical data into numerical data.
```

```python
# Extracting data from the Income_Category column

X['Income_Category'].value_counts()
```

```
Less than $40K    3561
$40K - $60K       1790
$80K - $120K      1535
$60K - $80K       1402
Unknown           1112
$120K +            727
Name: Income_Category, dtype: int64
```

```python
X.head(3)
```

```
Income_Category ['$120K +' '$40K - $60K' '$60K - $80K' '$80K - $120K' 'Less than $40K'
 'Unknown']
Education_Level ['College' 'Doctorate' 'Graduate' 'High School' 'Post-Graduate'
 'Uneducated' 'Unknown']
Marital_Status ['Divorced' 'Married' 'Single' 'Unknown']
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level
0	768805383	0	45	1	3	3
1	818770008	0	49	0	5	2
2	713982108	0	51	1	3	2

3 rows × 21 columns

```python
X.describe()
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Months_on_book	Total_Relationship_Count	Mont
count	1.012700e+04	10127.0	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	
mean	7.391776e+08	0.0	46.325960	0.470919	2.346203	3.096574	1.463415	2.863928	35.928409	3.812580	
std	3.690378e+07	0.0	8.016814	0.499178	1.298908	1.834812	0.737808	1.504700	7.986416	1.554408	
min	7.080821e+08	0.0	26.000000	0.000000	0.000000	0.000000	0.000000	0.000000	13.000000	1.000000	
25%	7.130368e+08	0.0	41.000000	0.000000	1.000000	2.000000	1.000000	2.000000	31.000000	3.000000	
50%	7.179264e+08	0.0	46.000000	0.000000	2.000000	3.000000	1.000000	3.000000	36.000000	4.000000	
75%	7.731435e+08	0.0	52.000000	1.000000	3.000000	5.000000	2.000000	4.000000	40.000000	5.000000	
max	8.283431e+08	0.0	73.000000	1.000000	5.000000	6.000000	3.000000	5.000000	56.000000	6.000000	

```python
X = X.drop(['CLIENTNUM', 'Card_Category'], axis = 1)
```

```python
X.shape
```

```
(10127, 19)
```

→Model Building and Prediction

1. Attrited Customer / Churned Customer

```
## The Sequential model
```

```
### A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.
```

```
## A Sequential model is not appropriate when:

#-Your model has multiple inputs or multiple outputs
#-Any of your layers has multiple inputs or multiple outputs
#-You need to do layer sharing
#-You want non-linear topology (e.g. a residual connection, a multi-branch model)
```

```python
import tensorflow as tf
from tensorflow import keras


model = keras.Sequential([
    keras.layers.Dense(12 , input_shape=(None,32,14), activation='relu'),
    keras.layers.Dense(6, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])


model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100)
```

```
250/250 [==============================] - 1s 3ms/step - loss: 0.3568 - accuracy: 0.8525
Epoch 73/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3485 - accuracy: 0.8541
Epoch 74/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3469 - accuracy: 0.8549
Epoch 75/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3483 - accuracy: 0.8537
Epoch 76/100
250/250 [==============================] - 1s 2ms/step - loss: 0.4300 - accuracy: 0.8426
Epoch 77/100
Epoch 99/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3434 - accuracy: 0.8576
Epoch 100/100
250/250 [==============================] - 1s 3ms/step - loss: 0.3435 - accuracy: 0.8571
<keras.src.callbacks.History at 0x7c656f019f30>
```

```
model.evaluate(X_test, y_test)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.3549 - accuracy: 0.8505
[0.354936808347702, 0.8504999876022339]
```

```
yp = model.predict(X_test)
yp
```

```
63/63 [==============================] - 0s 4ms/step
array([[0.02784799],
       [0.0834761 ],
       [0.05151472],
       ...,
       [0.01324045],
       [0.01115167],
       [0.06177692]], dtype=float32)
```

```
### Converting our predictions to 0,1 to check accuracy
```

```
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

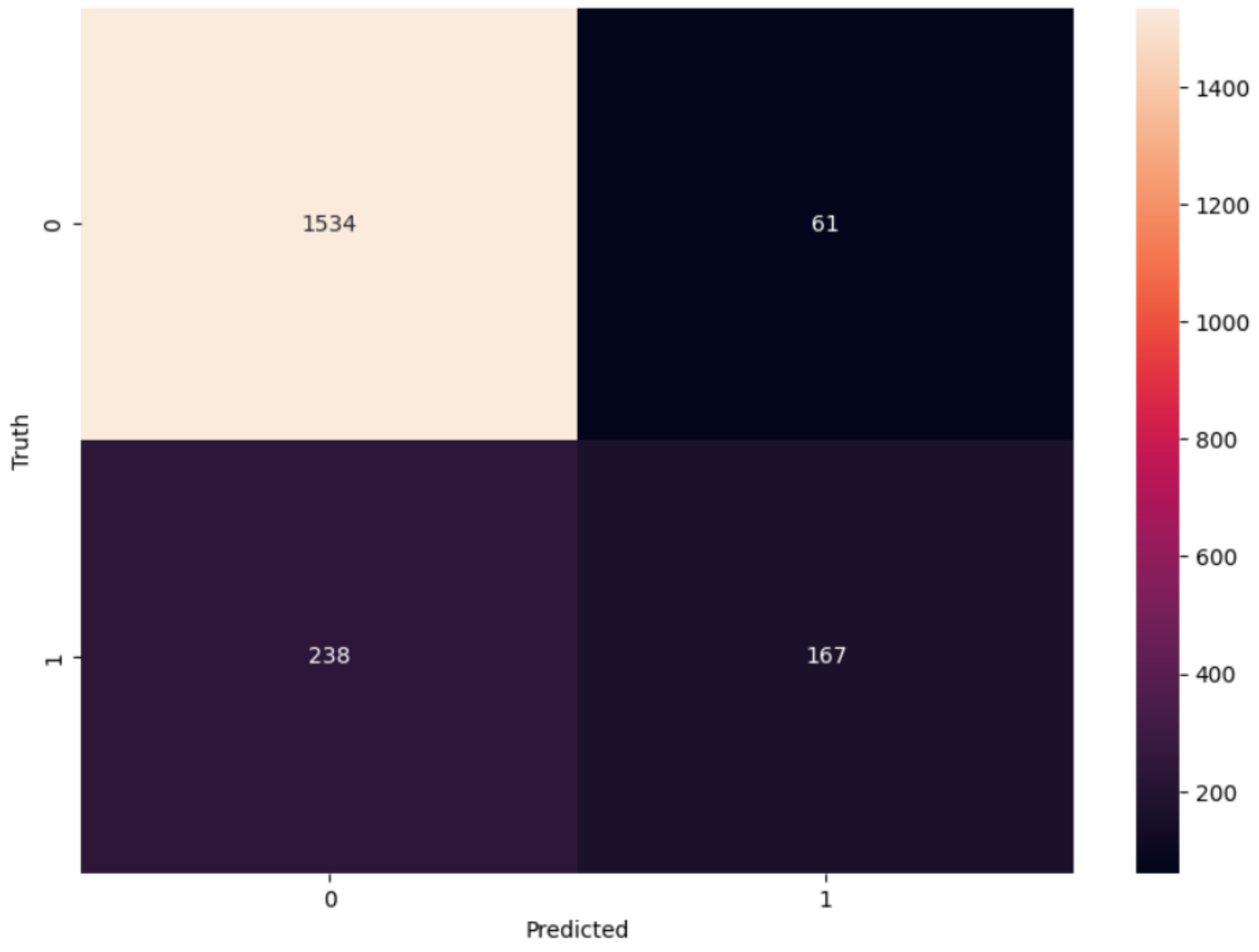
```
### Checking the accuracy
```

```
from sklearn.metrics import confusion_matrix , classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91      1595
           1       0.73      0.41      0.53       405

    accuracy                           0.85      2000
   macro avg       0.80      0.69      0.72      2000
weighted avg       0.84      0.85      0.83      2000
```

```
import seaborn as sn
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(95.72222222222221, 0.5, 'Truth')
```



```
from sklearn.metrics import accuracy_score
```

```
print("Accuracy score is: ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy score is:  85.05 %
```

2. Existing customers

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 7)
pca2 = PCA(n_components = 10)
pca_fit = pca.fit_transform(X)
pca_fit2 = pca2.fit_transform(X)
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
```

```
# You can use both of the reduced dataset to see how much accuracy or predictive power of the model is affected.

Xtrain, Xtest, ytrain, ytest = train_test_split(pca_fit2, y, test_size=0.2, random_state = 42)
```

```
random_model = RandomForestClassifier(n_estimators=300, n_jobs = -1)
```

```
Xtrain, Xtest, ytrain, ytest = train_test_split(pca_fit2, y, test_size=0.2, random_state = 42)
```

```
random_model = RandomForestClassifier(n_estimators=300, n_jobs = -1)
```

```
#Fit
random_model.fit(Xtrain, ytrain)

y_pred = random_model.predict(Xtest)

#Checking the accuracy
random_model_accuracy = round(random_model.score(Xtrain, ytrain)*100,2)
print(round(random_model_accuracy, 2), '%')
```

```
100.0 %
```

```
random_model_accuracy1 = round(random_model.score(Xtest, ytest)*100,2)
print(round(random_model_accuracy1, 2), '%')
```

```
95.61 %
```