Capstone Project

Credit Card Churn Prediction Report

Shubham Harsh

Problem Statement: EXL's Credit Card Analytics Division faces declining customer retention. This project aims to predict churn using behavioral and demographic data (Gender, Age, Tenure, Balance, etc.), where churn is defined as no transactions for 3+ months, a 50%+ drop in spending, and marked as "Churn = Yes."

Data Understanding:

- **Customer churn** refers to **when a customer stops using a company's product or service**.
- Dataset describes customer's physical attributes, alongwith their financial attributes, signifying how person uses his/her Credit Card.
- The goal of this project is to predict whether a **Customer** will churn or stay based on their characteristics and account activity.
- This is a classic binary classification problem in machine learning:
  - Churn = **1** means customer left
  - Churn = **0** means customer stayed

```
CustomerID          object
Gender              object
Age                float64
Tenure             float64
Balance            float64
NumOfProducts      float64
HasCrCard           object
IsActiveMember      object
EstimatedSalary    float64
Churn               object
dtype: object
```

	CustomerID	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Churn
0	CUST0001	Male	56.0	4.0	0.00	4.0	0.0	0.0	40282.42	1.0
1	CUST0002	NaN	28.0	8.0	67408.01	4.0	0.0	1.0	27333.51	0.0
2	CUST0003	Female	47.0	6.0	1154.97	1.0	0.0	1.0	99514.91	1.0
3	CUST0004	Male	42.0	1.0	0.00	2.0	1.0	1.0	146588.22	0.0
4	CUST0005	Male	64.0	3.0	77109.94	4.0	0.0	0.0	131792.25	0.0

- Columns were filled with NULL values.
- Severe inconsistencies were there like:
    - Gender has inconsistent casing and spacing: 'Male', 'MALE', ' male ', 'Female', 'FEMALE', ' Female’.
    - HasCrCard includes mixed types and invalid value: '0.0', '1.0', '2.0', 'Yes'.
    - IsActiveMember contains unexpected values: '0.0', '1.0', '-1', 'No', '-1.0'.
    - Churn includes invalid entries: '1.0', '0.0', 'Maybe', '2.0', '2’.
- Age column (numerical in nature), had max value of 120 and min value of -5, mean was appx. 43 years.

```
Column: Gender (Type: object)
Unique Values (6): ['Male' 'Female' 'FEMALE' ' male ' 'MALE' ' Female']

Column: HasCrCard (Type: object)
Unique Values (4): ['0.0' '1.0' '2.0' 'Yes']

Column: IsActiveMember (Type: object)
Unique Values (5): ['0.0' '1.0' '-1' 'No' '-1.0']

Column: Churn (Type: object)
Unique Values (5): ['1.0' '0.0' 'Maybe' '2.0' '2']
```

```
df.isnull().sum()
[23]
...
Gender             6
Age                2
Tenure             3
Balance            4
NumOfProducts      4
HasCrCard          2
IsActiveMember     5
EstimatedSalary    1
Churn              0
dtype: int64
```

Data Cleaning:

- Upon observing, Gender column had whitespaces around them.
- Replaced all instances of Male and Female with 'Male' and 'Female' respectively.
- For 'HasCrCard' attribute, replaced all values > 0 as Yes, otherwise a No.
- For 'IsActiveMember' attribute, for all the string having positive values were treated as Yes otherwise No.
- For 'Churn' as it was target attribute, all the rows where values where not in terms of 0 or 1. All such rows were dropped.

```
df['Gender'] = df['Gender'].str.strip().str.lower()  # remove spaces and lowercase
df['Gender'] = df['Gender'].replace({
    'male': 'Male',
    'female': 'Female'
})
```

```
# Convert all values to string first
df['HasCrCard'] = df['HasCrCard'].astype(str).str.strip()

# Map valid values only
df['HasCrCard'] = df['HasCrCard'].replace({
    '1.0': 1, '0.0': 0, 'Yes': 1, 'No': 0, '2.0': 1  # if 2.0 means yes
})

df['HasCrCard'] = df['HasCrCard'].astype(int)
```


```
# Convert all values to string first
df['HasCrCard'] = df['HasCrCard'].astype(str).str.strip()

# Map valid values only
df['HasCrCard'] = df['HasCrCard'].replace({
    '1.0': 1, '0.0': 0, 'Yes': 1, 'No': 0, '2.0': 1  # if 2.0 means yes
})

df['HasCrCard'] = df['HasCrCard'].astype(int)
```

```
print(df['Age'].describe())
```

[5]

```
count    1001.000000
mean       43.785215
std        15.895560
min        -5.000000
25%        31.000000
50%        43.000000
75%        56.000000
max       120.000000
Name: Age, dtype: float64
```

As it's credit card, user must be between 18 and 100(max life exp. assumed)

```
df = df[(df['Age'] >= 18) & (df['Age'] <= 100)]
```

[86]

Drop rows where 'Churn' is missing (target variable)

```
# Drop rows where 'Churn' is missing (target variable)
df.dropna(subset=['Churn'], inplace=True)
```

2]

Data Preparation:

- Filled the NaN values in numerical columns with mean, and in categorical columns with mode.
- Gender column had whitespaces around them, so we have to remove them, alongwith that all inconsistencies of male and female were mapped as 'Male' and 'Female'.
- For each binary categorised column, if the string value was positive, it was assigned a value of 1 (yes), otherwise a value of 0 (no) was assigned.
- For the target column 'Churn', the values which were absurd like "-1", "Maybe", etc. Those rows were dropped.
- For age column, upon observation, it was found that age ranges between -5 to 120. So, the outliers were removed by taking only those rows where the range was in between 18 and 100.
- CustomerId column was dropped as these were just identifiers.

Filling null values(mean for numeric cols, median for categorical cols.)

```
# Fill numeric columns with median
num_cols = ['Age', 'Tenure', 'Balance', 'NumOfProducts', 'EstimatedSalary']
for col in num_cols:
    df[col] = df[col].fillna(df[col].median())

# Fill categorical columns with mode
cat_cols = ['Gender', 'HasCrCard', 'IsActiveMember']
for col in cat_cols:
    df[col] = df[col].fillna(df[col].mode()[0])
```

[73]

Cleaning and standardising categorical columns

Upon observing, was found that values inside Gender column has whitespaces around them.

```
df['Gender'] = df['Gender'].str.strip().str.lower()  # remove spaces and lowercase
df['Gender'] = df['Gender'].replace({
    'male': 'Male',
    'female': 'Female'
})
```

```python
# Convert all values to string first
df['HasCrCard'] = df['HasCrCard'].astype(str).str.strip()

# Map valid values only
df['HasCrCard'] = df['HasCrCard'].replace({
    '1.0': 1, '0.0': 0, 'Yes': 1, 'No': 0, '2.0': 1  # if 2.0 means yes
})

df['HasCrCard'] = df['HasCrCard'].astype(int)
```

[78]

```python
df['IsActiveMember'] = df['IsActiveMember'].astype(str).str.strip()

df['IsActiveMember'] = df['IsActiveMember'].replace({
    '1.0': 1, '0.0': 0, '-1': 0, '-1.0': 0, 'No': 0, 'Yes': 1
}).astype(int)
```

]

```python
df['Churn'] = df['Churn'].astype(str).str.strip()

# Drop rows that are not 0 or 1 (e.g., "Maybe", "2.0")
df = df[df['Churn'].isin(['0.0', '1.0'])]

# Now map to integers
df['Churn'] = df['Churn'].replace({'1.0': 1, '0.0': 0}).astype(int)
```

As it's credit card, user must be between 18 and 100(max life exp. assumed)

Generate + Code + Markdown

```python
df = df[(df['Age'] >= 18) & (df['Age'] <= 100)]
```

6]

One Hot Encoding:

- Gender was the only categorical column, left after all cleaning and preprocessing.
- So we applied one hot encoding on it, for better convergence of random forest model.

After standardizing and cleaning, only Gender attribute was found to be categorical

```
# Encoding categorical features
df = pd.get_dummies(df, columns=['Gender'], drop_first=True)
```

[82]

Generate + Code + Markdown

	CustomerID	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Churn	Gender_Male
0	CUST0001	56.0	4.0	0.00	4.0	0	0	40282.42	1	True
1	CUST0002	28.0	8.0	67408.01	4.0	0	1	27333.51	0	False
2	CUST0003	47.0	6.0	1154.97	1.0	0	1	99514.91	1	False
3	CUST0004	42.0	1.0	0.00	2.0	1	1	146588.22	0	True
4	CUST0005	64.0	3.0	77109.94	4.0	0	0	131792.25	0	True

Data Standardisation:

- Applied Min-max scaling for better convergence of RandomForest Algorithm.
- Min-Max scaling was applied on all columns of training part.

```
# Split features and target variable
X = df.drop('Churn', axis=1)
y = df['Churn']
```

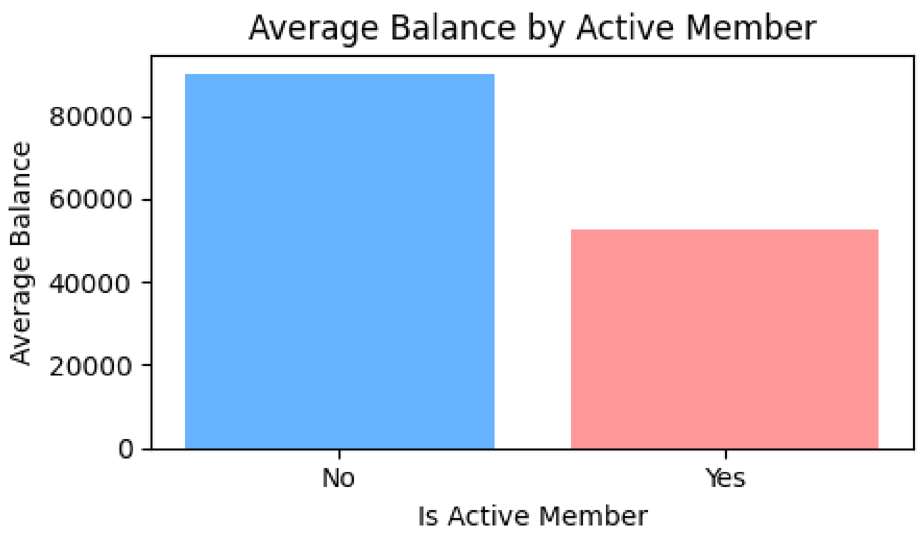
[92]

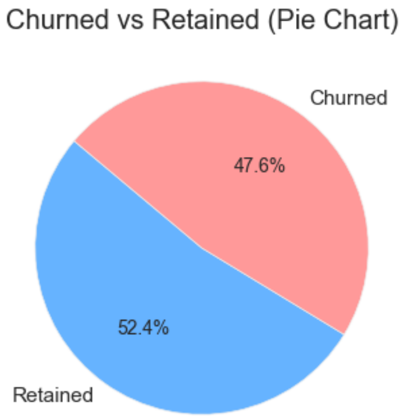
MinMax Scaling

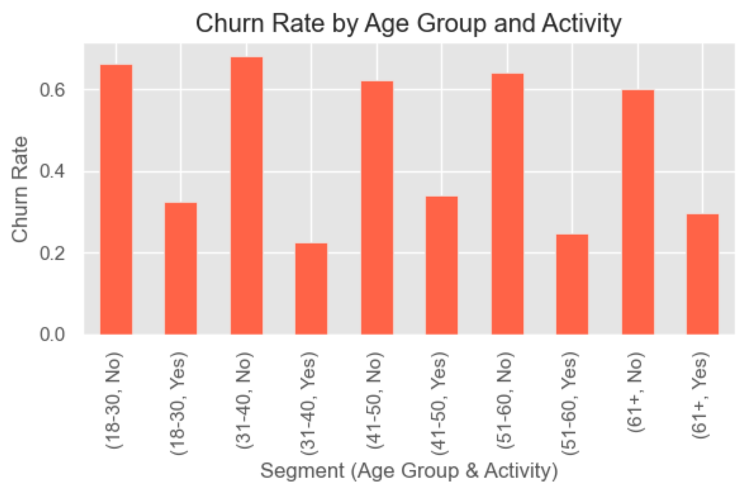
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

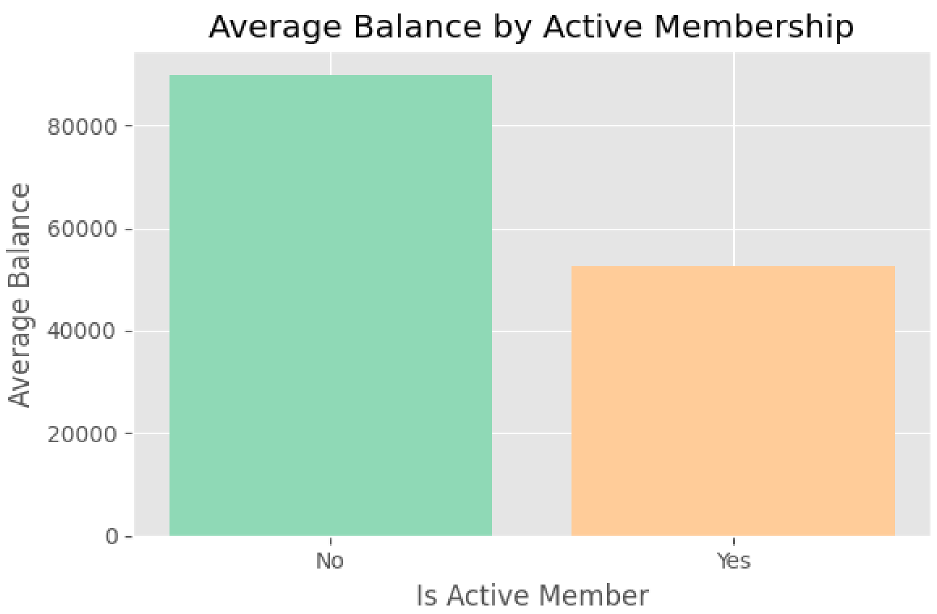
[93]

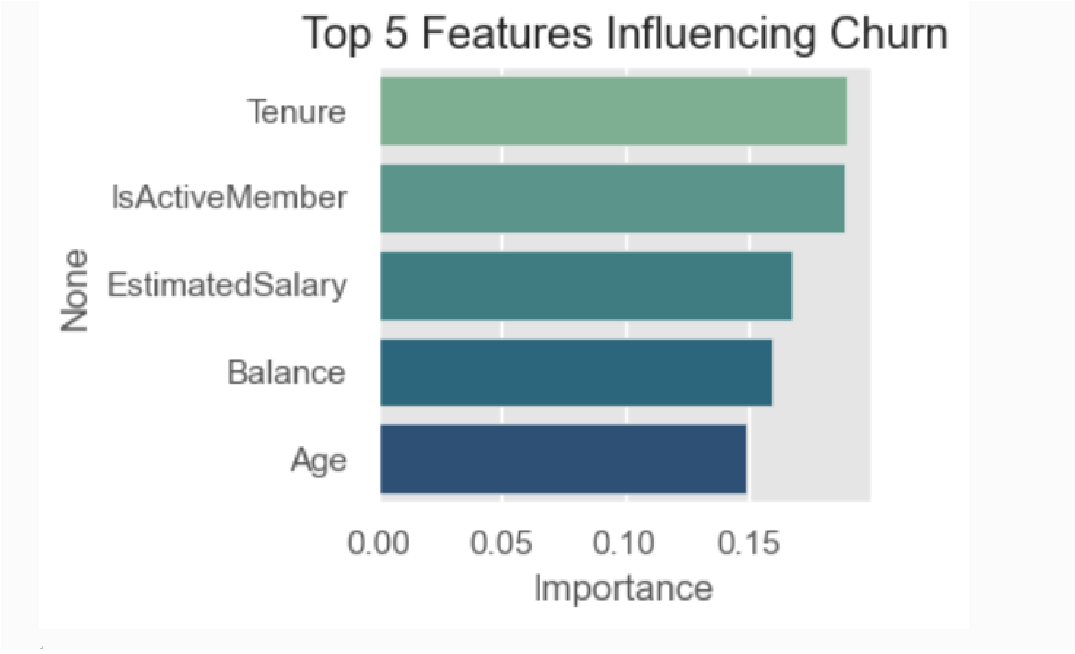
Visualisations:

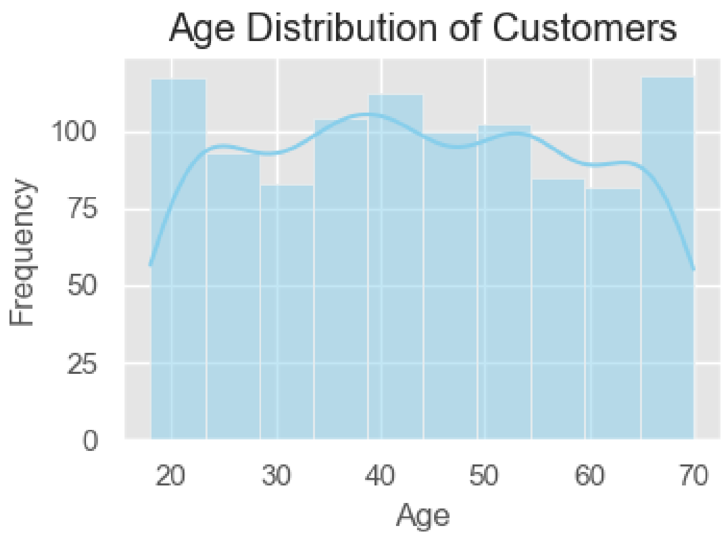


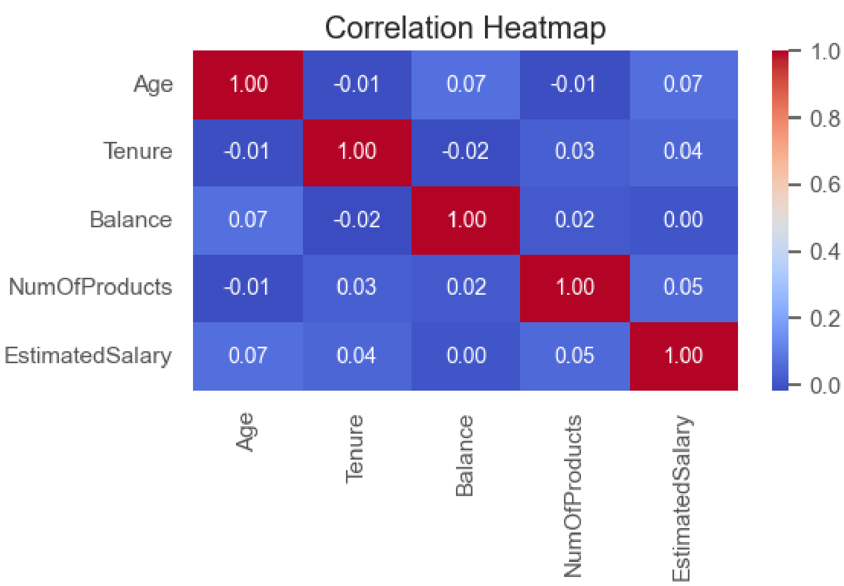







Top 5 Features Influencing Churn
Tenure
IsActiveMember
EstimatedSalary
Balance
Age
None
0.00
0.05
0.10
0.15
Importance


Age Distribution of Customers
Frequency
100
75
50
25
0
20
30
40
50
60
70
Age


Correlation Heatmap
Age
Tenure
Balance
NumOfProducts
EstimatedSalary
1.00
-0.01
0.07
-0.01
0.07
-0.01
1.00
-0.02
0.03
0.04
0.07
-0.02
1.00
0.02
0.00
-0.01
0.03
0.02
1.00
0.05
0.07
0.04
0.00
0.05
1.00
1.0
0.8
0.6
0.4
0.2
0.0

Training:

Test Train Split on DataSet:

- Cleaned dataset was saved.
- On that dataset 20% was used as test data, rest as training data.
- Stratify was used to ensure the class balance in both train and test data set.
- Applied RandomForest on training data.
- Results were these:

```
Confusion Matrix:
 [[78 27]
 [36 59]]

Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.74      0.71       105
           1       0.69      0.62      0.65        95

    accuracy                           0.69       200
   macro avg       0.69      0.68      0.68       200
weighted avg       0.69      0.69      0.68       200

Accuracy: 68.5000
```

- Applied GridSearchCV with randomforest classifier, for hyper parameter tuning.
- Accuracy was increased by 3.5%

```
Confusion Matrix:
 [[78 27]
 [29 66]]

Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.74      0.74       105
           1       0.71      0.69      0.70        95

    accuracy                           0.72       200
   macro avg       0.72      0.72      0.72       200
weighted avg       0.72      0.72      0.72       200

Accuracy: 72.0000
```

Probable Insights:

- A significant portion of churned customers were inactive members.
- Customers with **fewer products** were more likely to churn.
- Younger customers showed higher churn tendencies compared to older ones.
- Customers with lower tenure (newer customers) were more prone to churn.
- High balance with low activity may indicate disengaged customers at risk.