Churn_Rate_Analysis_by_Diptyait_Das

June 2, 2024

0.1 Analyzing Bank Customer Churn Rates

0.2 Problem Statement

In the rapidly evolving banking sector, customer retention has become a critical concern. Banks are increasingly seeking to understand the factors that influence customer decisions to stay with or leave their banking service provider. This project focuses on analyzing a dataset containing various attributes of bank customers to identify key predictors of customer churn. By leveraging data analytics, we aim to uncover patterns and insights that could help devise strategies to enhance customer retention and reduce churn rates.

0.3 Data Description

The dataset contains various attributes related to bank customers, which are used to analyze customer churn.

- **RowNumber**: Corresponds to the record (row) number and has no effect on the output.
- **CustomerId**: Contains random values and has no effect on customer leaving the bank.
- **Surname**: The surname of a customer has no impact on their decision to leave the bank.
- **CreditScore**: Can have an effect on customer churn, since a customer with a higher credit score is less likely to leave the bank.
- **Geography**: A customer's location can affect their decision to leave the bank.
- **Gender**: It's interesting to explore whether gender plays a role in a customer leaving the bank.
- **Age**: This is certainly relevant, since older customers are less likely to leave their bank than younger ones.
- **Tenure**: Refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.
- **Balance**: Also a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.
- **NumOfProducts**: Refers to the number of products that a customer has purchased through the bank.
- **HasCrCard**: Denotes whether or not a customer has a credit card. This column is also relevant, since people with a credit card are less likely to leave the bank.
- **IsActiveMember**: Active customers are less likely to leave the bank.
- **EstimatedSalary**: As with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
- **Exited**: Whether or not the customer left the bank.
- **Complain**: Customer has complaint or not.
- **Satisfaction Score**: Score provided by the customer for their complaint resolution.

- **Card Type**: Type of card held by the customer.
- **Points Earned**: The points earned by the customer for using credit card.

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind,chi2_contingency,sem,t,randint
import warnings
warnings.simplefilter('ignore')
```

[2]:
```python
df = pd.read_csv('Bank-Records.csv',index_col=0).
    ↪drop(columns=['CustomerId','Surname'])
```

[3]:
```python
df.shape
```

[3]: (10000, 15)

0.4 10000 Rows and 15 Columns.

0.4.1 Segmentation of numerical features into groups helps us to identify which groups are more likely to churn.

0.4.2 Since target variable `Exited` is categorical, segmenting numerical columns into categorical columns allows us to use Chi-squared test of independence.

0.4.3 Binning

[4]:
```python
df['Tenurebin']=pd.
    ↪cut(df['Tenure'],bins=[float('-inf'),2,4,6,8,float('inf')],labels=['<2','2-4','4-6','6-8','
df['Agebin']=pd.
    ↪cut(df['Age'],bins=[float('-inf'),20,40,60,80,float('inf')],labels=['<20','20-40','40-60','
df['Salarybin']=pd.
    ↪cut(df['EstimatedSalary'],bins=[float('-inf'),50000,75000,100000,125000,float('inf')],label
df['Balancebin']=pd.
    ↪cut(df['Balance'],bins=[float('-inf'),30000,60000,90000,120000,float('inf')],labels=['<30k'
df['Pointsbin']=pd.cut(df['Point␣
    ↪Earned'],bins=[float('-inf'),400,500,600,700,float('inf')],labels=['<400','400-500','500-60
df['Creditbin']=pd.
    ↪cut(df['CreditScore'],bins=[float('-inf'),400,500,600,700,float('inf')],labels=['<400','400
```

[5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 1 to 10000
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CreditScore         10000 non-null  int64
```

```
 1   Geography            10000 non-null  object
 2   Gender               10000 non-null  object
 3   Age                  10000 non-null  int64
 4   Tenure               10000 non-null  int64
 5   Balance              10000 non-null  float64
 6   NumOfProducts        10000 non-null  int64
 7   HasCrCard            10000 non-null  int64
 8   IsActiveMember       10000 non-null  int64
 9   EstimatedSalary      10000 non-null  float64
 10  Exited               10000 non-null  int64
 11  Complain             10000 non-null  int64
 12  Satisfaction Score   10000 non-null  int64
 13  Card Type            10000 non-null  object
 14  Point Earned         10000 non-null  int64
 15  Tenurebin            10000 non-null  category
 16  Agebin               10000 non-null  category
 17  Salarybin            10000 non-null  category
 18  Balancebin           10000 non-null  category
 19  Pointsbin            10000 non-null  category
 20  Creditbin            10000 non-null  category
dtypes: category(6), float64(2), int64(10), object(3)
memory usage: 1.3+ MB
```

[6]:

```
bool=['HasCrCard','IsActiveMember','Complain']
num_cat=['Satisfaction Score','NumOfProducts']
num=['Balance','EstimatedSalary','Point Earned','CreditScore']
cat=['Geography','Gender','Card␣
 ↪Type','Agebin','Tenurebin','Salarybin','Balancebin','Pointsbin','Creditbin']
```

0.5 No missing values

[7]:

```
df.isna().sum()
```

[7]:

```
CreditScore           0
Geography             0
Gender                0
Age                   0
Tenure                0
Balance               0
NumOfProducts         0
HasCrCard             0
IsActiveMember        0
EstimatedSalary       0
Exited                0
Complain              0
Satisfaction Score    0
Card Type             0
```

```
Point Earned          0
Tenurebin             0
Agebin                0
Salarybin             0
Balancebin            0
Pointsbin             0
Creditbin             0
dtype: int64
```

0.6 Univariate

0.6.1 Categorical

[8]:
```
categorical_cols = bool + num_cat + cat
print('MODE')
for col in categorical_cols:
    mode = df[col].mode()
    print(f'{col} : {mode[0]}')
```

```
MODE
HasCrCard : 1
IsActiveMember : 1
Complain : 0
Satisfaction Score : 3
NumOfProducts : 1
Geography : France
Gender : Male
Card Type : DIAMOND
Agebin : 20-40
Tenurebin : <2
Salarybin : >125k
Balancebin : <30k
Pointsbin : >700
Creditbin : 600-700
```

0.6.2 Numerical

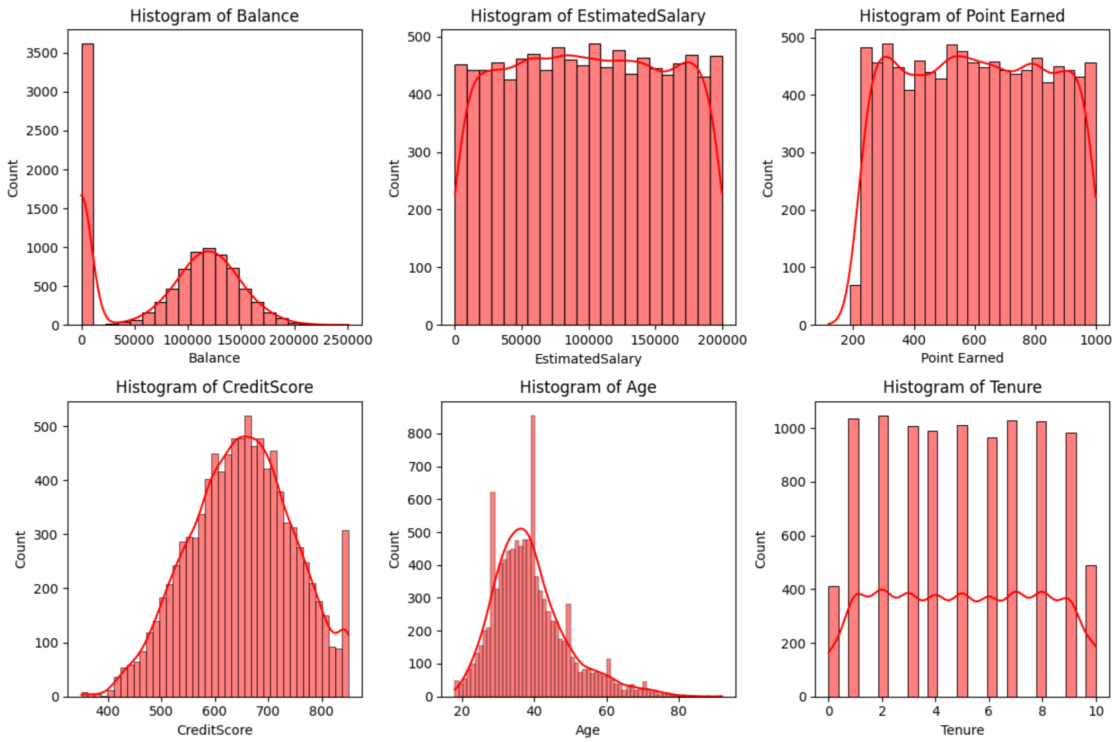
[9]:
```
numerical_cols=num+['Age','Tenure']
def confidence_interval(data, confidence=0.95):
    n = len(data)
    mean = np.mean(data)
    _sem = sem(data)
    margin_of_error = _sem * t.ppf((1 + confidence) / 2., n-1)
    return mean - margin_of_error, mean + margin_of_error
for col in numerical_cols+['Exited']:
    mean = df[col].mean()
    median = df[col].median()
    ci_lower, ci_upper = confidence_interval(df[col].dropna())
```

```python
    print(f"Column: {col}")
    print(f"Mean: {mean}")
    print(f"Median: {median}")
    print(f"95% Confidence Interval: [{ci_lower}, {ci_upper}]\n")
```

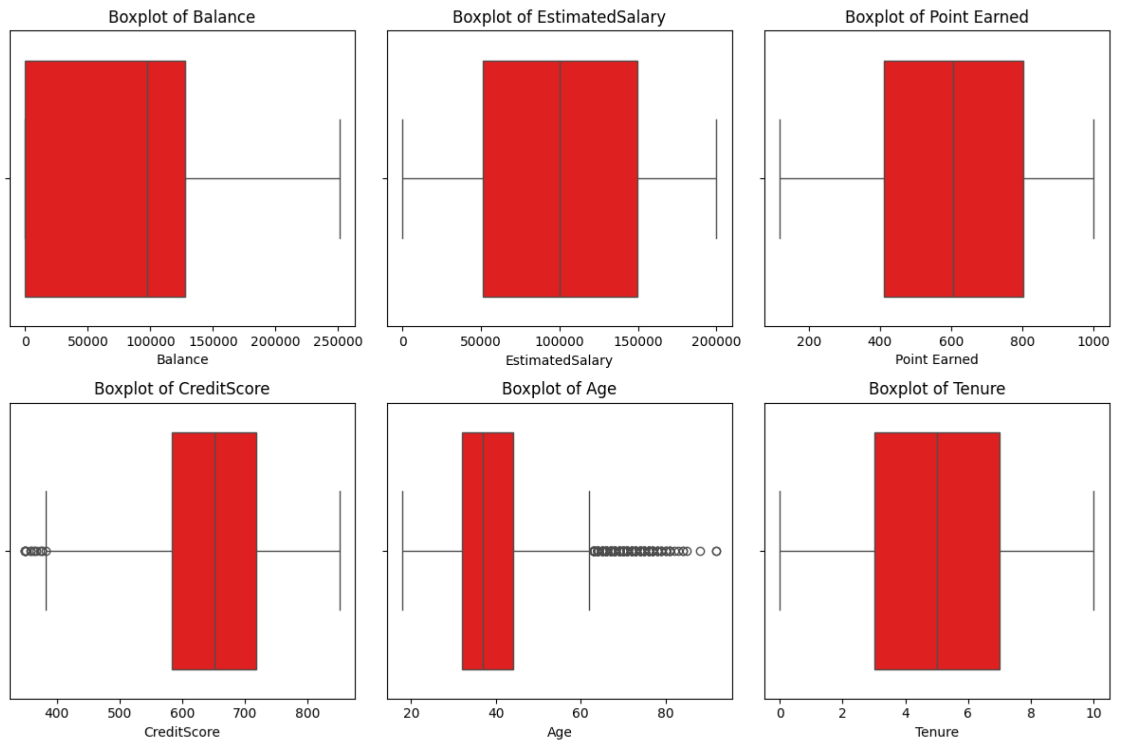
```
Column: Balance
Mean: 76485.889288
Median: 97198.54000000001
95% Confidence Interval: [75262.77456275589, 77709.00401324412]

Column: EstimatedSalary
Mean: 100090.239881
Median: 100193.915
95% Confidence Interval: [98962.9184740726, 101217.5612879274]

Column: Point Earned
Mean: 606.5151
Median: 605.0
95% Confidence Interval: [602.0865184468312, 610.9436815531687]

Column: CreditScore
Mean: 650.5288
Median: 652.0
95% Confidence Interval: [648.6342008168427, 652.4233991831574]

Column: Age
Mean: 38.9218
Median: 37.0
95% Confidence Interval: [38.716217885407524, 39.12738211459247]

Column: Tenure
Mean: 5.0128
Median: 5.0
95% Confidence Interval: [4.95610756131494, 5.069492438685061]

Column: Exited
Mean: 0.2038
Median: 0.0
95% Confidence Interval: [0.19590348331978494, 0.21169651668021508]
```

[10]:
```python
fig, axes = plt.subplots(2, 3, figsize=(12, 8))
axes = axes.flatten()
for i, col in enumerate(numerical_cols):
    sns.histplot(data=df, x=col, ax=axes[i], color='Red',kde=True)
    axes[i].set_title(f'Histogram of {col}')
```

```
plt.tight_layout()
plt.show()
```



[11]:
```
fig, axes = plt.subplots(2, 3, figsize=(12, 8))
axes = axes.flatten()
for i, col in enumerate(numerical_cols):
    sns.boxplot(data=df, x=col, ax=axes[i], color='Red')
    axes[i].set_title(f'Boxplot of {col}')

plt.tight_layout()
plt.show()
```



0.6.3 Except `Age` and `CreditScore` all other numerical columns are without any outliers.

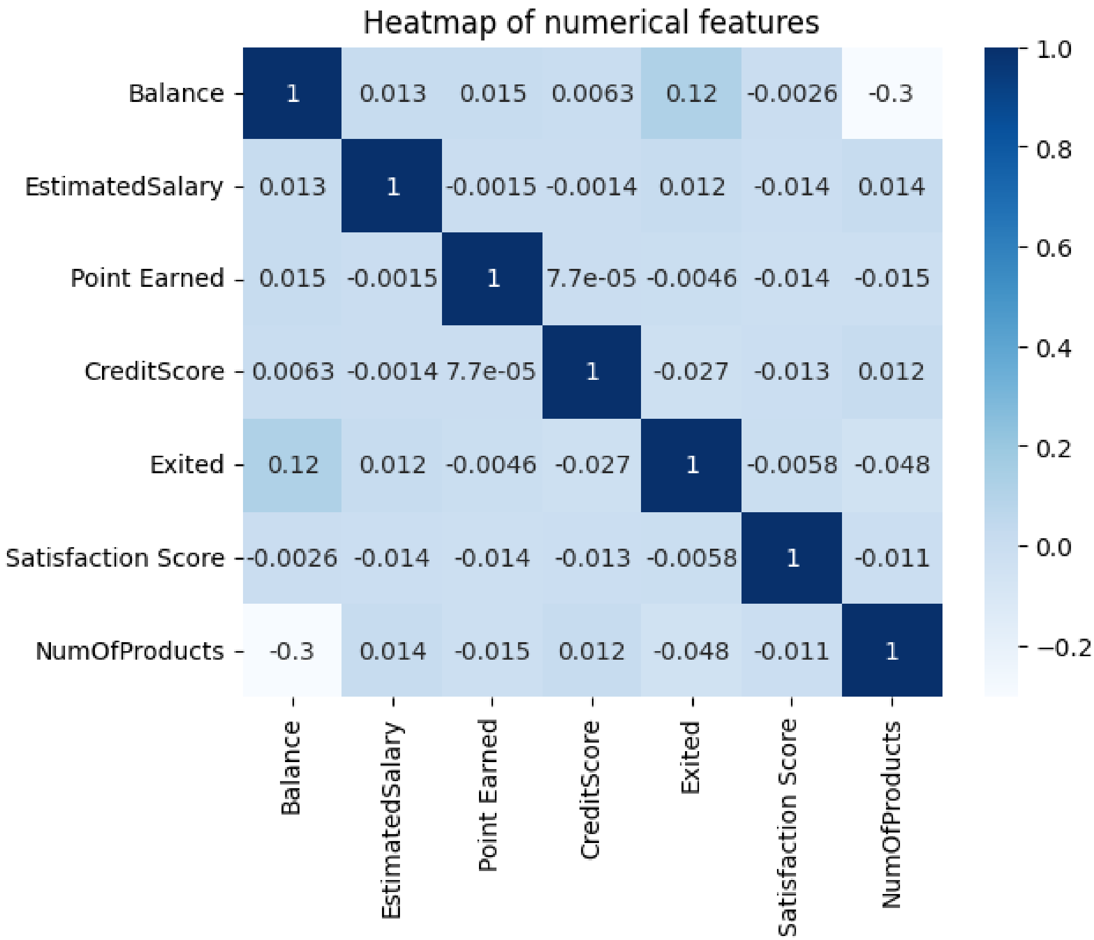
0.6.4 `Balance` and `CreditScore` is left skewed whereas `Age` is right skewed.

0.6.5 `EstimatedSalary` , `Point Earned` and `Tenure` are normally distributed.

0.7 Bivariate

0.8 Correlation

[12]:
```
sns.heatmap(df[num+['Exited']+num_cat].corr(),annot=True,cmap='Blues')
plt.title('Heatmap of numerical features')
plt.show()
```



0.8.1 No such notable correlations.

0.9 Chisquare test of Independence for all categorical columns.

0.9.1 Hypothesis Statements

0.9.2 Null Hypothesis (H0)

There is no significant relationship between the column and churn.

0.9.3 Alternate Hypothesis (H1)

There is significant relationship between the column and churn.

0.9.4 Significance level is set to .01

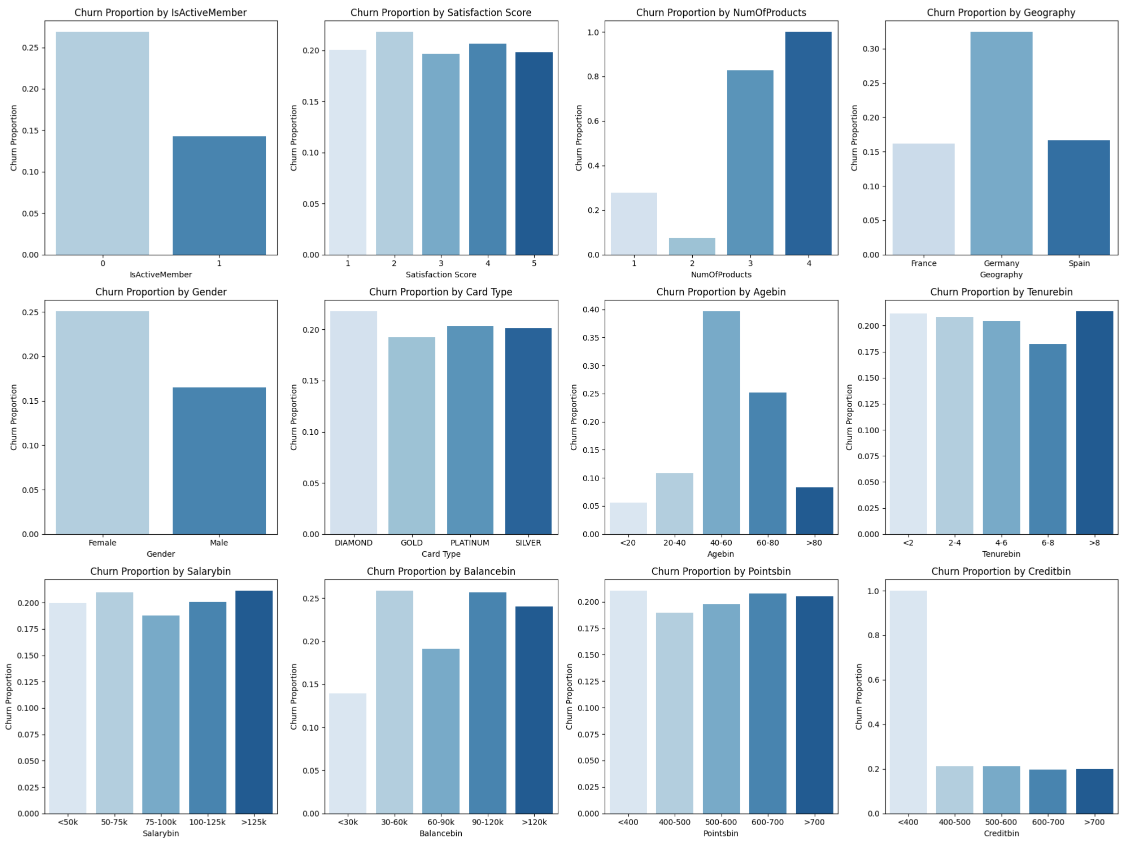
[13]:
```
significance_level=.01
for col in categorical_cols:
    contingency_table = pd.crosstab(df[col], df['Exited'])
```

```
    chi2, p, dof, expected = chi2_contingency(contingency_table)

    if p < significance_level:
        print(f'The relationship between {col.upper()} and EXITED is␣
↪statistically significant (p={p:.4f})')
    else:
        print(f'The relationship between {col.upper()} and EXITED is NOT␣
↪statistically significant (p={p:.4f})')
```

```
The relationship between HASCRCARD and EXITED is NOT statistically significant
(p=0.5026)
The relationship between ISACTIVEMEMBER and EXITED is statistically significant
(p=0.0000)
The relationship between COMPLAIN and EXITED is statistically significant
(p=0.0000)
The relationship between SATISFACTION SCORE and EXITED is NOT statistically
significant (p=0.4334)
The relationship between NUMOFPRODUCTS and EXITED is statistically significant
(p=0.0000)
The relationship between GEOGRAPHY and EXITED is statistically significant
(p=0.0000)
The relationship between GENDER and EXITED is statistically significant
(p=0.0000)
The relationship between CARD TYPE and EXITED is NOT statistically significant
(p=0.1679)
The relationship between AGEBIN and EXITED is statistically significant
(p=0.0000)
The relationship between TENUREBIN and EXITED is NOT statistically significant
(p=0.0924)
The relationship between SALARYBIN and EXITED is NOT statistically significant
(p=0.4108)
The relationship between BALANCEBIN and EXITED is statistically significant
(p=0.0000)
The relationship between POINTSBIN and EXITED is NOT statistically significant
(p=0.6197)
The relationship between CREDITBIN and EXITED is statistically significant
(p=0.0000)
```

[14]:
```
cols=['IsActiveMember']+num_cat+cat
churn_proportions = {}

for col in cols:
    churn_proportions[col] = df.groupby(col)['Exited'].mean()

fig, axes = plt.subplots(3, 4, figsize=(20, 15))
axes = axes.flatten()
```

```
for i, col in enumerate(cols):
    sns.barplot(x=churn_proportions[col].index, y=churn_proportions[col].
 ↪values, ax=axes[i],palette='Blues')
    axes[i].set_title(f'Churn Proportion by {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Churn Proportion')

plt.tight_layout()
plt.show()
```



[15]:
```
print(churn_proportions)
```

```
{'IsActiveMember': IsActiveMember
0    0.268715
1    0.142691
Name: Exited, dtype: float64, 'Satisfaction Score': Satisfaction Score
1    0.200311
2    0.217974
3    0.196376
4    0.206175
5    0.198104
```

```
Name: Exited, dtype: float64, 'NumOfProducts': NumOfProducts
1    0.277144
2    0.076035
3    0.827068
4    1.000000
Name: Exited, dtype: float64, 'Geography': Geography
France     0.161747
Germany    0.324432
Spain      0.166734
Name: Exited, dtype: float64, 'Gender': Gender
Female    0.250715
Male      0.164743
Name: Exited, dtype: float64, 'Card Type': Card Type
DIAMOND     0.217790
GOLD        0.192646
PLATINUM    0.203607
SILVER      0.201122
Name: Exited, dtype: float64, 'Agebin': Agebin
<20      0.056180
20-40    0.107741
40-60    0.396535
60-80    0.252212
>80      0.083333
Name: Exited, dtype: float64, 'Tenurebin': Tenurebin
<2     0.211538
2-4    0.208208
4-6    0.204649
6-8    0.182172
>8     0.213704
Name: Exited, dtype: float64, 'Salarybin': Salarybin
<50k        0.199348
50-75k      0.209614
75-100k     0.187697
100-125k    0.200627
>125k       0.211302
Name: Exited, dtype: float64, 'Balancebin': Balancebin
<30k       0.139708
30-60k     0.258741
60-90k     0.191566
90-120k    0.257104
>120k      0.240490
Name: Exited, dtype: float64, 'Pointsbin': Pointsbin
<400       0.210437
400-500    0.189669
500-600    0.197778
600-700    0.208006
>700       0.205026
Name: Exited, dtype: float64, 'Creditbin': Creditbin
```

```
<400       1.000000
400-500    0.213141
500-600    0.211721
600-700    0.197224
>700       0.198973
Name: Exited, dtype: float64}
```

0.10 2 sample Independent Ttest for all numerical columns to test whether churn and not churn means are significantly different.

0.10.1 Hypothesis Statements

0.10.2 Null Hypothesis (H0)

There is no significant difference between the mean of numerical column for churn and not churn.

0.10.3 Alternate Hypothesis (H1)

There is significant difference between the mean of numerical column for churn and not churn.

0.10.4 Significance level is set to .01

[16]:
```
results = {}
for col in numerical_cols:
    exited_group = df[df['Exited'] == 1][col]
    non_exited_group = df[df['Exited'] == 0][col]

    t_stat, p_value = ttest_ind(exited_group, non_exited_group)

    results[col] = {
        't-statistic': t_stat,
        'p-value': p_value,
        'significant': p_value < 0.01
    }

for col, result in results.items():
    print(f"Column: {col}")
    print(f"  t-statistic: {result['t-statistic']:.4f}")
    print(f"  p-value: {result['p-value']:.4f}")
    if result['significant']:
        print("  Result: Significant difference at the 0.01 level\n")
    else:
        print("  Result: NO significant difference at the 0.01 level\n")
```

```
Column: Balance
  t-statistic: 11.9407
  p-value: 0.0000
  Result: Significant difference at the 0.01 level

Column: EstimatedSalary
```

```
  t-statistic: 1.2489
  p-value: 0.2117
  Result: NO significant difference at the 0.01 level

Column: Point Earned
  t-statistic: -0.4628
  p-value: 0.6435
  Result: NO significant difference at the 0.01 level

Column: CreditScore
  t-statistic: -2.6778
  p-value: 0.0074
  Result: Significant difference at the 0.01 level

Column: Age
  t-statistic: 29.7638
  p-value: 0.0000
  Result: Significant difference at the 0.01 level

Column: Tenure
  t-statistic: -1.3656
  p-value: 0.1721
  Result: NO significant difference at the 0.01 level
```

0.10.5 Important columns from EDA:

IsActiveMember,Complain,NumOfProducts,Geography,Gender,Age,Balance,CreditScore.

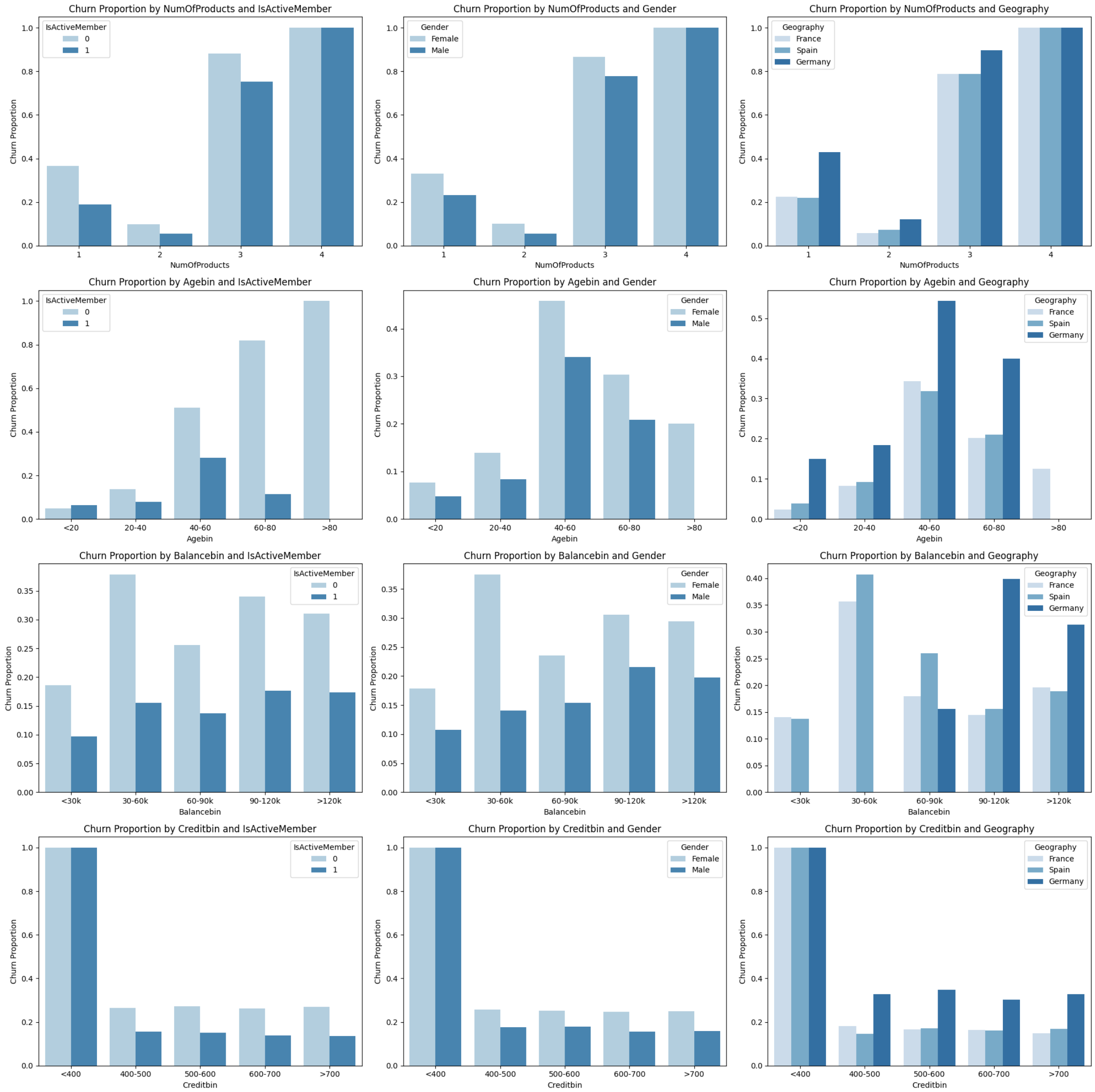
0.10.6 These features largely influence churn and can be used to predict and reduce churn.

0.10.7 Multivariate

0.10.8 We can see that if an user has complain he or she is very likely to churn so not including it further.

[17]:
```python
imp_x=['IsActiveMember','Gender','Geography'] #ignoring `Complain`
imp_y=['NumOfProducts','Agebin','Balancebin','Creditbin']
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 20))
axes = axes.flatten()
for i, col_y in enumerate(imp_y):
    for j, col_x in enumerate(imp_x):
        sns.barplot(x=col_y, y='Exited', hue=col_x, data=df, ax=axes[i*3 + j],␣
 ↪palette='Blues', ci=None)
        axes[i*3 + j].set_title(f'Churn Proportion by {col_y} and {col_x}')
        axes[i*3 + j].set_xlabel(col_y)
        axes[i*3 + j].set_ylabel('Churn Proportion')
plt.tight_layout()
```

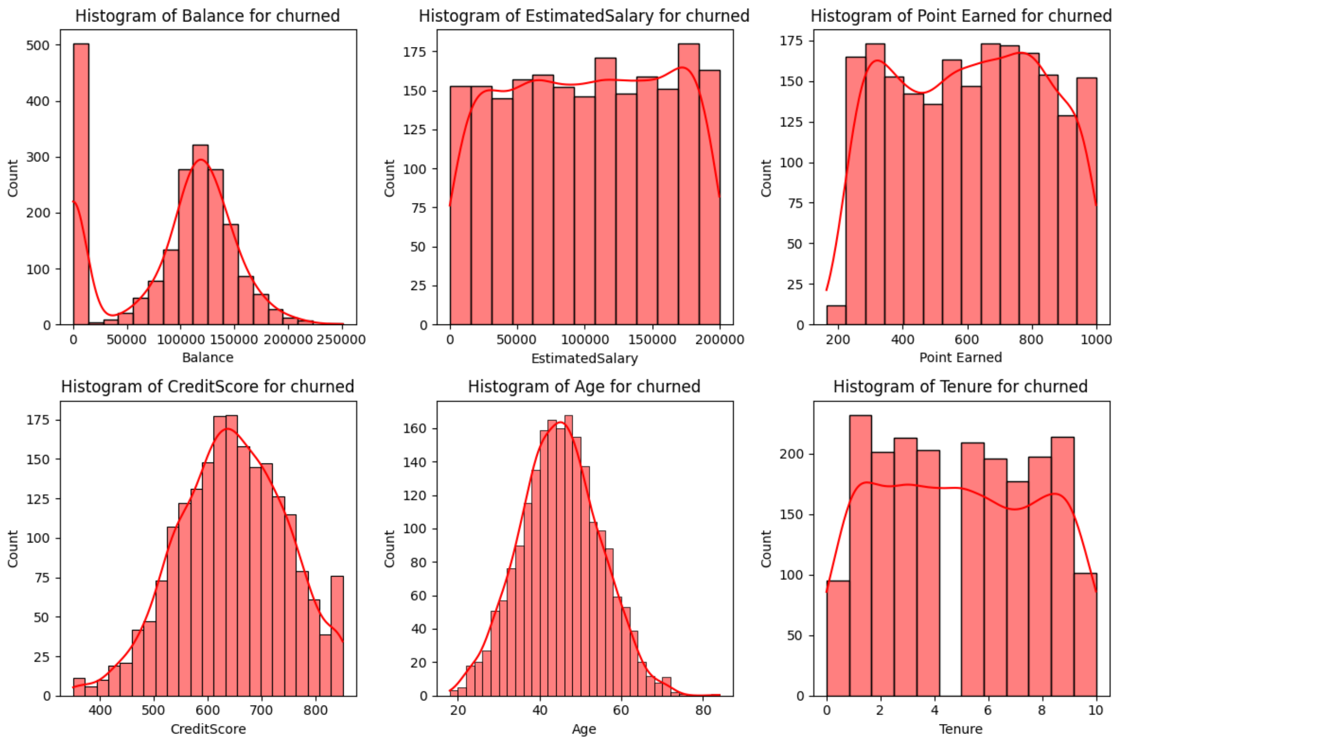
```
plt.show()
```



0.10.9 Profiling w.r.t Numerical columns

[18]:
```
fig, axes = plt.subplots(2, 3, figsize=(12, 8))
axes = axes.flatten()
for i, col in enumerate(numerical_cols):
    sns.histplot(data=df[df['Exited']==1], x=col, ax=axes[i],␣
  ↪color='Red',kde=True)
    axes[i].set_title(f'Histogram of {col} for churned')

plt.tight_layout()
```

```
plt.show()
```



[19]:
```
for col in numerical_cols:
    mean = df[df['Exited']==1][col].mean()
    median = df[df['Exited']==1][col].median()
    ci_lower, ci_upper = confidence_interval(df[df['Exited']==1][col])

    print(f"Column: {col}")
    print(f"Mean: {mean}")
    print(f"Median: {median}")
    print(f"95% Confidence Interval: [{ci_lower}, {ci_upper}]\n")
```

```
Column: Balance
Mean: 91109.47600588812
Median: 109344.23
95% Confidence Interval: [88574.82124842044, 93644.1307633558]

Column: EstimatedSalary
Mean: 101509.90878312069
Median: 102489.33499999999
95% Confidence Interval: [98993.23268806985, 104026.58487817152]

Column: Point Earned
Mean: 604.4484789008832
```

```
Median: 610.5
95% Confidence Interval: [594.6604596733546, 614.2364981284118]

Column: CreditScore
Mean: 645.4146221786065
Median: 646.0
95% Confidence Interval: [641.0558240080665, 649.7734203491465]

Column: Age
Mean: 44.83562315996075
Median: 45.0
95% Confidence Interval: [44.41164549191687, 45.25960082800462]

Column: Tenure
Mean: 4.934739941118744
Median: 5.0
95% Confidence Interval: [4.807162544405494, 5.062317337831995]
```

0.10.10 Customer Profile for Churned Users:

Financial Characteristics:

- **Balance**: Median balance of 109k with most concentration around 89k-94k.
- **Estimated Salary**: Average salary of 101k, typically ranging between 99k and 104k.

Credit and Loyalty Metrics:

- **Points Earned**: Average of 604 points, mostly between 595 and 614.
- **Credit Score**: Average score of 645, with a median of 646, generally between 641 and 649.

Demographics:

- **Age**: Age consistently around 45.
- **Tenure**: Average tenure of 5 years.

0.11 1)Comparative Analysis

0.11.1 i)Churn by Geography

Spain and France have similar lower churn rates whereas Germany has a significant higher churn rate.

0.11.2 ii)Gender Differences in Churn

We can conclude that female users are significantly more likely to churn.

0.11.3 iii)Age Difference in Churn

We can conclude that users in the age range of (40-80) is more likely to churn. Mean `Age` of churn and not churn are significantly different.

0.12 2)Behavioral Analysis

0.12.1 i)Product and Services Usage

Users with `numOfProducts` 3 and 4 are more likely to churn.

0.12.2 ii)Tenure Difference in Churn

`Tenure` does not significantly affect churn rate.

0.12.3 iii)Activity Level Analysis

Inactive users have significantly more chance of churning.

0.13 3)Financial Analysis

0.13.1 i)Balance vs. Churn

Although there is no clear trend between `Balance` and churn but balance groups (30k-60k) and (90k-120k) have higher probability of churn. Mean `Balance` of churn and not churn are significantly different.

0.13.2 ii)Credit Card Ownership

Owning a credit card does not significantly affect the churn rate.

0.13.3 iii)Credit Score vs. Churn

Users with credit score less than 400 have significantly high chance of churning out. Mean `CreditScore` of churn and not churn are significantly different.

0.14 4)Customer Satisfaction and Feedback

0.14.1 i)Complaint Analysis

If an user have a `Complain` there is significantly high chance that he or she might churn out.

0.14.2 ii)Satisfaction and Churn

There is no significant effect of `Satisfaction Score` on churn rate.

0.15 5)Card Usage Analysis

0.15.1 i)Impact of Card Type on Churn

There is no significant effect of `Card Type` on churn rate.

0.15.2 ii)Loyalty Points Analysis

There is no significant effect of `Points Earned` on churn rate.

0.16 6)Salary Analysis

0.16.1 i)Salary and Churn

There is no significant effect of `EstimatedSalary` on churn rate.

0.17 Random Forest Classification

[20]:
```
from sklearn.model_selection import train_test_split,RandomizedSearchCV
from sklearn.metrics import f1_score
import category_encoders as ce
from sklearn.ensemble import RandomForestClassifier
import pickle
```

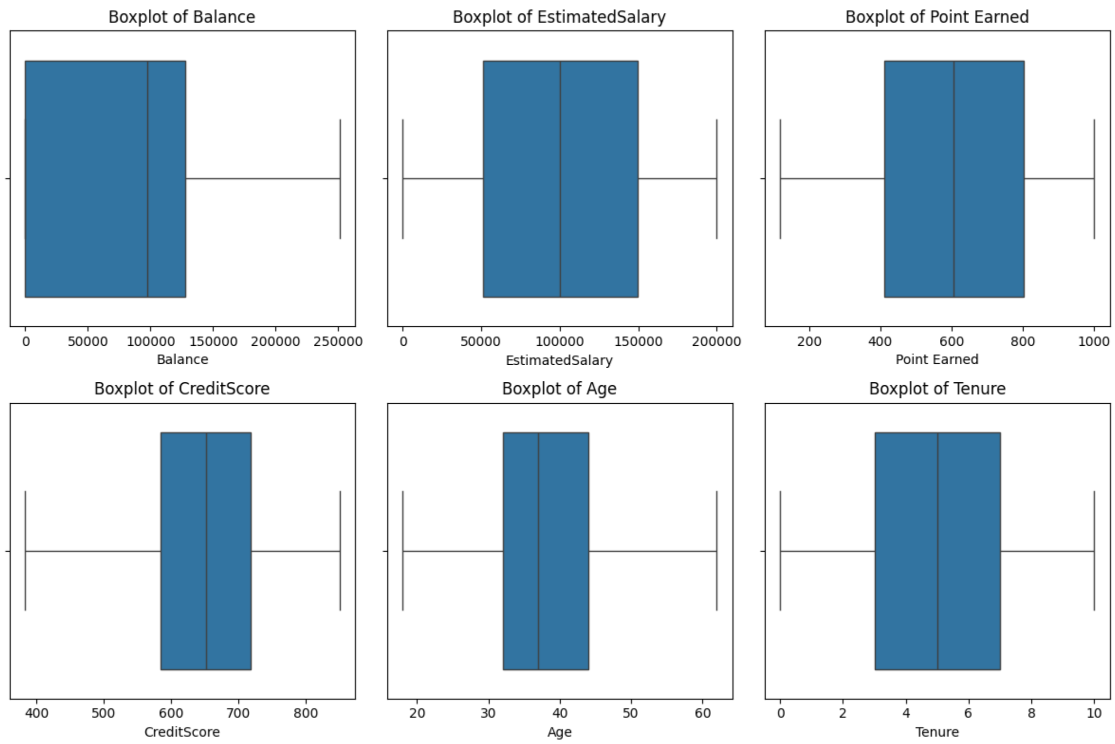
0.17.1 Preprocessing

[21]:
```
df = pd.read_csv('Bank-Records.csv',index_col=0).
 ↪drop(columns=['CustomerId','Surname'])
```

0.17.2 Clipping `Age` and `CreditScore`

[22]:
```
def clip_by_iqr(column):
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    return column.clip(lower=lower_bound, upper=upper_bound)

df['Age'] = clip_by_iqr(df['Age'])
df['CreditScore'] = clip_by_iqr(df['CreditScore'])
```

[23]:
```
fig, axes = plt.subplots(2, 3, figsize=(12, 8))
axes = axes.flatten()
for i, col in enumerate(numerical_cols):
    sns.boxplot(data=df, x=col, ax=axes[i])
    axes[i].set_title(f'Boxplot of {col}')

plt.tight_layout()
plt.show()
```



0.17.3 Encoding

Ordinal Encoding for `Card Type`

Target Encoding for `Geography` and `Gender`

[24]:
```
ordinal_encoder = ce.OrdinalEncoder(cols=['Card Type'], mapping=[{'col': 'Card␣
↪Type', 'mapping': {'DIAMOND': 1, 'GOLD': 2, 'PLATINUM': 3, 'SILVER':4}}])
df = ordinal_encoder.fit_transform(df)
target_encoder = ce.TargetEncoder(cols=['Geography', 'Gender'])
df= target_encoder.fit_transform(df, df['Exited'])
```

0.17.4 Splitting into Train and Test dataset

[25]:
```
X,y=df[['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure',␣
↪'Balance','NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary',␣
↪'Complain', 'Satisfaction Score', 'Card Type','Point Earned']],df[['Exited']]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.2,random_state=95)
```

0.17.5 Important features selection

[26]:
```
clf = RandomForestClassifier(random_state=95)
clf.fit(X_train, y_train)
feature_importances = clf.feature_importances_.round(4)
print('(FEATURE,IMPORTANCE_SCORE,IMPORTANT)')
list(zip(X_train.columns,feature_importances,feature_importances>.012))
```

```
(FEATURE,IMPORTANCE_SCORE,IMPORTANT)
```

[26]:
```
[('CreditScore', 0.0112, False),
 ('Geography', 0.0124, True),
 ('Gender', 0.0027, False),
 ('Age', 0.0721, True),
 ('Tenure', 0.0053, False),
 ('Balance', 0.0162, True),
 ('NumOfProducts', 0.0519, True),
 ('HasCrCard', 0.0014, False),
 ('IsActiveMember', 0.0159, True),
 ('EstimatedSalary', 0.0109, False),
 ('Complain', 0.783, True),
 ('Satisfaction Score', 0.0038, False),
 ('Card Type', 0.003, False),
 ('Point Earned', 0.0103, False)]
```

[27]:
```
X_train,X_test=X_train.loc[:,feature_importances>.012],X_test.loc[:
 ↪,feature_importances>.012]
X_train
```

[27]:

RowNumber	Geography	Age	Balance	NumOfProducts	IsActiveMember	Complain
9578	0.324432	52	112383.03	1	0	1
7664	0.161747	39	0.00	2	1	0
8294	0.324432	28	90696.78	1	1	0
1119	0.166734	61	91070.43	1	1	0
561	0.324432	29	105204.01	1	1	0
…	…	…	…	…	…	…
1795	0.161747	38	0.00	2	0	0
6918	0.324432	41	115897.73	1	0	0
6263	0.161747	44	63562.02	2	1	1
7466	0.324432	30	110153.27	1	1	0
7575	0.161747	33	74385.98	1	0	0

[8000 rows x 6 columns]

0.17.6 Important features predicting churn from Supervised Learning:

Geography,Age,Balance,NumOfProducts,IsActiveMember,Complain.

0.17.7 These features largely influence churn and can be used to predict and reduce churn.

0.17.8 Hyperparameter Tuning

[28]:
```
%%capture
'''
params = {'n_estimators': randint(100, 1000), 'max_depth': [None] +
 ↪list(range(10, 110, 10)), 'min_samples_split': randint(2, 20),
 ↪'min_samples_leaf': randint(1, 20), 'max_features': ['auto', 'sqrt', 'log2']
 ↪}
rf_classifier = RandomForestClassifier(random_state=95)
random_search = RandomizedSearchCV(estimator=rf_classifier,
 ↪param_distributions=params, n_iter=100, cv=5, scoring='accuracy',
 ↪random_state=95)
random_search.fit(X_train, y_train)
best_params = random_search.best_params_
best_estimator = random_search.best_estimator_
'''
```

[29]:
```
with open('random_forest_model.pkl', 'rb') as file:
    best_estimator = pickle.load(file)
    test_f1 = f1_score(y_test,best_estimator.predict(X_test))
    best_params={'max_depth': 60, 'max_features': 'sqrt', 'min_samples_leaf':
 ↪6, 'min_samples_split': 4, 'n_estimators': 520}
    print("Best Parameters:")
    print(best_params)
    print("Best Test F1 score:", test_f1)
```

```
Best Parameters:
{'max_depth': 60, 'max_features': 'sqrt', 'min_samples_leaf': 6,
'min_samples_split': 4, 'n_estimators': 520}
Best Test F1 score: 0.9963369963369962
```

0.17.9 Best Parameters and Test Accuracy

Best Parameters:

- **max_depth: 60**
  - The maximum depth of the tree. This controls the maximum number of levels in the tree.
- **max_features: 'sqrt'**
  - The number of features to consider when looking for the best split. Using the square root of the total number of features is a common heuristic that often provides a good balance between performance and computational efficiency.
- **min_samples_leaf: 6**
  - The minimum number of samples required to be at a leaf node.
- **min_samples_split: 4**
  - The minimum number of samples required to split an internal node.

- **n_estimators: 520**
  - The number of trees in the forest. More trees can lead to better performance, but with diminishing returns and increased computational cost.

Best Test F1 score: 0.9963

- This high F1 score indicates that the model performs extremely well on the test data. It suggests that the chosen hyperparameters have optimized the model's performance, balancing complexity and generalization effectively.

[30]:
```
%%capture
'''
print("Best Parameters:")
print(best_params)
test_f1 = f1_score(y_test,best_estimator.predict(X_test))
print("Best Test F1 score:", test_f1)
'''
```

0.17.10 Saving in pickle file

[31]:
```
%%capture
'''
with open('random_forest_model.pkl', 'wb') as file:
    pickle.dump(best_estimator, file)
'''
```

0.18 Recommendations

- Implement targeted retention strategies in regions with high churn rates, particularly in Germany.
- Develop personalized marketing campaigns to address the needs and preferences of female users.
- Offer incentives or rewards to encourage more active user engagement.
- Investigate the factors contributing to dissatisfaction among users aged around 45 and take proactive measures to address them.
- Prioritize customer support and complaint resolution to improve overall satisfaction and reduce churn proportion.
- Explore potential reasons behind the higher churn rates among users with specific balance ranges and lower credit scores, and tailor retention efforts accordingly.
- Investigate why higher number of products result in more churn proportion.
- Use the profile of churning users [45 years with 89k-94k balance and credit score around 641-649] to figure out retention strategies.
- Use the important features and model to predict churn as well as form strategies to prevent it.
- Continuously monitor user feedback and satisfaction metrics to identify emerging trends and areas for improvement.

[ ]:
```

```