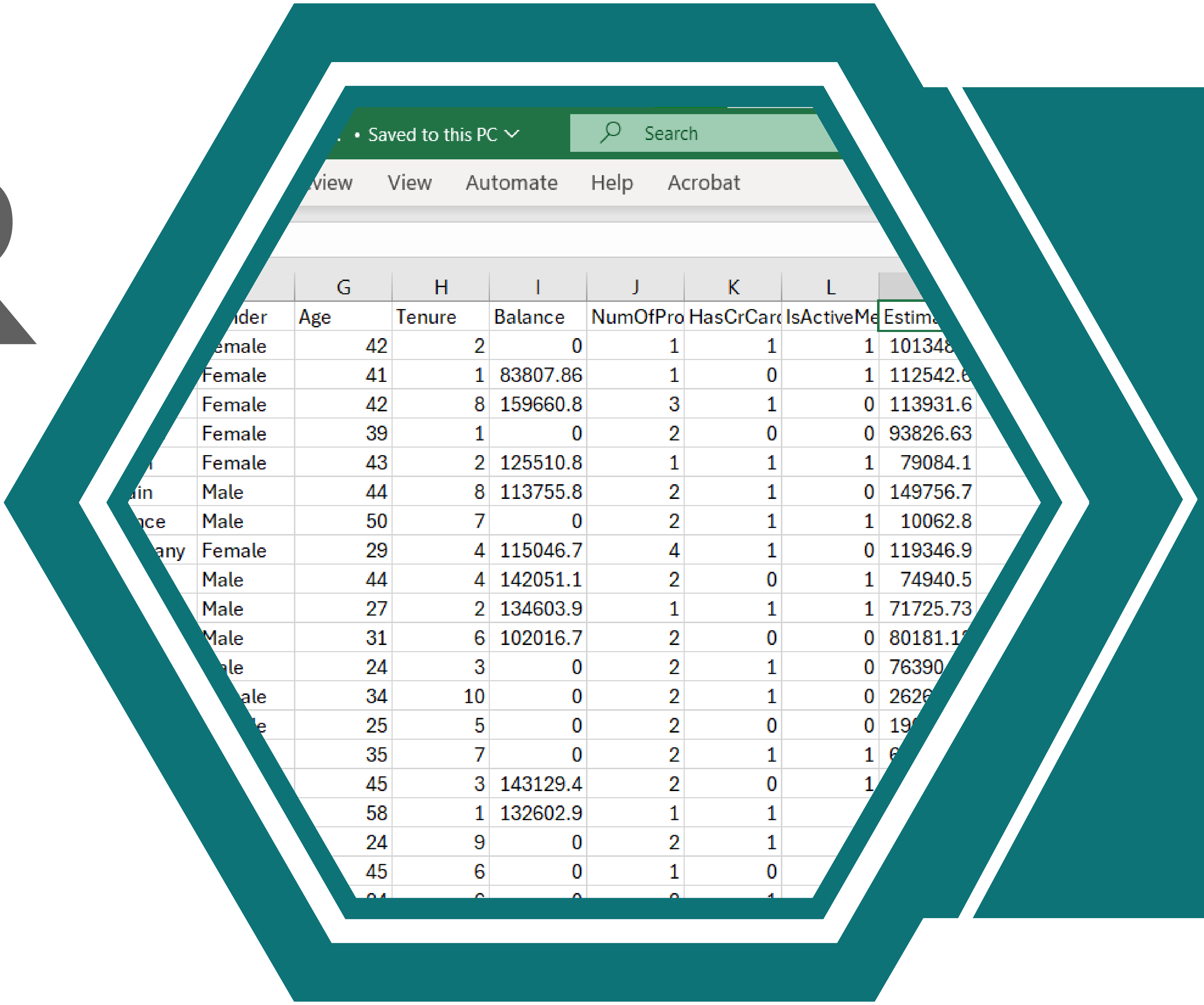
FINAL PRESENTATION

CUSTOMER CHURN



A DATA ANALYSIS PROJECT

BY: AHMAD SOHAIL | 64011748

THYNK UNLIMITED

Table Of Content

THYNK UNLIMITED

Introduction

- Banks collect vast amounts of data to enhance their services and meet business objectives.
- A critical aspect of this data analysis is understanding customer behavior, such as churn rates.
- For this Data Analysis Project, I was assigned to work on a model to predict Customer Churn based on a dataset from Kaggle.
- This analysis helps the bank identify at-risk customers and take proactive measures to retain them.

THYNK UNLIMITED

About Dataset

Dataset by *RADHESHYAM KOLLIPARA* that involves Customer data collected by a bank.

1). 10,000 records to analyze

- 2). 10+ necessary features to predict from
- 3). Includes non-numeric and numeric values.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	RowNumb	CustomerI	Surname	CreditScor	Geography	Gender	Age	Tenure	Balance	NumOfPro	HasCrCard	IsActiveM	Estimated	Exited	Complain	Satisfactio	Card Type	Point Earned	
2	1	1.6E+07	Hargrave	619	France	Female	42	2	0	1	1	1	101349	1	1	2	DIAMOND	464	
3	2	1.6E+07	Hill	608	Spain	Female	41	1	83807.9	1	0	1	112543	0	1	3	DIAMOND	456	
4	3	1.6E+07	Onio	502	France	Female	42	8	159661	3	1	0	113932	1	1	3	DIAMOND	377	
5	4	1.6E+07	Boni	699	France	Female	39	1	0	2	0	0	93826.6	0	0	5	GOLD	350	
6	5	1.6E+07	Mitchell	850	Spain	Female	43	2	125511	1	1	1	79084.1	0	0	5	GOLD	425	
7	6	1.6E+07	Chu	645	Spain	Male	44	8	113756	2	1	0	149757	1	1	5	DIAMOND	484	
8	7	1.6E+07	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0	0	2	SILVER	206	
9	8	1.6E+07	Obinna	376	Germany	Female	29	4	115047	4	1	0	119347	1	1	2	DIAMOND	282	
10	9	1.6E+07	He	501	France	Male	44	4	142051	2	0	1	74940.5	0	0	3	GOLD	251	
11	10	1.6E+07	H?	684	France	Male	27	2	134604	1	1	1	71725.7	0	0	3	GOLD	342	
12	11	1.6E+07	Bearce	528	France	Male	31	6	102017	2	0	0	80181.1	0	0	3	GOLD	264	
13	12	1.6E+07	Andrews	497	Spain	Male	24	3	0	2	1	0	76390	0	0	3	GOLD	249	
14	13	1.6E+07	Kay	476	France	Female	34	10	0	2	1	0	26261	0	0	3	SILVER	119	
15	14	1.6E+07	Chin	549	France	Female	25	5	0	2	0	0	190858	0	0	3	PLATINUM	549	
16	15	1.6E+07	Scott	635	Spain	Female	35	7	0	2	1	1	65951.7	0	0	2	GOLD	318	
17	16	1.6E+07	Goforth	616	Germany	Male	45	3	143129	2	0	1	64327.3	0	0	5	GOLD	308	
18	17	1.6E+07	Romeo	653	Germany	Male	58	1	132603	1	1	0	5097.67	1	0	2	SILVER	163	
19	18	1.6E+07	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.4	0	0	3	SILVER	544	
20	19	1.6E+07	Muldrow	587	Spain	Male	45	6	0	1	0	0	158685	0	0	3	PLATINUM	732	
21	20	1.6E+07	Hao	726	France	Female	24	6	0	2	1	1	54724	0	0	4	GOLD	477	
22	21	1.6E+07	McDonald	732	France	Male	41	8	0	2	1	1	170886	0	0	3	PLATINUM	568	
23	22	1.6E+07	Dellucci	636	Spain	Female	32	8	0	2	1	0	138555	0	0	2	DIAMOND	336	
24	23	1.6E+07	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118914	1	1	2	DIAMOND	887	
25	24	1.6E+07	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0	0	2	SILVER	665	
26	25	1.6E+07	Yen	846	France	Female	38	5	0	1	1	1	187616	0	0	3	PLATINUM	225	
27	26	1.6E+07	Maclean	577	France	Male	25	3	0	2	0	1	124508	0	0	3	PLATINUM	659	
28	27	1.6E+07	Young	756	Germany	Male	36	2	136816	1	1	1	170042	0	0	5	DIAMOND	236	
29	28	1.6E+07	Nebechi	571	France	Male	44	9	0	2	0	0	38433.4	0	0	4	PLATINUM	448	
30	29	1.6E+07	McWilliam	574	Germany	Female	43	3	141349	1	1	1	100187	0	0	5	DIAMOND	499	
31	30	1.6E+07	Lucciano	411	France	Male	29	0	59697.2	2	1	1	53483.2	0	0	2	GOLD	343	
32	31	1.6E+07	Azikiwe	591	Spain	Female	39	3	0	3	1	0	140469	1	1	3	DIAMOND	298	
33	32	1.6E+07	Odinakach	533	France	Male	36	7	85311.7	1	0	1	156732	0	0	5	SILVER	628	
34	33	1.6E+07	Sanderson	553	Germany	Male	41	9	110113	2	0	0	81898.8	0	0	3	GOLD	611	

THYNK UNLIMITED

Recap

- Developed an initial predictive model to analyze customer churn.
- Conducted basic Exploratory Data Analysis (EDA).
- Executed data preprocessing to prepare the dataset.

New EDA

Extensive research

Extended the Textual EDA.

More visualizations

Extended the usage of plots that are helpful.

Key Observation(s)

101010
101010
101010

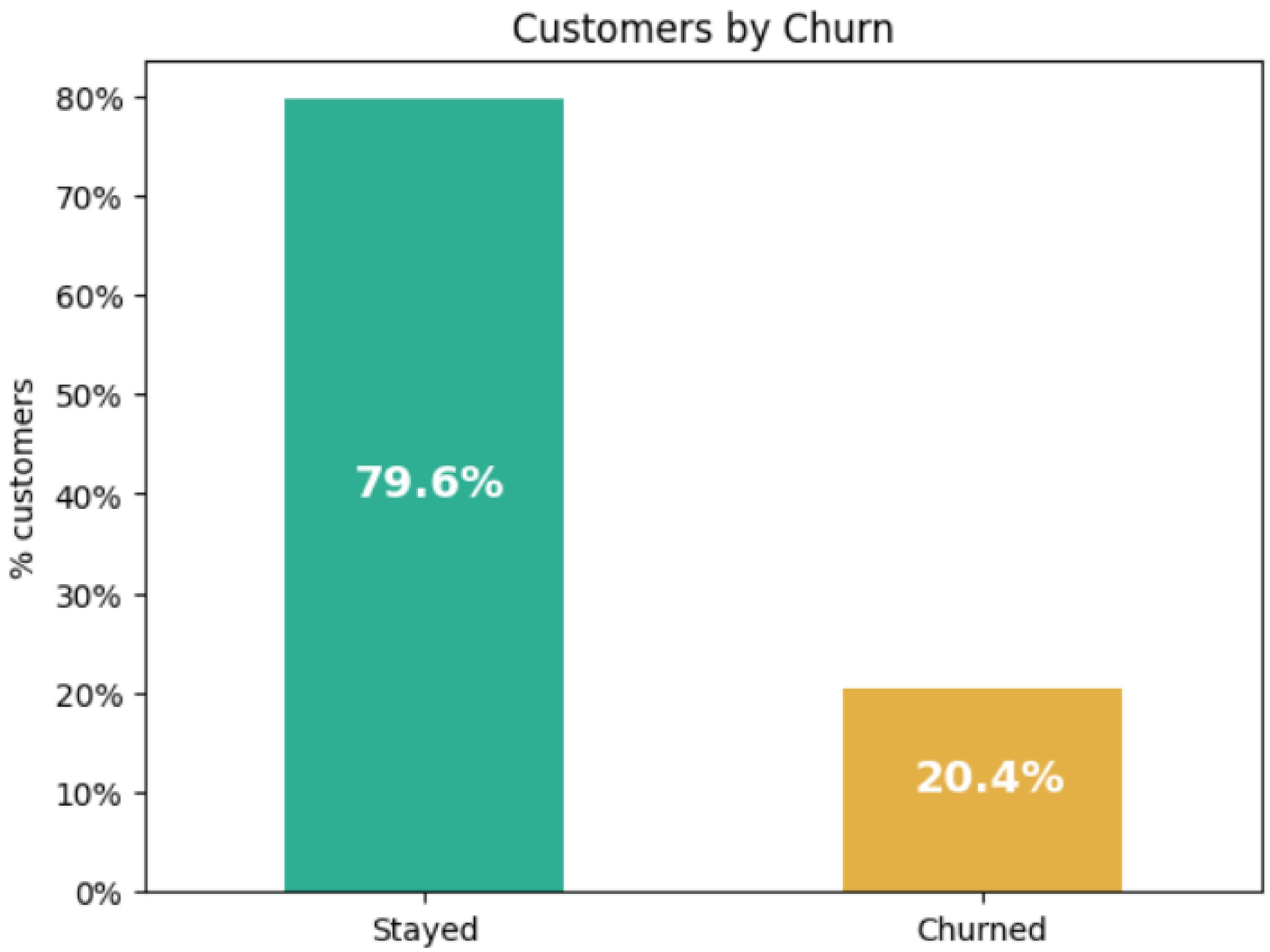
Found Dataset's key error prone areas.

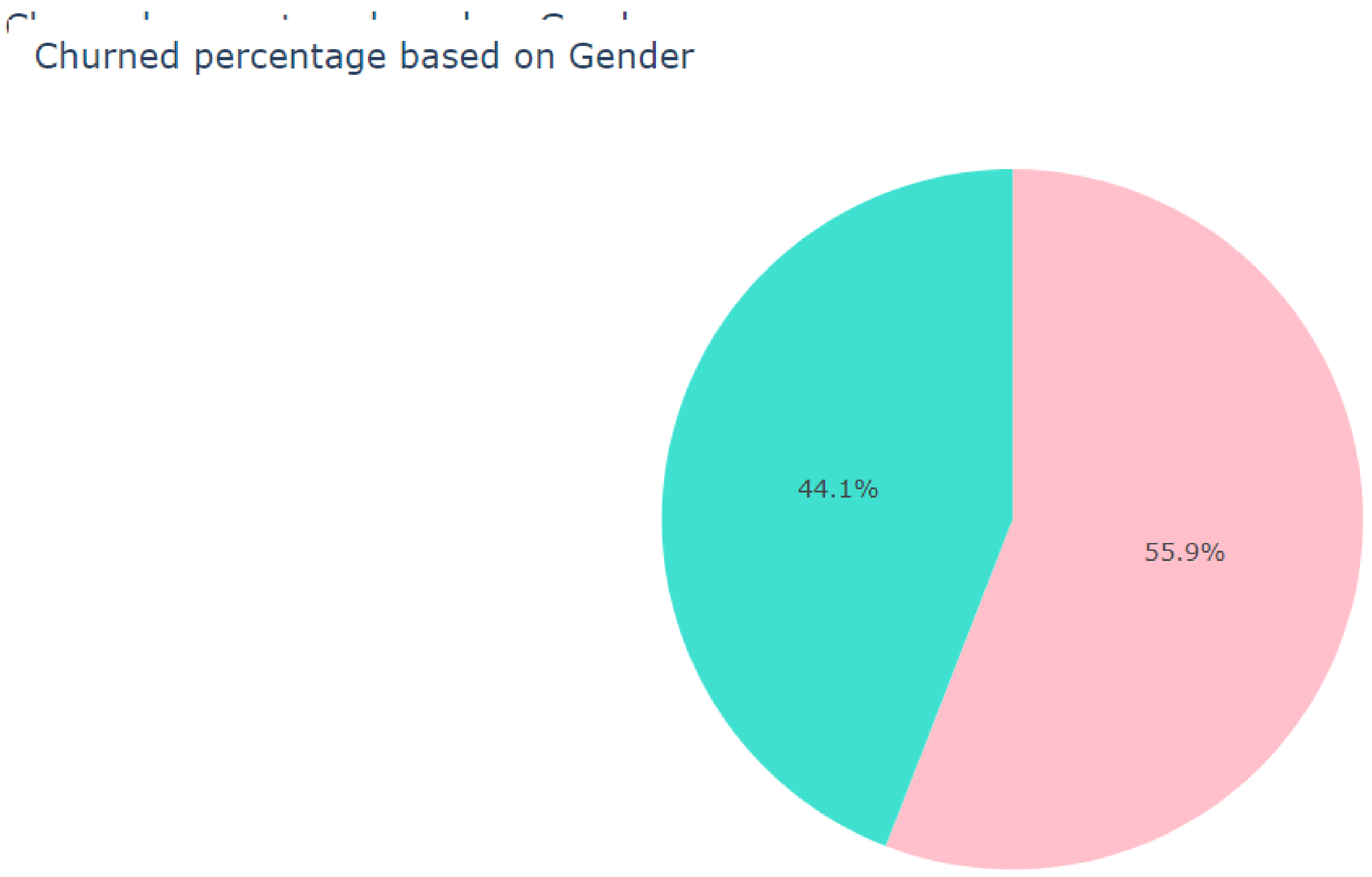
```
df.info()
```

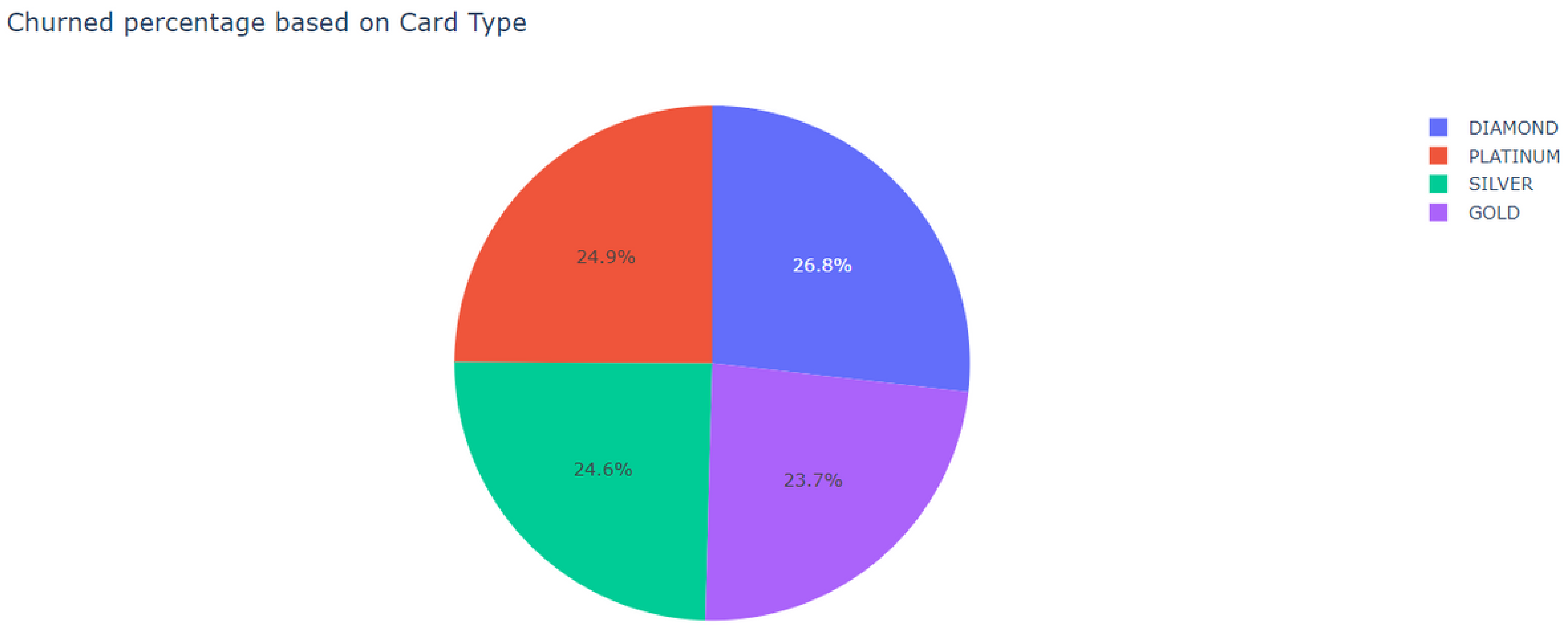
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   RowNumber           10000 non-null  int64  
 1   CustomerId          10000 non-null  int64  
 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  
 14  Complain            10000 non-null  int64  
 15  Satisfaction Score  10000 non-null  int64  
 16  Card Type           10000 non-null  object 
 17  Point Earned        10000 non-null  int64  
dtypes: float64(2), int64(12), object(4)
memory usage: 1.4+ MB
```

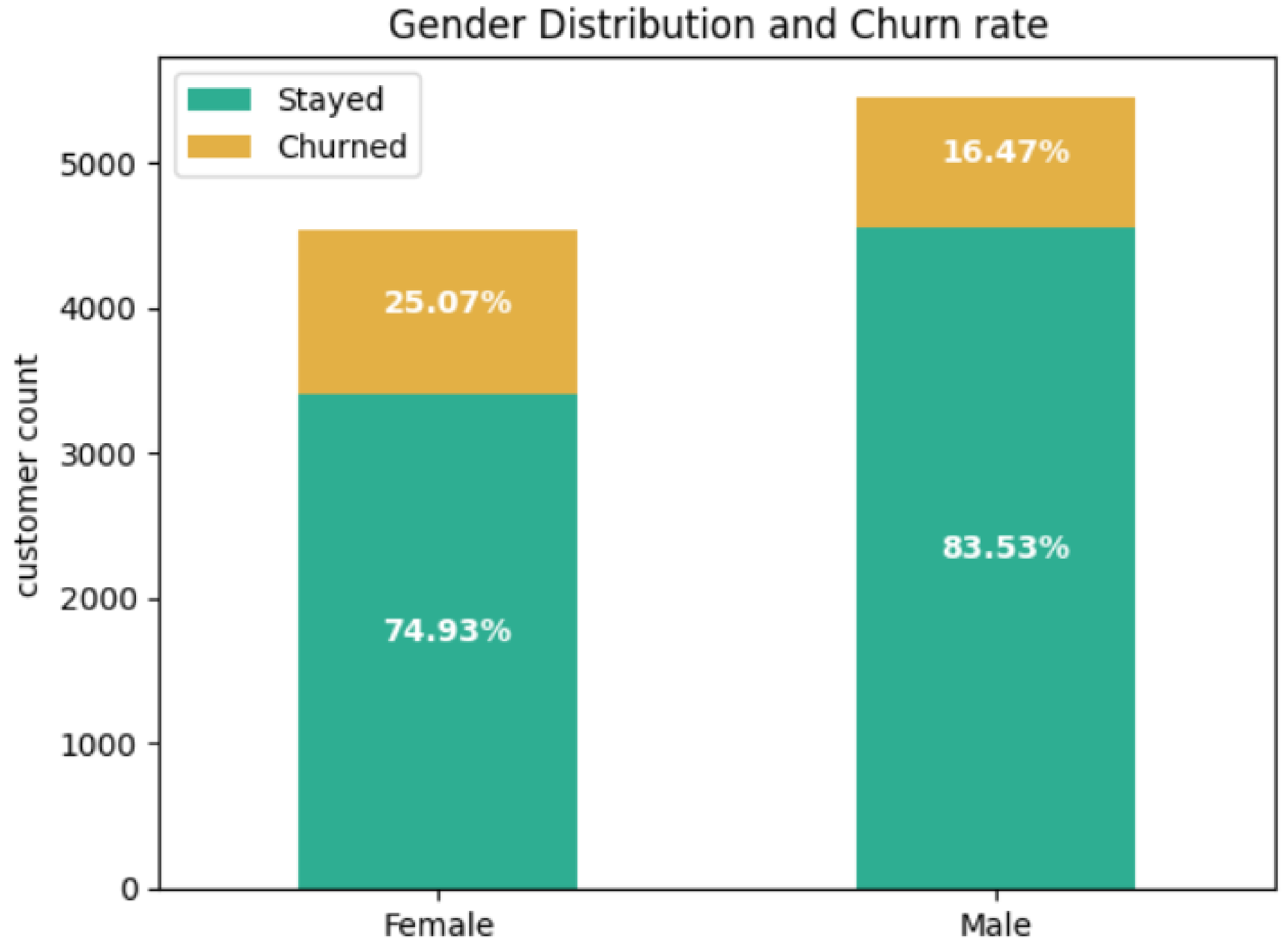
```
df.isnull().sum()
```

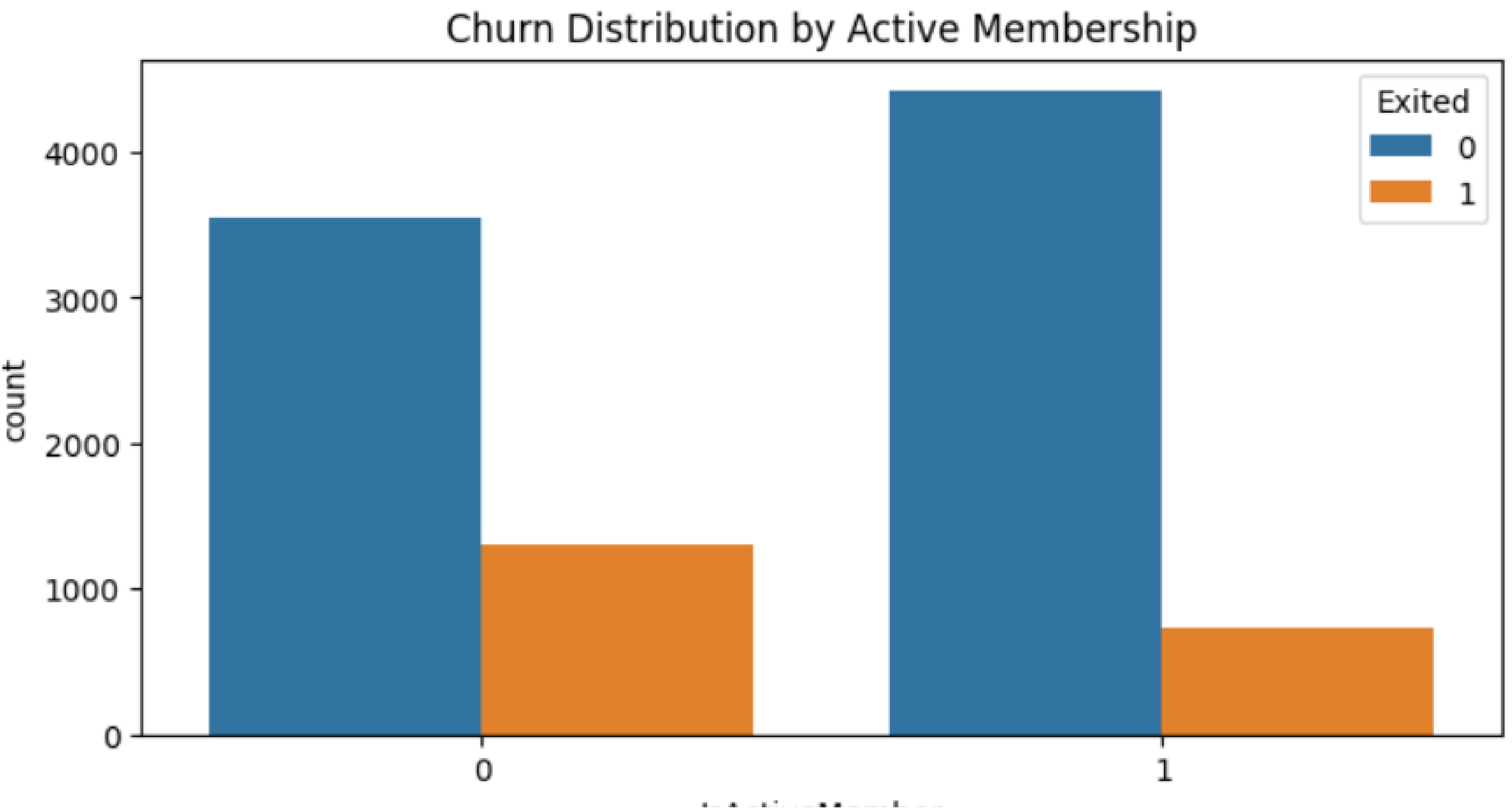
```
RowNumber             0
CustomerId            0
Surname               0
CreditScore           0
Geography             0
Gender                0
Age                   0
Tenure                0
Balance               0
NumOfProducts         0
HasCrCard             0
IsActiveMember        0
EstimatedSalary       0
Exited                0
Complain              0
Satisfaction Score    0
Card Type             0
Point Earned          0
dtype: int64
```

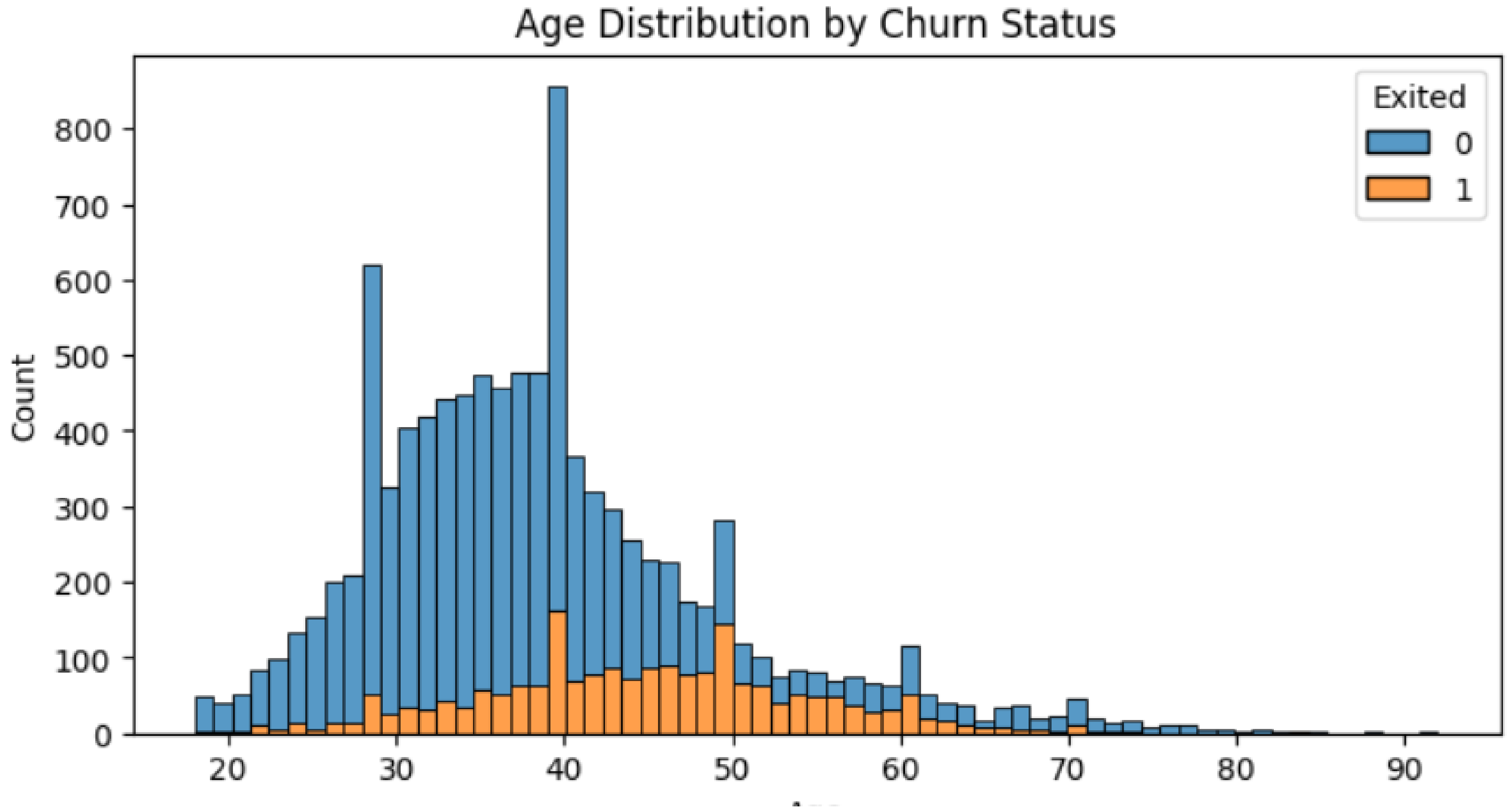

Customers by Churn
% customers
80%
70%
60%
50%
40%
30%
20%
10%
0%
79.6%
20.4%
Stayed
Churned

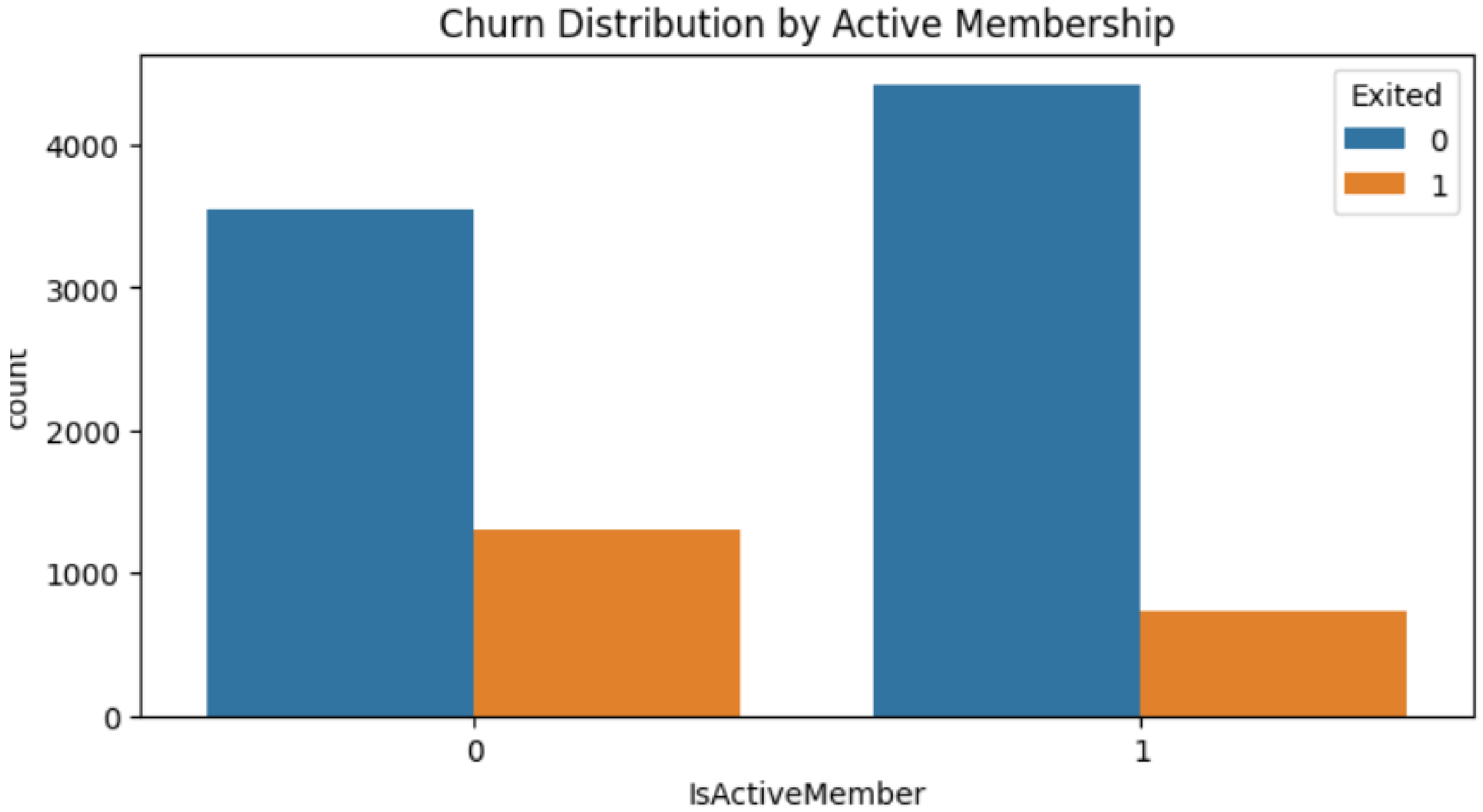

Churned percentage based on Gender
44.1%
55.9%

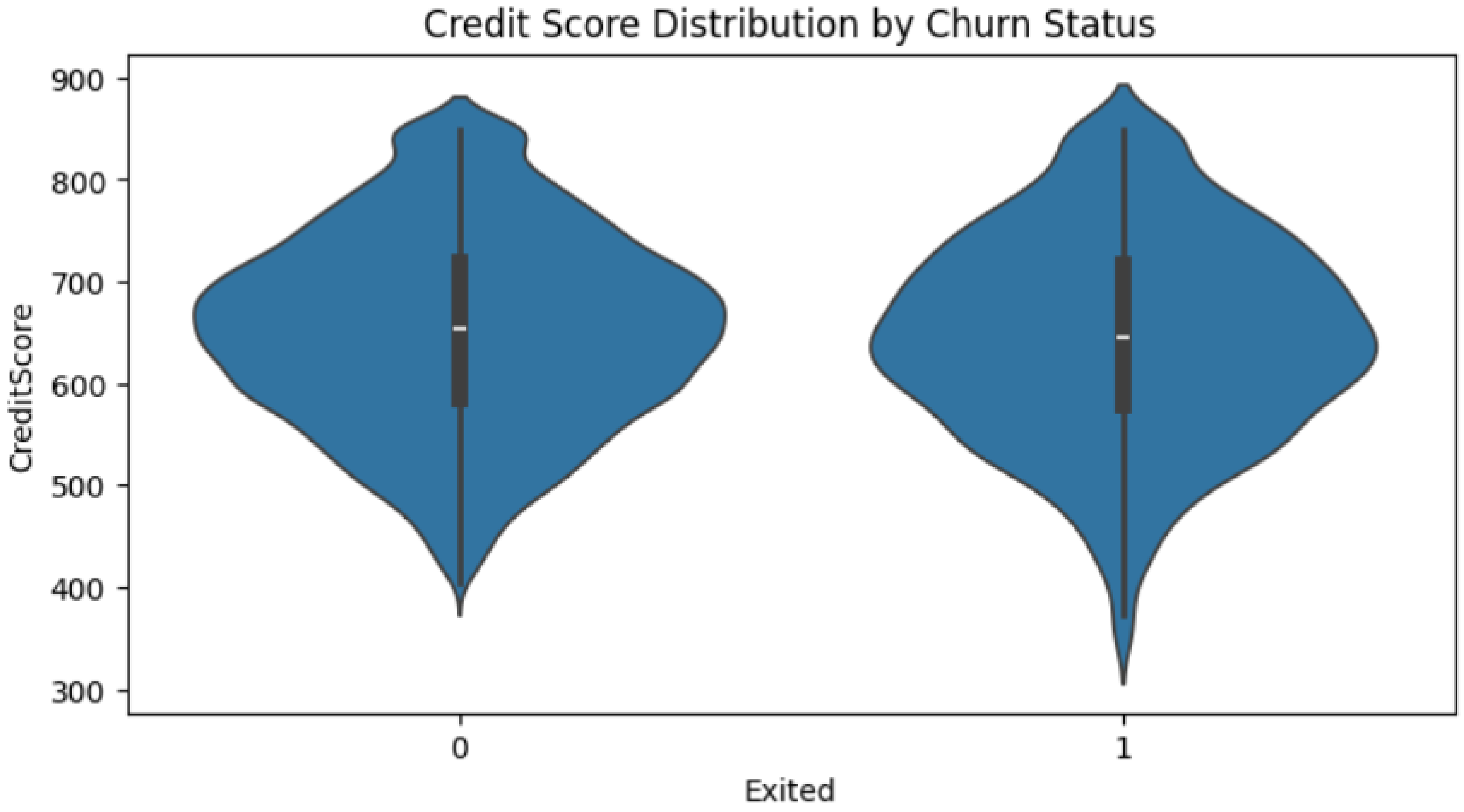

Churned percentage based on Card Type
24.9%
26.8%
24.6%
23.7%
DIAMOND
PLATINUM
SILVER
GOLD

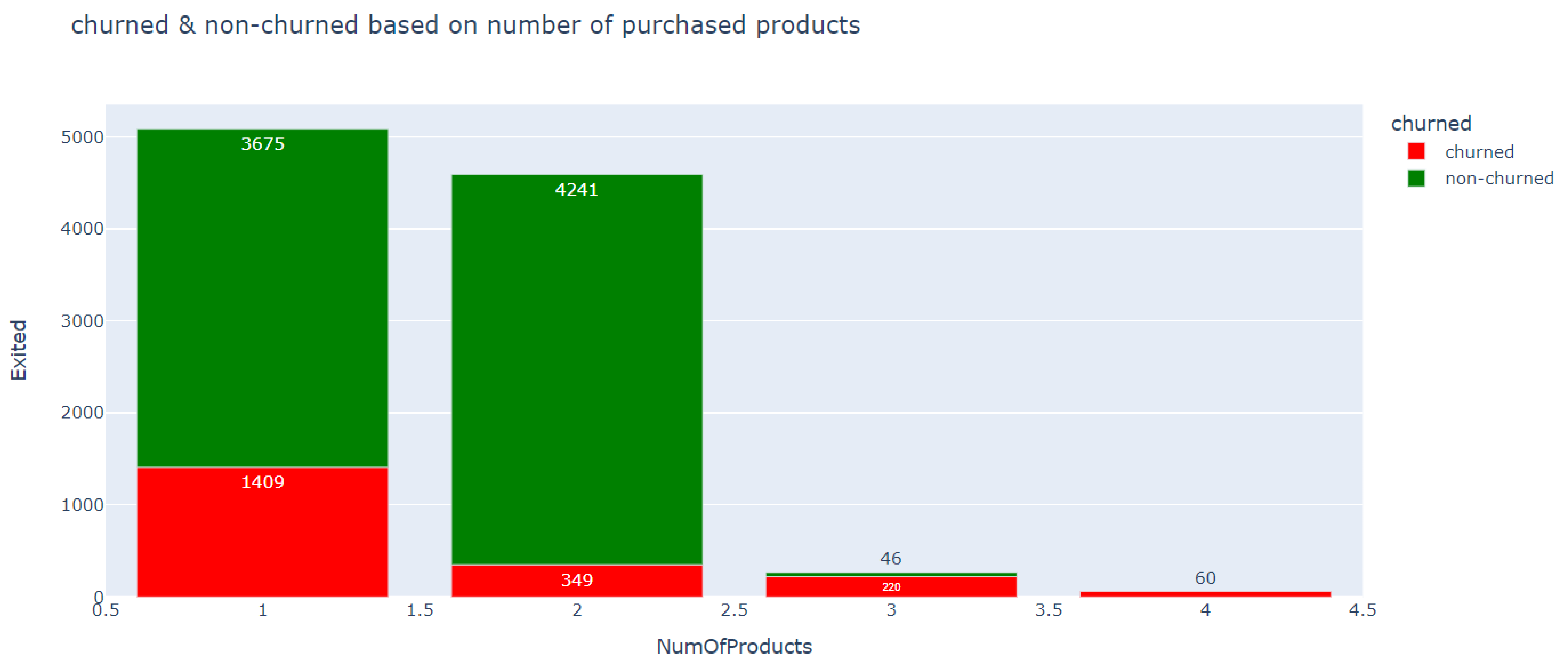

Gender Distribution and Churn rate
Stayed
Churned
customer count
5000
4000
3000
2000
1000
0
25.07%
74.93%
16.47%
83.53%
Female
Male


Churn Distribution by Active Membership
count
0
1000
2000
3000
4000
0
1
Exited
0
1

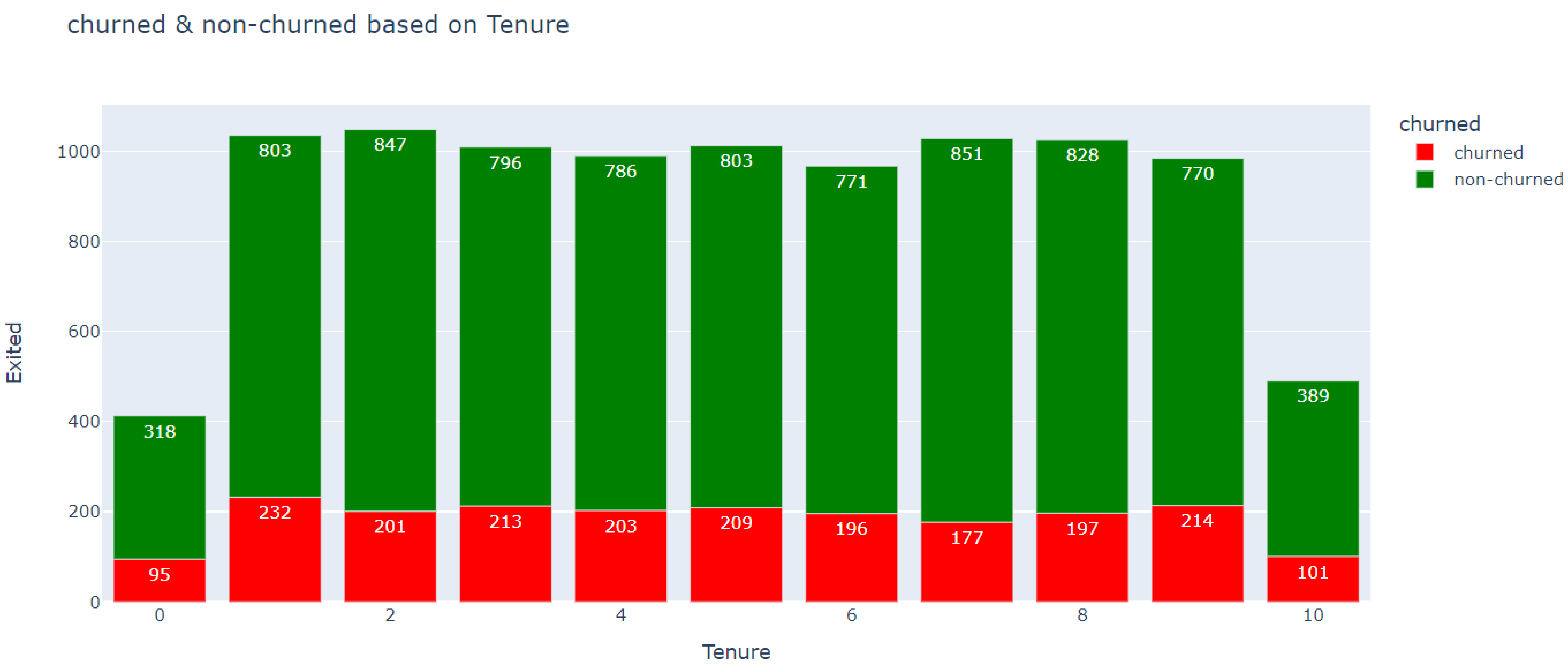

Age Distribution by Churn Status
Count
0
100
200
300
400
500
600
700
800
20
30
40
50
60
70
80
90
Exited
0
1

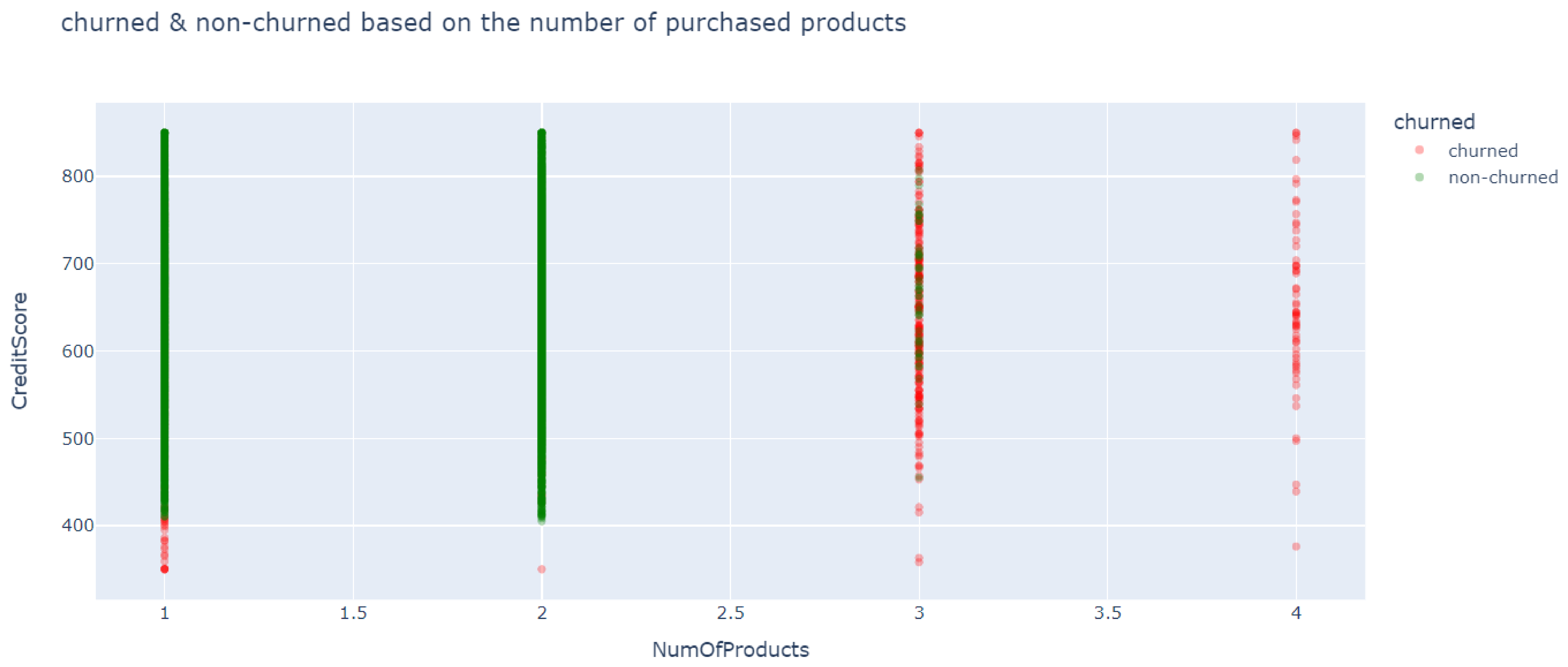

Churn Distribution by Active Membership
count
0
1000
2000
3000
4000
0
1
IsActiveMember
Exited
0
1


Credit Score Distribution by Churn Status
CreditScore
300
400
500
600
700
800
900
0
1
Exited



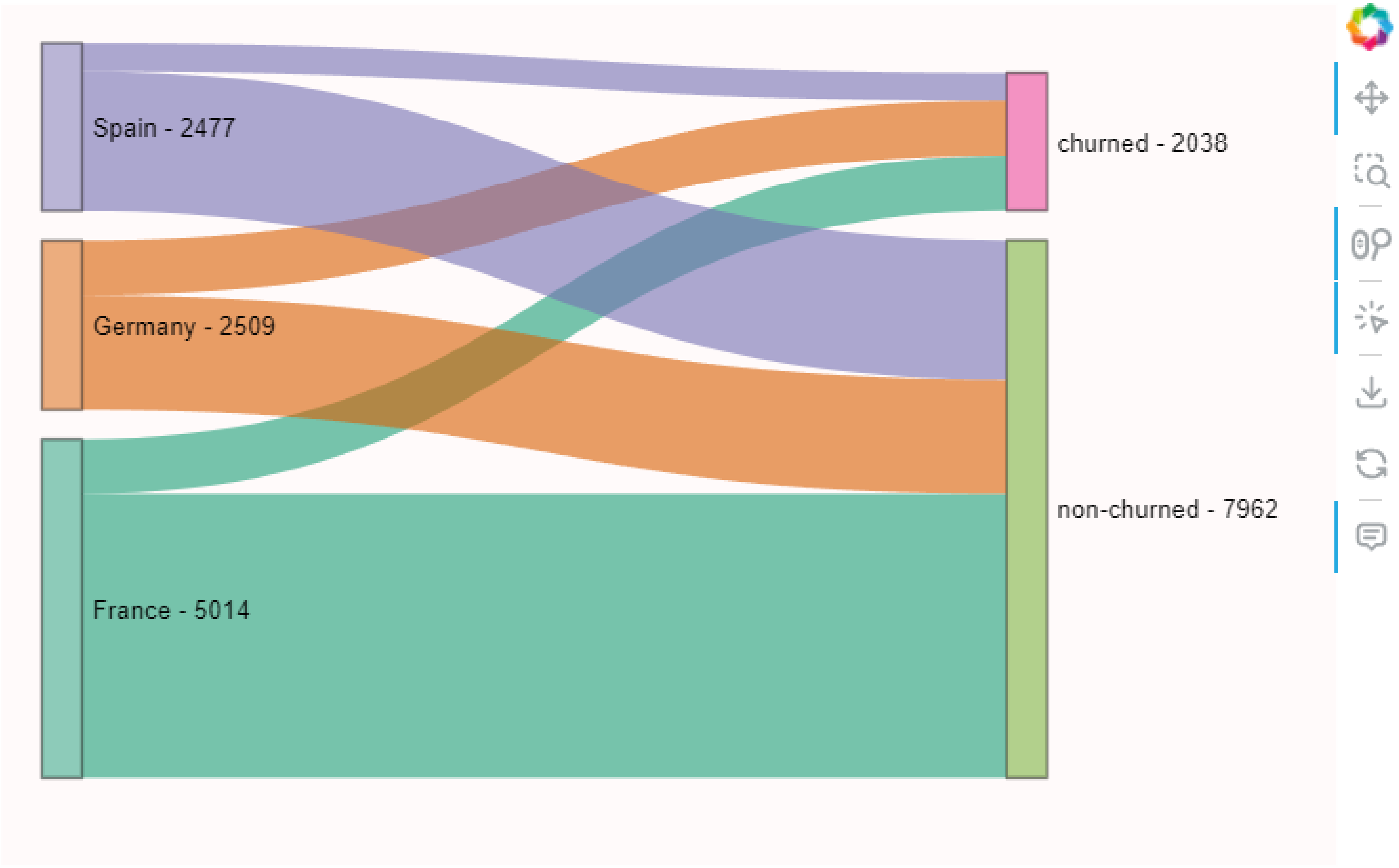
churned & non-churned based on number of purchased products
churned
churned
non-churned
Exited
NumOfProducts
0
1000
2000
3000
4000
5000
0.5
1
1.5
2
2.5
3
3.5
4
4.5
3675
1409
4241
349
46
220
60

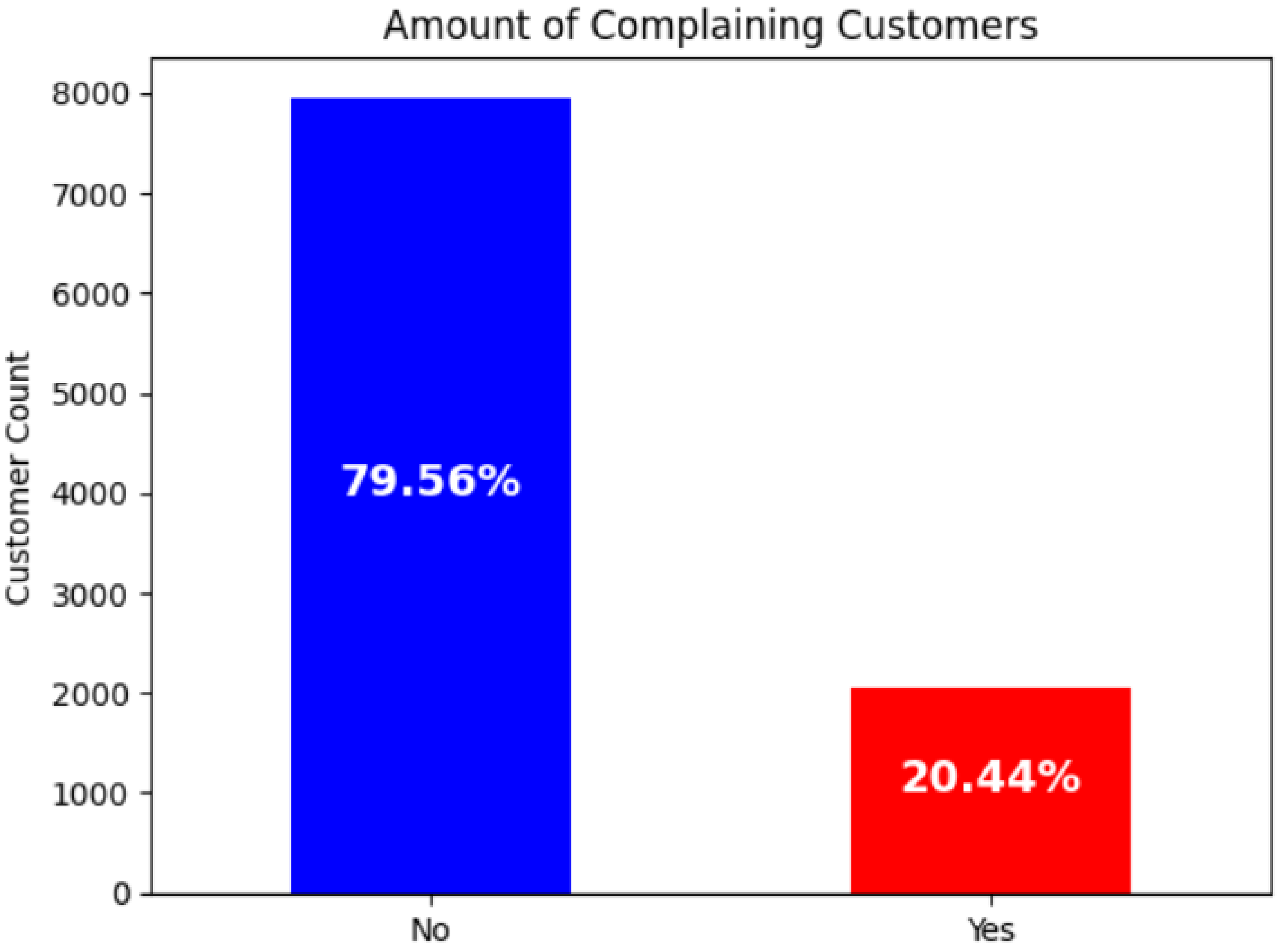

churned & non-churned based on Tenure
Exited
Tenure
churned
churned
non-churned
0
200
400
600
800
1000
0
2
4
6
8
10
318
95
803
232
847
201
796
213
786
203
803
209
771
196
851
177
828
197
770
214
389
101


churned & non-churned based on the number of purchased products
churned
churned
non-churned
CreditScore
400
500
600
700
800
NumOfProducts
1
1.5
2
2.5
3
3.5
4

uget` is deprecated and will be removed in version 1.4, use `transform_reference` instead.

Flow of churned and non-churned clients based on country



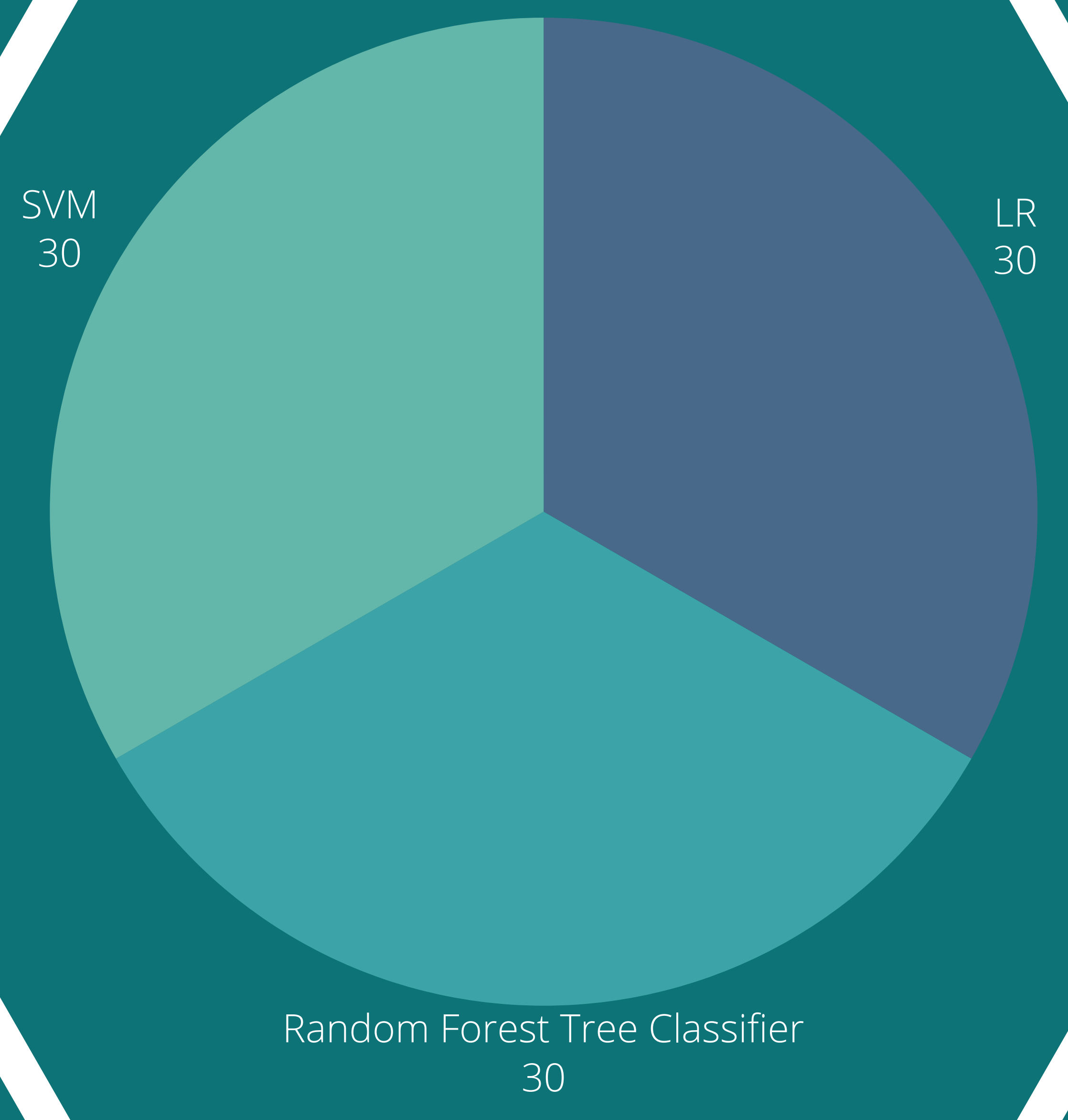


Exited	0	1
Complain		
0	7952	4
1	10	2034

THYNK UNLIMITED

Model (s)

I explored three algorithms for this dataset: logistic regression, Support Vector Machine (SVM), and Random Forest Classifier. Each model underwent meticulous hyperparameter tuning to enhance accuracy and ensure and other evaluation metrics. Addressed potential overfitting and bias to uphold the integrity of the predictions.



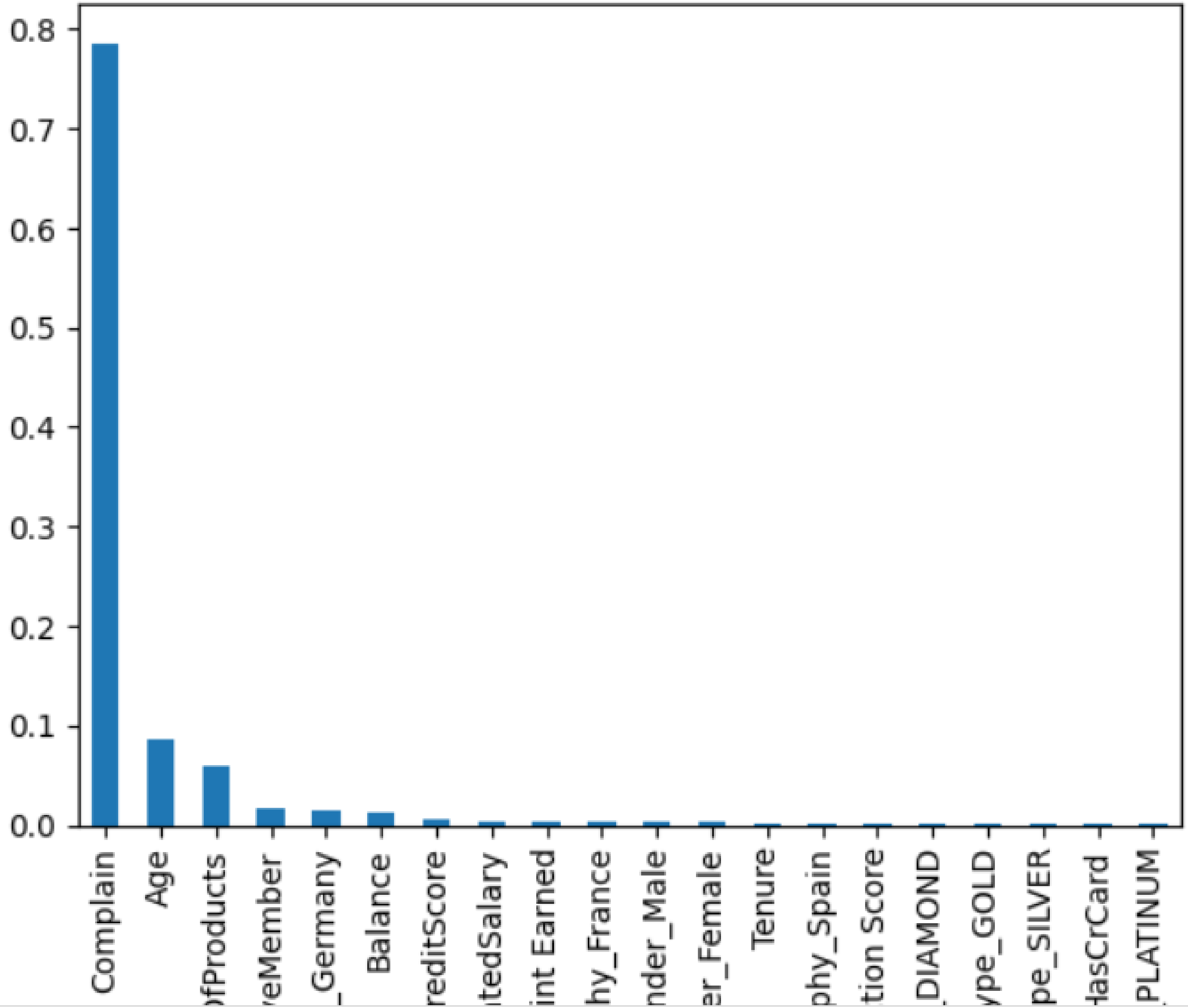
First Model: Random Forest Tree Classifier

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

#one hot encoding
bank_dummies = pd.get_dummies(df)

y = bank_dummies['Exited'].values
X = bank_dummies.drop(columns = ['Exited'])

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)

model_rf = RandomForestClassifier(n_estimators=1000, oob_score= True, random_state=77, max_leaf_nodes=30)

model_rf.fit(X_train, y_train)

preds = model_rf.predict(X_test)

metrics.accuracy_score(y_test, preds)
```

```
0.9985
```

```
#Finding important features and plotting them based on their affect on the model
forest_importances = pd.Series(model_rf.feature_importances_, index=X_train.columns).sort_values(ascending=False)
forest_importances.plot(kind='bar')

#Complain causing bias for our dataset
```

<Axes: >



THYNK UNLIMITED

Finetuning

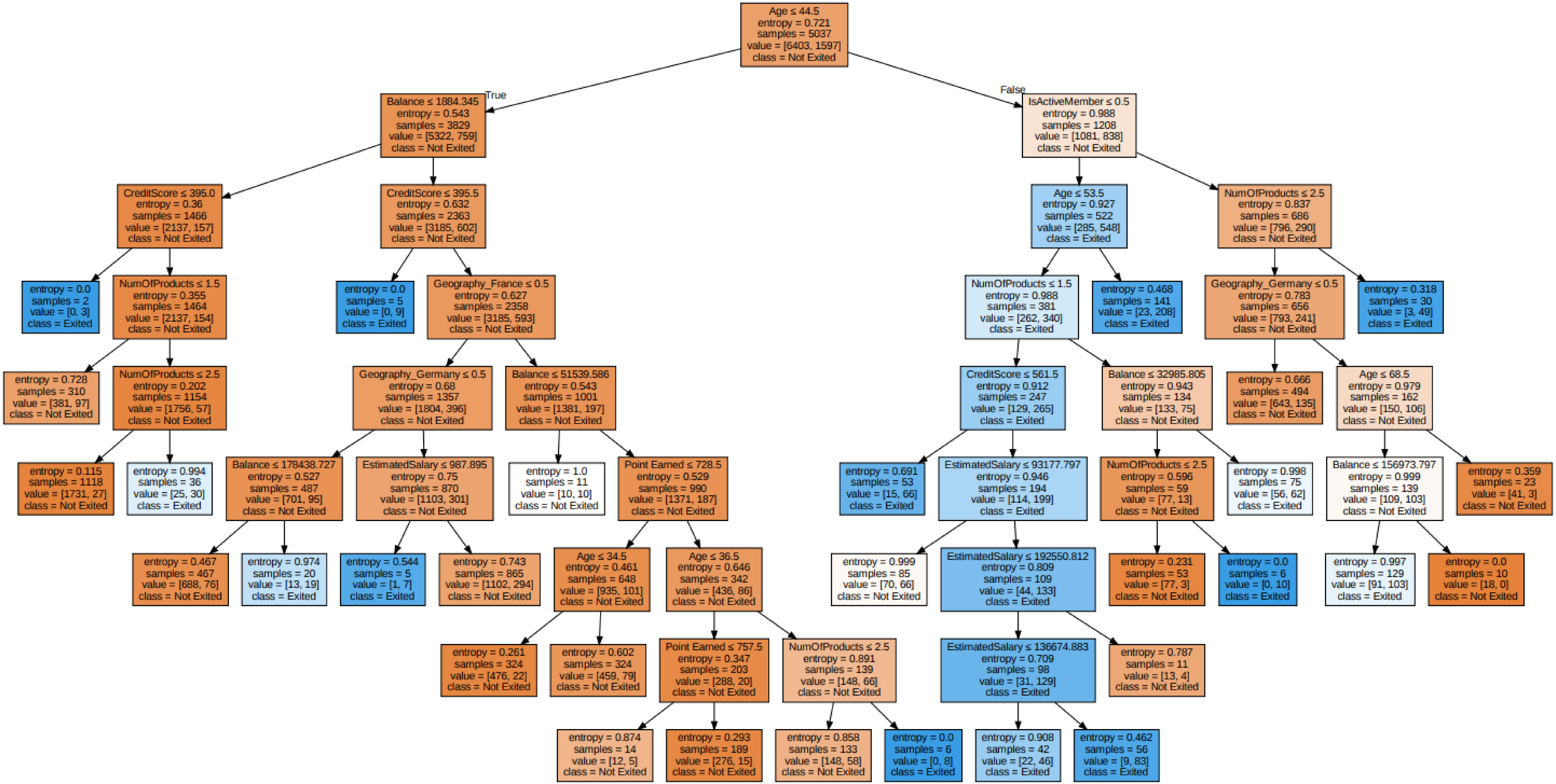
- Varied hyperparameters that will be showed in demo for higher evaluation score.
- Used different techniques for overfitting.

```python
model_rf.fit(X_train, y_train)

preds = model_rf.predict(X_test)

metrics.accuracy_score(y_test, preds)
```

```
0.84575
```

```python
# Calculating the accuracy
accuracy = accuracy_score(y_test, preds)
print(f"Accuracy with entropy: {accuracy}")
```

```
Accuracy with entropy: 0.847
```


Age ≤ 44.5 entropy = 0.721 samples = 5037 value = [6403, 1597] class = Not Exited
True
False
Balance ≤ 1884.345 entropy = 0.543 samples = 3829 value = [5322, 759] class = Not Exited
IsActiveMember ≤ 0.5 entropy = 0.988 samples = 1208 value = [1081, 838] class = Not Exited
CreditScore ≤ 395.0 entropy = 0.36 samples = 1466 value = [2137, 157] class = Not Exited
CreditScore ≤ 395.5 entropy = 0.632 samples = 2363 value = [3185, 602] class = Not Exited
Age ≤ 53.5 entropy = 0.927 samples = 522 value = [285, 548] class = Exited
NumOfProducts ≤ 2.5 entropy = 0.837 samples = 686 value = [796, 290] class = Not Exited
entropy = 0.0 samples = 2 value = [0, 3] class = Exited
NumOfProducts ≤ 1.5 entropy = 0.355 samples = 1464 value = [2137, 154] class = Not Exited
entropy = 0.0 samples = 5 value = [0, 9] class = Exited
Geography_France ≤ 0.5 entropy = 0.627 samples = 2358 value = [3185, 593] class = Not Exited
NumOfProducts ≤ 1.5 entropy = 0.988 samples = 381 value = [262, 340] class = Exited
entropy = 0.468 samples = 141 value = [23, 208] class = Exited
Geography_Germany ≤ 0.5 entropy = 0.783 samples = 656 value = [793, 241] class = Not Exited
entropy = 0.318 samples = 30 value = [3, 49] class = Exited
entropy = 0.728 samples = 310 value = [381, 97] class = Not Exited
NumOfProducts ≤ 2.5 entropy = 0.202 samples = 1154 value = [1756, 57] class = Not Exited
Geography_Germany ≤ 0.5 entropy = 0.68 samples = 1357 value = [1804, 396] class = Not Exited
Balance ≤ 51539.586 entropy = 0.543 samples = 1001 value = [1381, 197] class = Not Exited
CreditScore ≤ 561.5 entropy = 0.912 samples = 247 value = [129, 265] class = Exited
Balance ≤ 32985.805 entropy = 0.943 samples = 134 value = [133, 75] class = Not Exited
entropy = 0.666 samples = 494 value = [643, 135] class = Not Exited
Age ≤ 68.5 entropy = 0.979 samples = 162 value = [150, 106] class = Not Exited
entropy = 0.115 samples = 1118 value = [1731, 27] class = Not Exited
entropy = 0.994 samples = 36 value = [25, 30] class = Exited
Balance ≤ 178438.727 entropy = 0.527 samples = 487 value = [701, 95] class = Not Exited
EstimatedSalary ≤ 987.895 entropy = 0.75 samples = 870 value = [1103, 301] class = Not Exited
entropy = 1.0 samples = 11 value = [10, 10] class = Not Exited
Point Earned ≤ 728.5 entropy = 0.529 samples = 990 value = [1371, 187] class = Not Exited
entropy = 0.691 samples = 53 value = [15, 66] class = Exited
EstimatedSalary ≤ 93177.797 entropy = 0.946 samples = 194 value = [114, 199] class = Exited
NumOfProducts ≤ 2.5 entropy = 0.596 samples = 59 value = [77, 13] class = Not Exited
entropy = 0.998 samples = 75 value = [56, 62] class = Exited
Balance ≤ 156973.797 entropy = 0.999 samples = 139 value = [109, 103] class = Not Exited
entropy = 0.359 samples = 23 value = [41, 3] class = Not Exited
entropy = 0.467 samples = 467 value = [688, 76] class = Not Exited
entropy = 0.974 samples = 20 value = [13, 19] class = Exited
entropy = 0.544 samples = 5 value = [1, 7] class = Exited
entropy = 0.743 samples = 865 value = [1102, 294] class = Not Exited
Age ≤ 34.5 entropy = 0.461 samples = 648 value = [935, 101] class = Not Exited
Age ≤ 36.5 entropy = 0.646 samples = 342 value = [436, 86] class = Not Exited
entropy = 0.999 samples = 85 value = [70, 66] class = Not Exited
EstimatedSalary ≤ 192550.812 entropy = 0.809 samples = 109 value = [44, 133] class = Exited
entropy = 0.231 samples = 53 value = [77, 3] class = Not Exited
entropy = 0.0 samples = 6 value = [0, 10] class = Exited
entropy = 0.997 samples = 129 value = [91, 103] class = Exited
entropy = 0.0 samples = 10 value = [18, 0] class = Not Exited
entropy = 0.261 samples = 324 value = [476, 22] class = Not Exited
entropy = 0.602 samples = 324 value = [459, 79] class = Not Exited
Point Earned ≤ 757.5 entropy = 0.347 samples = 203 value = [288, 20] class = Not Exited
NumOfProducts ≤ 2.5 entropy = 0.891 samples = 139 value = [148, 66] class = Not Exited
EstimatedSalary ≤ 136674.883 entropy = 0.709 samples = 98 value = [31, 129] class = Exited
entropy = 0.787 samples = 11 value = [13, 4] class = Not Exited
entropy = 0.874 samples = 14 value = [12, 5] class = Not Exited
entropy = 0.293 samples = 189 value = [276, 15] class = Not Exited
entropy = 0.858 samples = 133 value = [148, 58] class = Not Exited
entropy = 0.0 samples = 6 value = [0, 8] class = Exited
entropy = 0.908 samples = 42 value = [22, 46] class = Exited
entropy = 0.462 samples = 56 value = [9, 83] class = Exited

Second Model: Logistics Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Drop 'Complain' and set 'Exited' as the target variable
X = bank_dummies.drop(columns=['Exited', 'Complain'])
y = bank_dummies['Exited'].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Feature scaling for better performance of the logistic regression
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


logistic_classifier = LogisticRegression(random_state=0)
logistic_classifier.fit(X_train, y_train)
```

```
[[1528   67]
 [ 309   96]]
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      1595
           1       0.59      0.24      0.34       405

    accuracy                           0.81      2000
   macro avg       0.71      0.60      0.61      2000
weighted avg       0.78      0.81      0.78      2000

Accuracy of the model: 0.812
```

```
# Apply SMOTE to oversample the minority class
smote = SMOTE(random_state=42)
X_smote, y_smote = smote.fit_resample(X, y)
```

```
[[3023  231]
 [ 410 2706]]
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      3254
           1       0.92      0.87      0.89      3116

    accuracy                           0.90      6370
   macro avg       0.90      0.90      0.90      6370
weighted avg       0.90      0.90      0.90      6370

Accuracy: 0.8993720565149137
```

Third Model: SVM

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

```
# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(classification_report(y_test, predictions))
print(f"SVM Accuracy without SMOTE: {accuracy}")
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1607
           1       1.00      1.00      1.00       393

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

SVM Accuracy without SMOTE: 0.999
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the pipeline
pipeline.fit(X_train, y_train)

# Make predictions
predictions = pipeline.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"SVM Accuracy with SMOTE: {accuracy}")
```

```
SVM Accuracy with SMOTE: 0.8035
```

THYNK UNLIMITED

Difficulties

- Finding out clear correlation between Complain and Exit. (Dataset flaw)
- Encoding process.
- Model finetuning.

THYNK UNLIMITED

THANKS.
Q&A