Churn Prediction Models: Reducing Costs for Banking Companies

Introduction

In banking, there are two key marketing expenses: **Acquisition Costs** and **Retention Costs**. Acquisition Costs refer to the investments required to attract new customers, while Retention Costs are the expenditures aimed at keeping existing customers satisfied.

Predicting customer behavior can be challenging, often leading to inaccurate forecasts about which customers will stay or leave. This misprediction can result in inefficient allocation of resources and increased costs.

Additionally, it is well-documented that acquiring a new customer is approximately seven times more expensive than retaining an existing one. Therefore, misjudging a customer's intent to remain when they actually leave can lead to unnecessary spending.

Objectives

This project aims to develop a sophisticated machine learning model to accurately predict which customers are likely to stay or leave. By improving prediction accuracy, we can optimize our spending on customer retention and reduce unnecessary costs.

Objective 1
Identify the key factors that influence customer retention and churn.

Objective 2
Develop a machine learning model to predict customer churn.

Objective 3
Minimize costs associated with customer acquisition and retention.

GitHub Repository

For more details on the code and project presentation, visit: GitHub Repository

Assumptions

We will make initial assumptions about the costs involved:

- **Retention Cost:** $10 per customer
- **Acquisition Cost:** $100 per customer (ten times higher than retention cost)

Retention Cost
Refers to the expenses incurred to keep existing customers and prevent them from leaving.

Acquisition Cost
Represents the costs associated with acquiring new customers and encouraging them to use the bank's services.

The Dataset

```
df = pd.read_csv('Customer-Churn-Records.csv')
```

```
df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82

The dataset we are using contains 10,000 records with 18 variables from a dataset of banking customers. This data includes information on various customer interactions, services provided and personal information by the bank. For more details, visit this link to access the dataset.

```
def report(df):
    col = []
    d_type = []
    uniques = []
    n_uniques = []

    for i in df.columns:
        col.append(i)
        d_type.append(df[i].dtypes)
        uniques.append(df[i].unique()[:5])
        n_uniques.append(df[i].nunique())

    return pd.DataFrame({'Column': col, 'd_type': d_type, 'unique_sample': uniques, 'n_uniques': n_uniques})
```

	Column	d_type	unique_sample	n_uniques
0	RowNumber	int64	[1, 2, 3, 4, 5]	10000
1	CustomerId	int64	[15634602, 15647311, 15619304, 15701354, 15737...	10000
2	Surname	object	[Hargrave, Hill, Onio, Boni, Mitchell]	2932
3	CreditScore	int64	[619, 608, 502, 699, 850]	460
4	Geography	object	[France, Spain, Germany]	3
5	Gender	object	[Female, Male]	2
6	Age	int64	[42, 41, 39, 43, 44]	70
7	Tenure	int64	[2, 1, 8, 7, 4]	11
8	Balance	float64	[0.0, 83807.86, 159660.8, 125510.82, 113755.78]	6382
9	NumOfProducts	int64	[1, 3, 2, 4]	4
10	HasCrCard	int64	[1, 0]	2
11	IsActiveMember	int64	[1, 0]	2
12	EstimatedSalary	float64	[101348.88, 112542.58, 113931.57, 93826.63, 79...	9999
13	Exited	int64	[1, 0]	2
14	Complain	int64	[1, 0]	2
15	Satisfaction Score	int64	[2, 3, 5, 4, 1]	5
16	Card Type	object	[DIAMOND, GOLD, SILVER, PLATINUM]	4
17	Point Earned	int64	[464, 456, 377, 350, 425]	785

From the dataframe, I will exclude the columns **CustomerId**, **RowNumber**, and **Surname** as they do not contribute to predicting customer churn.

```
df1 = df.drop(['CustomerId', 'RowNumber', 'Surname'], axis=1).copy()
```

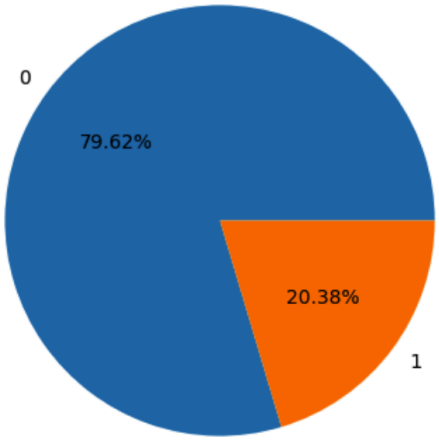
Data Analysis

Churn Ratio

```
df['Exited'].value_counts()
```

```
Exited
0    7962
1    2038
Name: count, dtype: int64
```

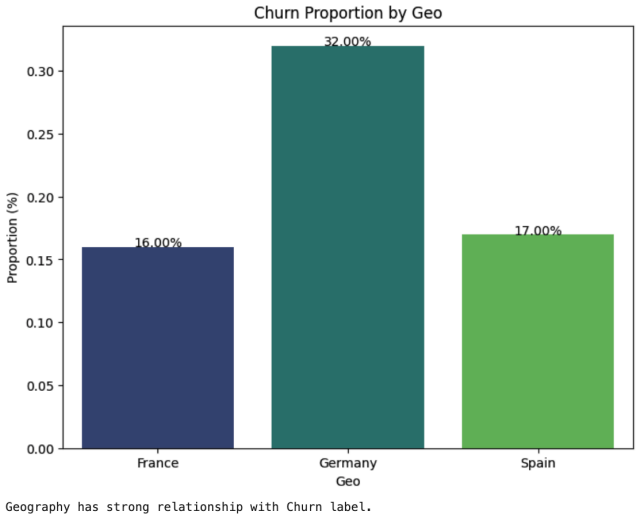
```
plt.figure(figsize=(10,5))
plt.pie(df['Exited'].value_counts(), labels=df['Exited'].value_counts().index, autopct='%.2f%%')
plt.show()
```



The pie chart reveals that 20.38% of the customers in this dataset are classified as churned. This suggests an imbalance between the number of customers who leave and those who remain. Although some might consider resampling the data to balance the labels, I prefer to conduct further analysis first to gain deeper insights before deciding on any data adjustments.

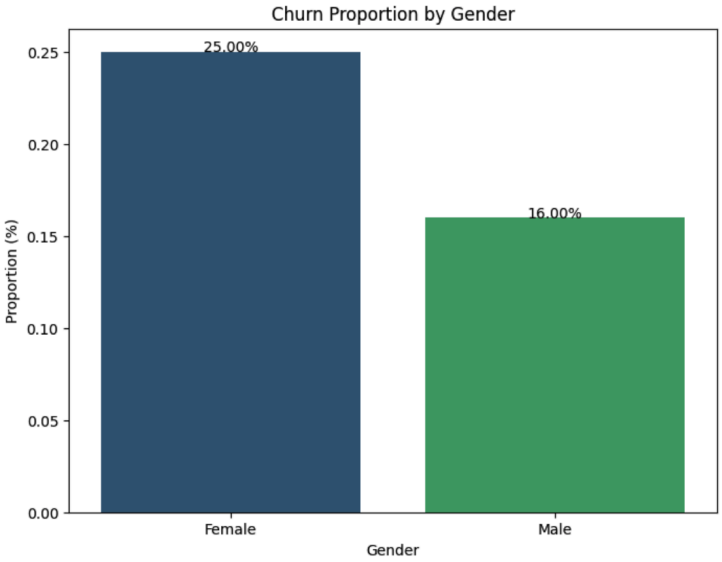
Churn by Categorical Features

Before proceeding with machine learning modeling, we will first analyze how each customer's demographic and behavioral characteristics affect their likelihood of churn.



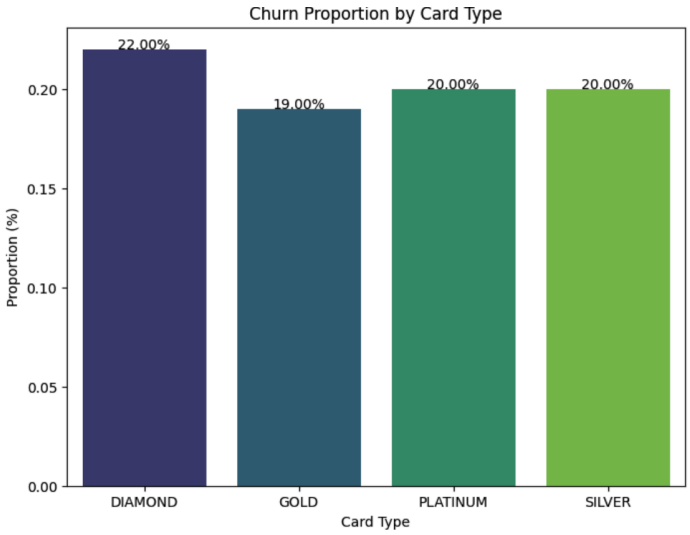
Geography has strong relationship with Churn label.

Germany have a **2 times greater chance of churning** compared to others.



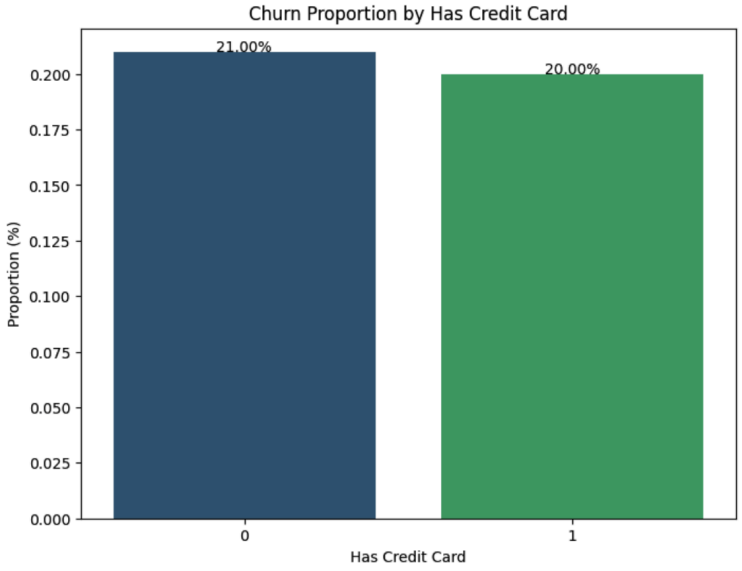

```
Gender has strong relationship with Churn label.
```

Customer Female are n**early twice as likely to churn** compared to Customer Male.



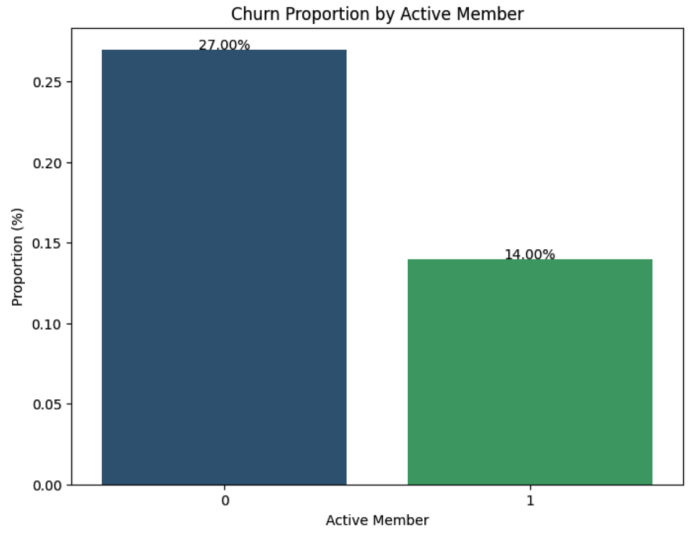

```
Card Type has weak relationship with Churn label.
```

The data indicates that **the churn rates are similar** across all card type.



Has Credit Card has weak relationship with Churn label.

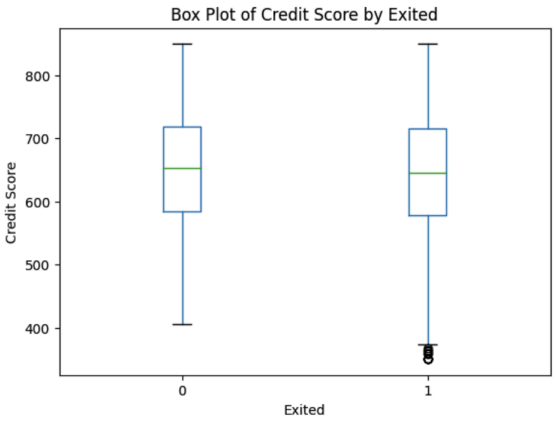
The data indicates that **the churn rates are similar** across both has credit card do not have credit card.



Active Member has strong relationship with Churn label.

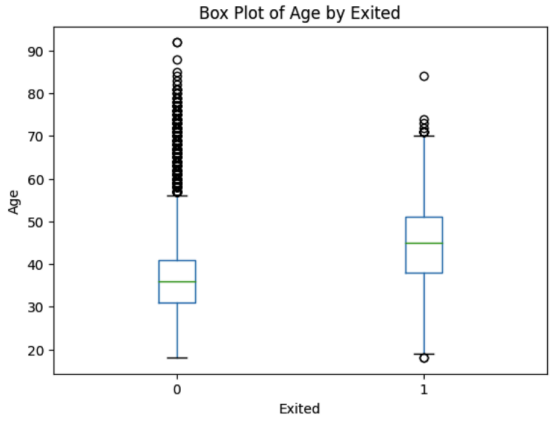
Customer Not Active Member are **nearly two times more likely to churn** compared to Active Member.

Churn by Numerical Features



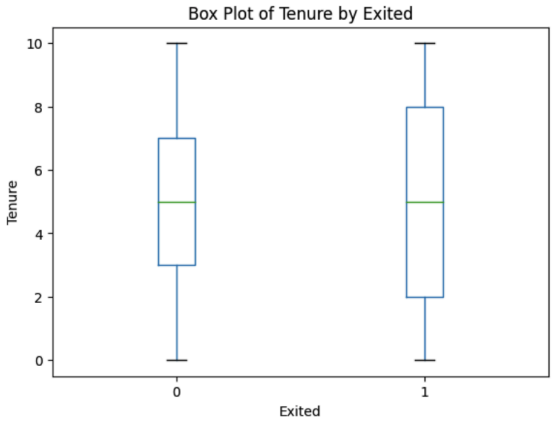

```
Credit Score has strong relationship with Churn label.
```

We will also use an assessment using the t-test to examine the relationship between the feature and the target.The analysis indicates that the credit score feature **show a strong correlation with churn**.



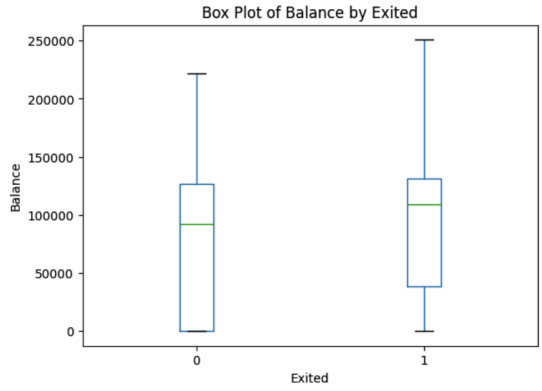

```
Age has strong relationship with Churn label.
```

It appears that the 'Age' feature also **show a significant relationship with churn**.



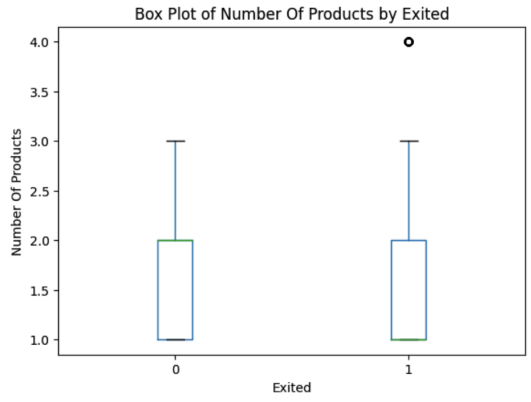

```
Tenure has weak relationship with Churn label.
```

It seems that the 'Tenure' (Year) column **show a weak relationship with churn.**



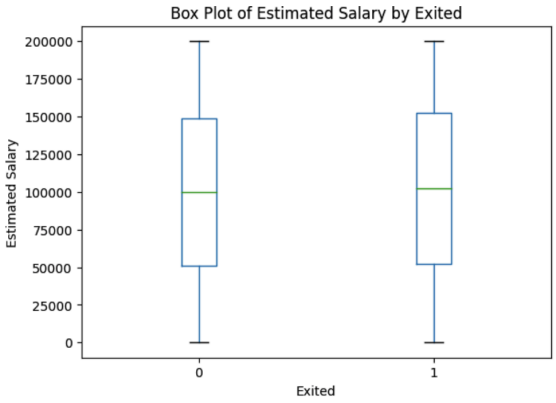

```
Balance has strong relationship with Churn label.
```

It appears that the 'Balance' feature also **show a significant relationship with churn**.



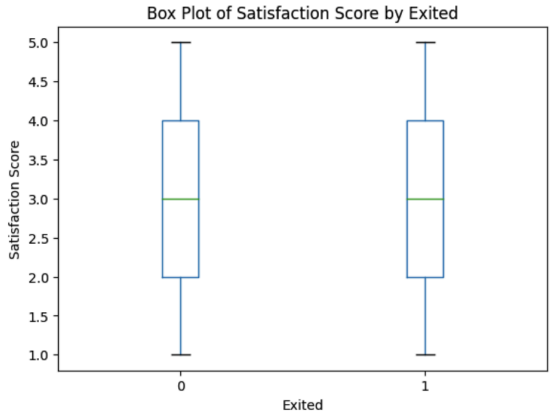

```
Number Of Products has strong relationship with Churn label.
```

It appears that the 'Number of Products' feature also **show a significant relationship with churn**.



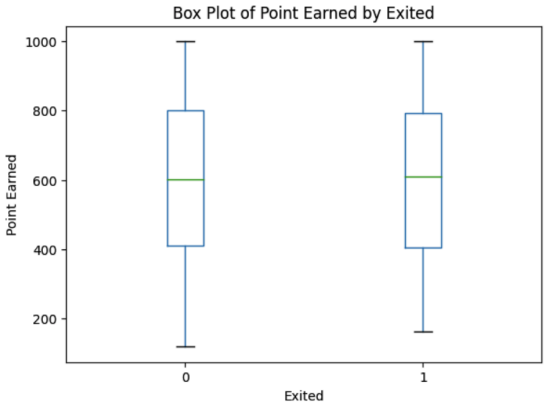

```
Estimated Salary has weak relationship with Churn label.
```

It seems that the 'Estimated Salary' column **show a weak relationship with churn.**




```
Satisfaction Score has weak relationship with Churn label.
```

It seems that the 'Satisfaction Score' column also **show a weak relationship with churn.**

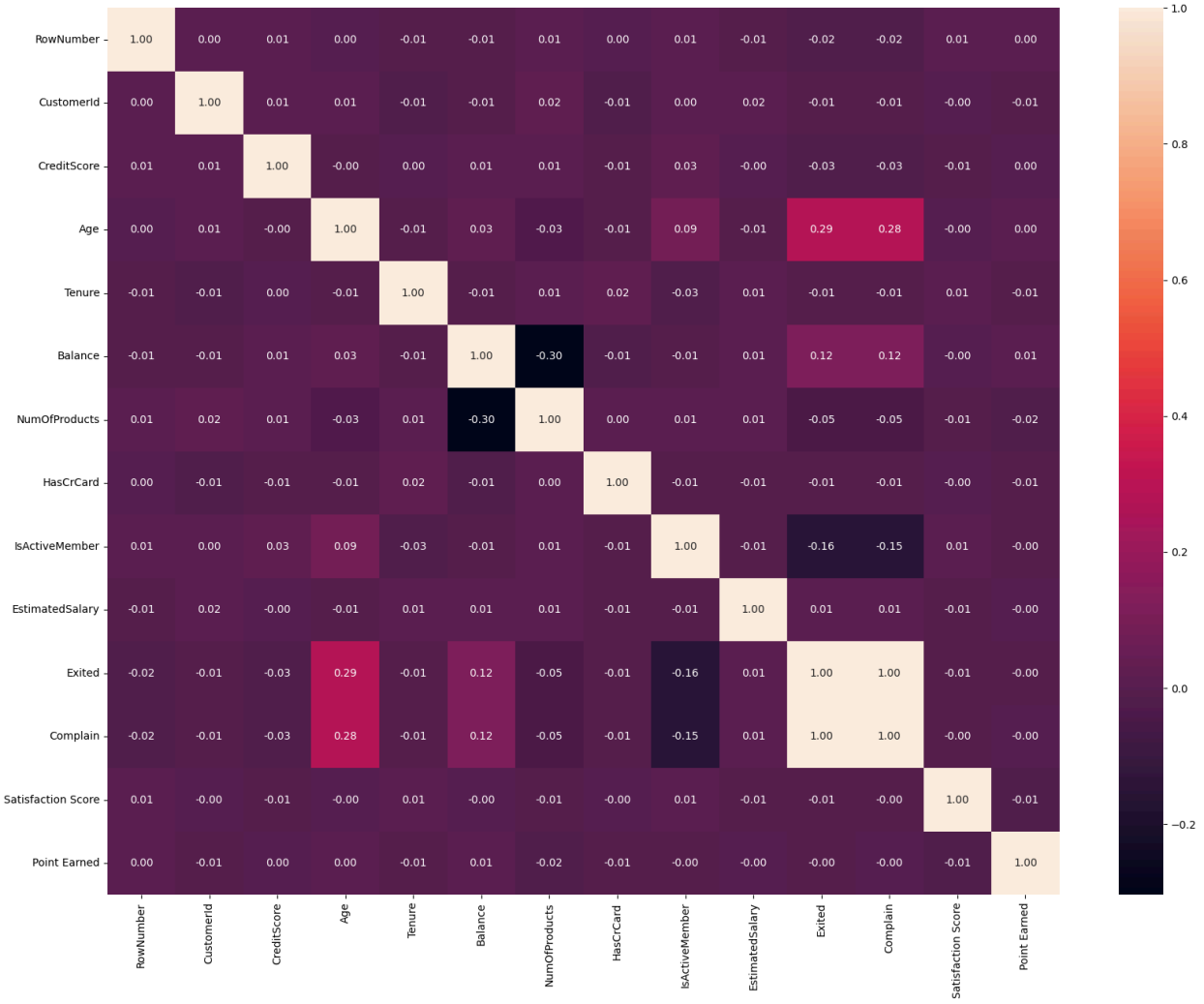



```
Point Earned has weak relationship with Churn label.
```

It seems that the 'Point Earned' column also **show a weak relationship with churn.**

Checking Multicollinearity

In addition to evaluating the relationship between dependent variable and the independent variables, we will also assess potential multicollinearity among numerical variables using correlation heatmap.



The columns ‘Complain,’ ‘Exited,’ have very high correlation and we should remove column Complain.

Modeling Strategies

For the modeling stage, our primary goal is to accurately predict customer churn. Given that predicting a customer as 'retained' when they will actually 'churn' results in losing that customer, and considering that customer acquisition costs are approximately ten times higher than retention costs, we will focus on optimizing the Recall Score for this model.

The strategy involves splitting the dataset into two main parts: 80% for Training and 20% for Testing. The Training Data will be further divided into Training and Validation sets, with the Validation Data used to evaluate the model's performance and refine its predictive capabilities.

```
X = df1.drop(columns=['Exited', 'Complain'])
y = df1['Exited']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.2, stratify=y, random_state=42)
```

Initiating Machine Learning Pipeline

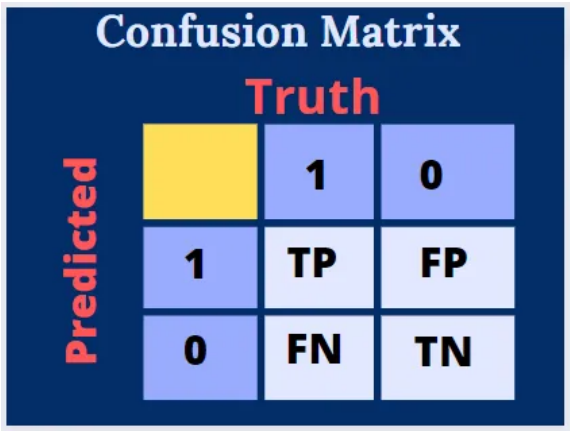
The next step involves constructing a Pipeline to manage data processing up to the model fitting stage. Why use a Pipeline? The primary reason is to prevent data leakage.

Data leakage happens when information from the testing dataset inadvertently influences the training process. This means the model might gain access to information it shouldn't have during training, leading to overly optimistic performance estimates that do not reflect real-world performance.

```
one_hot_cols = X.select_dtypes(include='object').columns
numeric_cols = X.select_dtypes(exclude='object').columns
logit = LogisticRegression(random_state=42)
smote = SMOTE(random_state=42)

logit_pipe_num = Pipeline([
    ('imputer', SimpleImputer(strategy='median', missing_values=np.nan)),
    ('scaler', RobustScaler()),
])

# for all object columns
logit_pipe_cat = Pipeline([
    ('onehot', OneHotEncoder(drop='first')),
])

# transforming all columns
logit_transformer = ColumnTransformer([
    ('pipe_num', logit_pipe_num, numeric_cols),
    ('pipe_cat', logit_pipe_cat, one_hot_cols)
])

# combine all pipeline
logit_pipe_combine = Pipeline([
    ('transformer', logit_transformer),
    ('resampling', smote),
    ('logit', logit)
])
```

Determine Metric Evaluation

Our goal is to optimize the Recall value. Given our primary objective to minimize costs—where the greatest expense is acquisition cost—we aim to reduce the likelihood of customers who are prone to churn but do not receive adequate retention efforts. By focusing on Recall, we strive to identify as many potential churners as possible to avoid incurring higher acquisition costs.



Cross Validation

```
logit_score
```

```
array([0.69631902, 0.67177914, 0.67177914, 0.66871166, 0.67177914])
```

Next, we will perform cross-validation to evaluate the model's stability as it learns from different subsets of the training data. This process will help us understand how well the model generalizes to varying training conditions. From this stage, we anticipate achieving an average Recall score of 0.676, or 67.6% which is not a good score.

```
logit_pipe_combine.fit(X_train, y_train)
```

```
Pipeline(steps=[('transformer',
                 ColumnTransformer(transformers=[('pipe_num',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer(strategy='median')),
                                                                  ('scaler',
                                                                   RobustScaler())]),
                                                  Index(['CreditScore', 'Age', 'Tenure', 'Balance',
       'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Satisfaction Score',
       'Point Earned'],
      dtype='object')),
                                                 ('pipe_cat',
                                                  Pipeline(steps=[('onehot',
                                                                   OneHotEncoder(drop='first'))]),
                                                  Index(['Geography', 'Gender', 'Card Type'], dtype
='object'))])),
                ('resampling', SMOTE(random_state=42)),
                ('logit', LogisticRegression(random_state=42))])
```

Following this, we will proceed with the training phase, where we obtain a data test Recall score of 0.715, or 71.5% and data train Recall score of 0.689 or 68.9%. This result is quite close to the

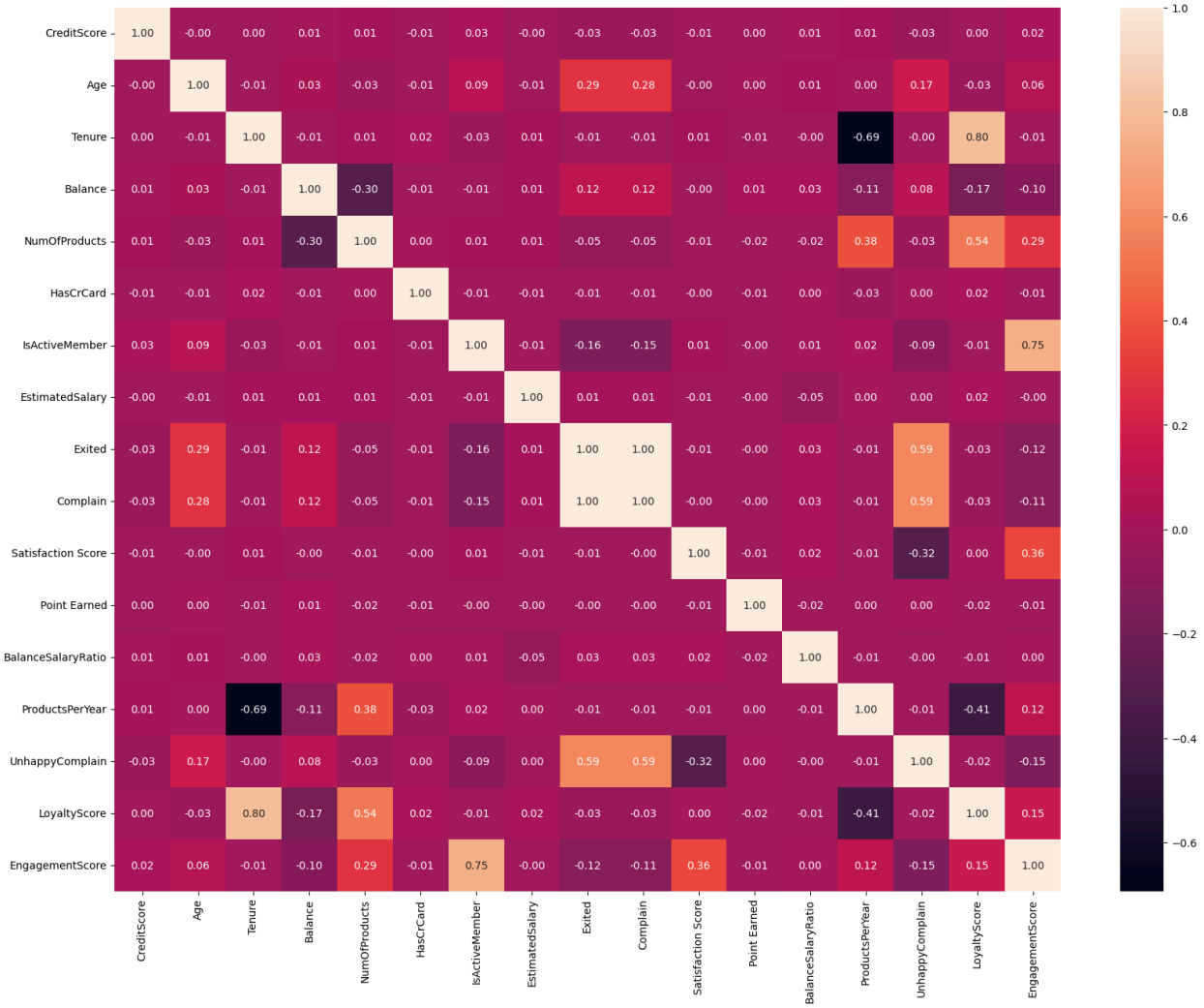
cross-validation score, indicating consistent performance across both training and validation stages but this score is to low which is indicating **underfitting**. The score we expect from the machine learning model calculation is around 85%.

Add New Features

Here, we will add features using feature engineering. The features we want to include in the model are:

Age Group, Balance Salary Ratio, Products per Year, Unhappy Complain, Loyalty Score, Engagement Score, Customer Lifetime Value, Credit Score Group, Activity Level, Satisfaction Category, Income Level, Tenure Group, Point Earned Category.

Let's re-examine the correlation between features to see if there are any indications of multicollinearity after adding the new feature.



It was found that there are multicollinear features, so we will remove those features. The features to be removed are Complain, Tenure, and Engagement Score.

```
X = df1.drop(columns=['Exited', 'Complain', 'Tenure', 'EngagementScore'])
y = df1['Exited']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.2, stratify=y, random_state=42)
```

Let's perform the calculations again using the logistic regression model.

```
logit_score
```

```
array([0.79141104, 0.85276074, 0.82208589, 0.81595092, 0.78220859])
```

```
recall_score(y_test,logit_pipe_combine.predict(X_test))
```

```
0.8700980392156863
```

```
recall_score(y_train,logit_pipe_combine.predict(X_train))
```

```
0.8208588957055215
```

we obtain a data test Recall score of 0.8700, or 87.0% and data train Recall score of 0.82 or 82.0%. It can be see, that the score has improved since adding the new feature. We will also try using the XGBoost model to see if its performance can provide better results.

Model Machine Learning XGBOOST

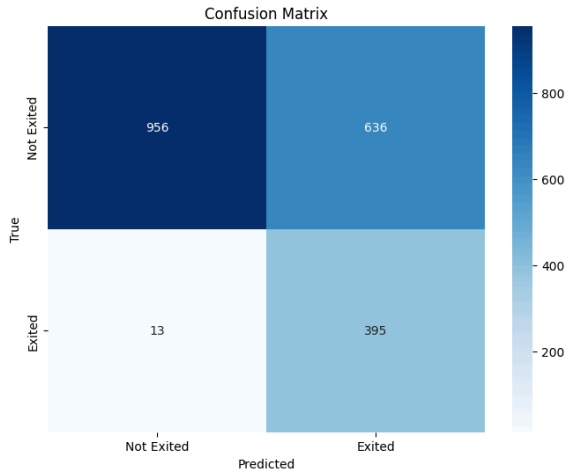
```
one_hot_cols = X.select_dtypes(include='object').columns
numeric_cols = X.select_dtypes(exclude='object').columns

xgb_pipe_num = Pipeline([
    ('imputer', SimpleImputer(strategy='median', missing_values=np.nan)),
    ('scaler', RobustScaler())
])

xgb_pipe_cat = Pipeline([
    ('onehot', OneHotEncoder(drop='first')),
])

xgb_transformer = ColumnTransformer([
    ('pipe_num', xgb_pipe_num, numeric_cols),
    ('pipe_cat', xgb_pipe_cat, one_hot_cols)
])

xgb = XGBClassifier(
    random_state=42,
    eval_metric='logloss'
)

xgb_pipe_combine = ImbPipeline([
    ('transformer', xgb_transformer),
    ('resampling', SMOTE(random_state=42)),
    ('xgb', xgb)
])
```

We will proceed with hyperparameter tuning here to improve the performance of the XGBoost model.

```
param_dist = {
    'xgb__n_estimators': [50, 100, 150],
    'xgb__learning_rate': np.linspace(0.01, 0.1, 10),
    'xgb__max_depth': [3, 4],
    'xgb__subsample': np.linspace(0.6, 0.9, 4),
    'xgb__colsample_bytree': np.linspace(0.6, 0.9, 4),
    'xgb__gamma': np.linspace(0, 0.5, 6),
    'xgb__min_child_weight': [3, 4],
    'xgb__scale_pos_weight': [1, 2],
    'xgb__lambda': [0.1, 0.5]
}

random_search = RandomizedSearchCV(
    estimator=xgb_pipe_combine,
    param_distributions=param_dist,
    n_iter=50,
    scoring='recall',
    cv=5,
    random_state=42,
    n_jobs=-1,
    verbose=2
)

random_search.fit(X_train, y_train)

best_params = random_search.best_params_
best_score = random_search.best_score_

y_pred = random_search.predict(X_test)
test_score = recall_score(y_test, y_pred)

print(f"Best Parameters: {best_params}")
print(f"Best Training Recall Score: {best_score}")
print(f"Test Recall Score: {test_score}")
```

We have data train recall score is 0.97 or 97% and data test recall score is 0.96 or 96%. The recall score for both the training and test data is very good here. There is no indication of overfitting or underfitting.

Next, we will calculate the confusion matrix to make business decisions.



Recall Interpretation

If we have 1,000 customers with a potential to churn, our model accurately predicts that 960 of them will churn. This allows us to allocate a retention budget of $9,600 (assuming a retention

cost of $10 per customer). However, there are 40 customers we did not predict accurately, requiring us to spend $4,000 on acquisition costs to replace them (assuming an acquisition cost of $100 per customer). This results in a total expenditure of approximately $10,000.

Now, what if we didn't use machine learning?

Without machine learning, the most optimistic scenario would involve a 50% accuracy rate. This means we would correctly identify 500 customers who are likely to churn. Consequently, we would allocate a retention budget of $5,000 (at $10 per customer). However, for the 500 customers we incorrectly predicted, we would need to spend $50,000 on acquisition costs (at $100 per customer) to replace them. This results in a total expenditure of approximately $55,000.

Precision Effect

We also need to consider Precision in our evaluation. To interpret this, we should first revisit the Confusion Matrix that we generated earlier.

From this, we can infer that with 960 correct predictions of churn (True Positives) and a Precision of 38%, we have approximately 1,567 false positives (FP). This means the total number of churn predictions made is 2,527.

Out of the 2,527 customers predicted to churn, we accurately identified 960 who actually did churn. The remaining customers were incorrectly predicted as churn but were retained. Consequently, we need to allocate retention costs for these incorrectly predicted customers. In other words, from the total retention expenditure of $252,700, only $9,600 was effectively spent on customers who genuinely churned, while the rest was allocated to customers who did not.

Cost Effectiveness

$$(\$10,000 + \$15,670) - \$55,000 = -\$29,330$$ **= -53%**

This means that if we total and compare the costs incurred between using the Machine Learning model and not using the Machine Learning model, we can save expenses by up to 53%. Of course, this doesn't yet take into account the precision of our predictions if we don't use machine learning. If we include that assessment, the savings ratio might be even higher.