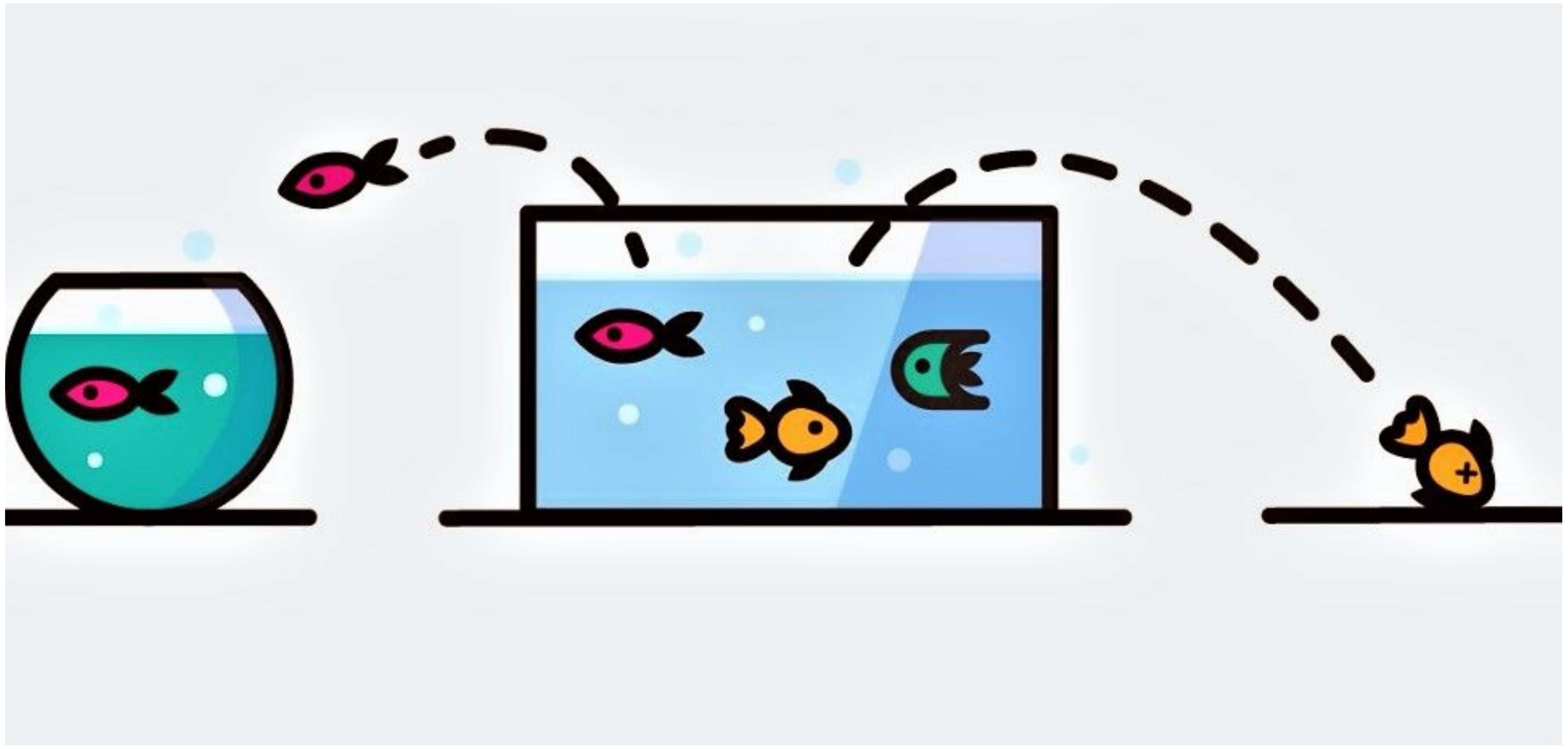
Customer Churn Prediction

Business Problem

A manager at the bank is disturbed with more and more customers leaving their credit card services. They would really appreciate if one could predict for them who is gonna get churned so they can proactively go to the customer to provide them better services and turn customers' decisions in the opposite direction.

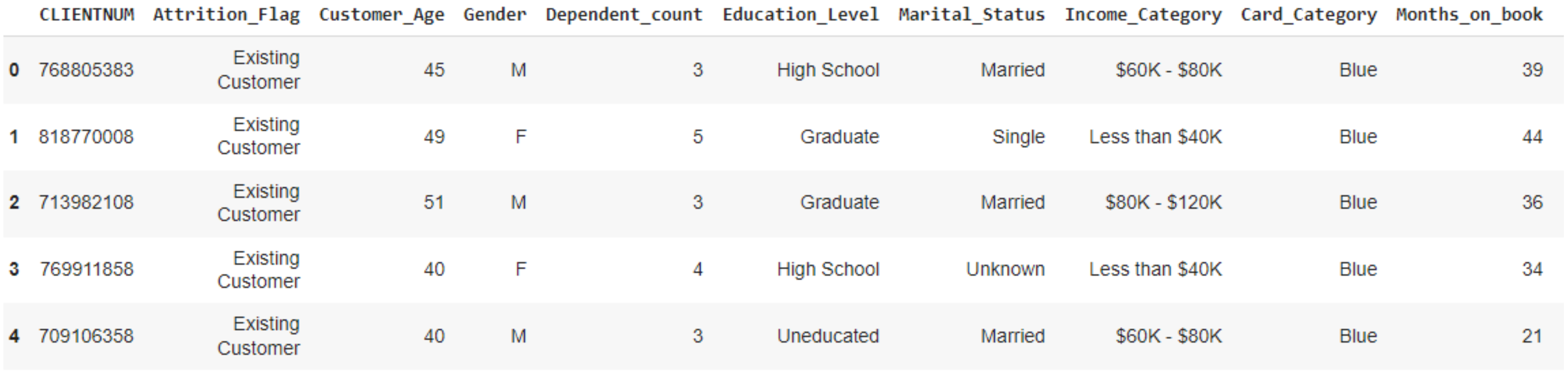
Customer Churn

Customer churn or ***customer attrition*** is the phenomenon where customers of a business no longer purchase or interact with it.



Dataset

- Dataset shape (10127,21)
- Each observation represents a customer
- Some features: *age*, *salary*, *marital status*, *credit card limit*, *credit card category*, etc.
- Target variable: ***Attrition_Flag***
- Basically the dataset is a snapshot of the clients in a 12 months period
- No missing values



	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - $80K	Blue	39
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - $120K	Blue	36
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	34
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - $80K	Blue	21

Procedure

1. Exploratory Data Analysis

This will help us to get some insights and have a preliminary idea on the features that are causing clients to leave the organization

2. Data Preprocessing and Model Building

In order to build a model that can predict whether a customer is going to leave or not, so as to take action before it happens

3. Clustering and Model Comparison

To test whether clusters can split the target variable and improve the performance of the model

Exploratory Data Analysis

Some Numbers

Categorical

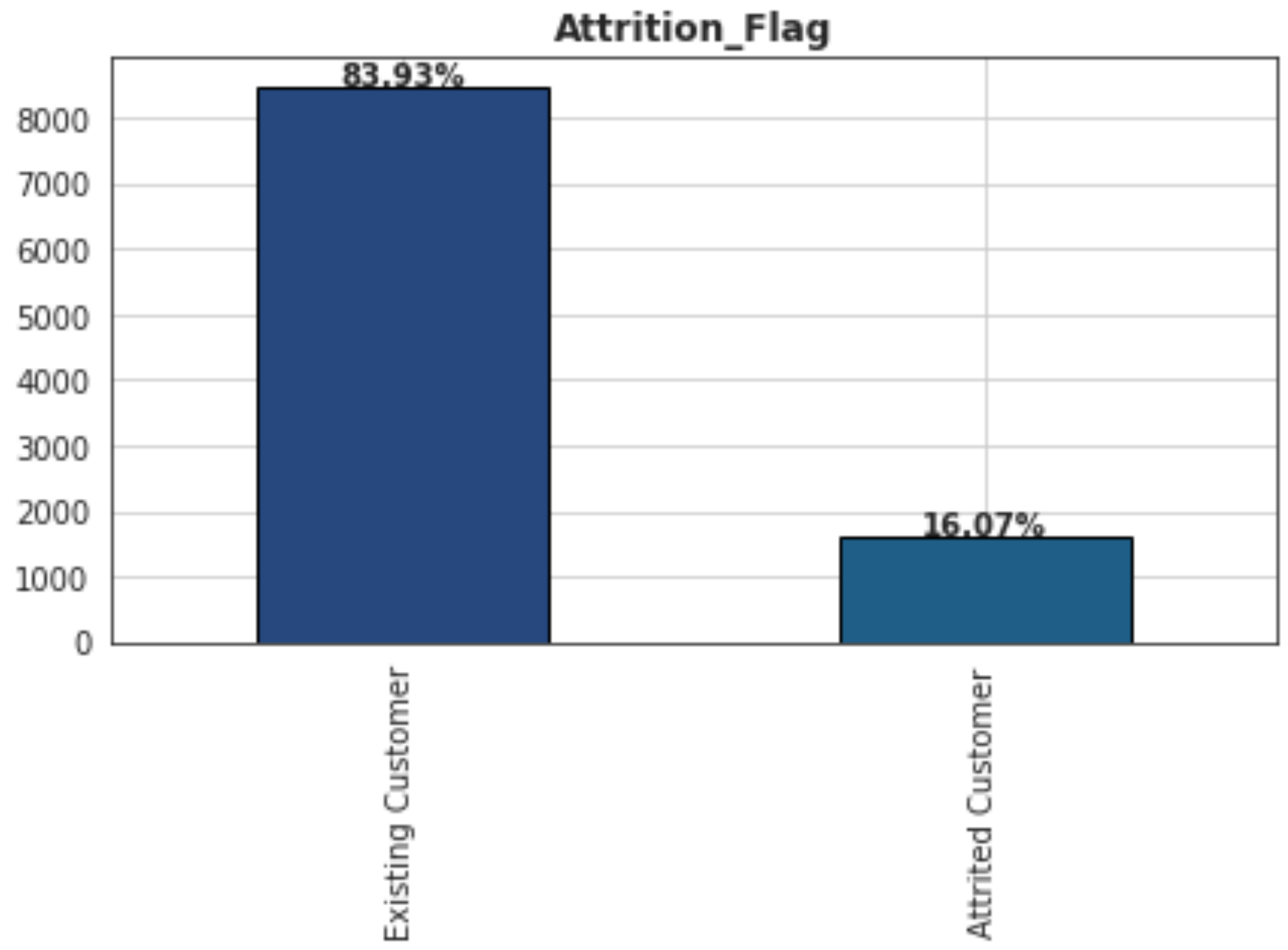
- 31% of the customers are graduated
- 35% has an income less than $40K
- 93% has the Blue Card
- 53% has 2 or 3 dependents

Numerical

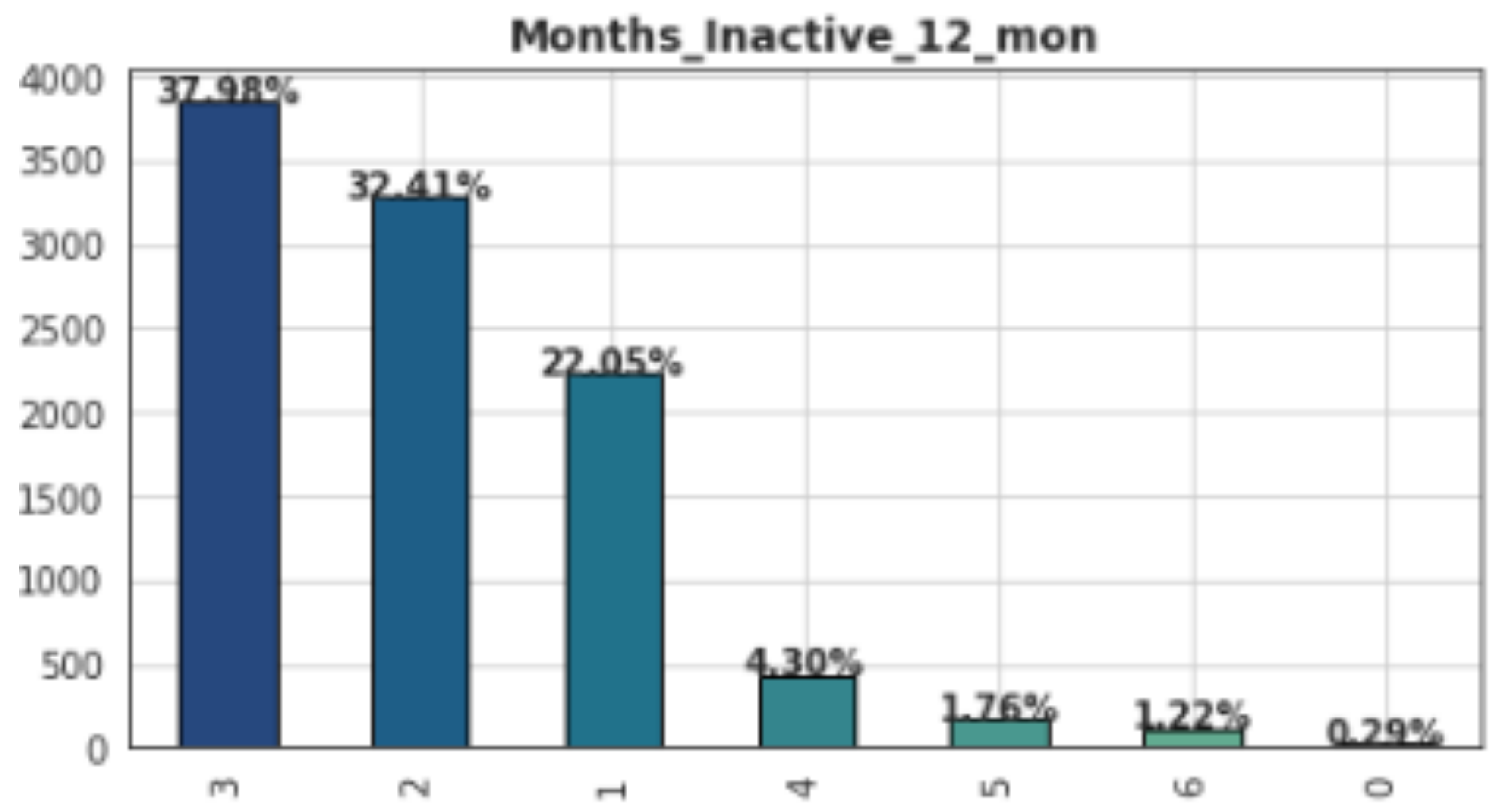
- Mean age is 46
- Median credit limit is $4549
- 75% has a total transactions amount of $4741 or less
- Median total transactions count is 67

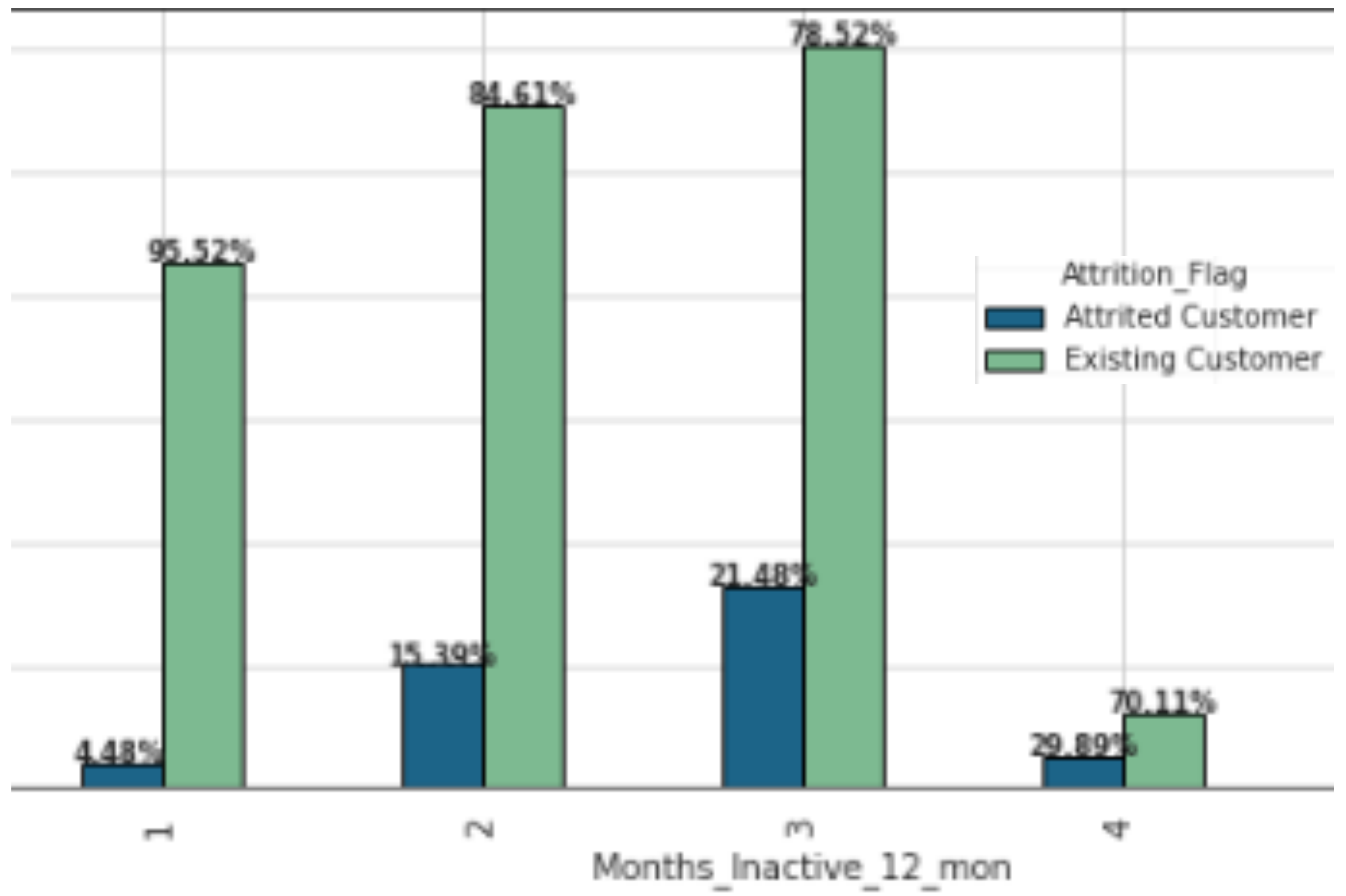
Unbalanced Dataset

Only 16% of customers left the bank. This is a clear example of an ***unbalanced dataset***.



No. of months of inactivity in the last 12 months

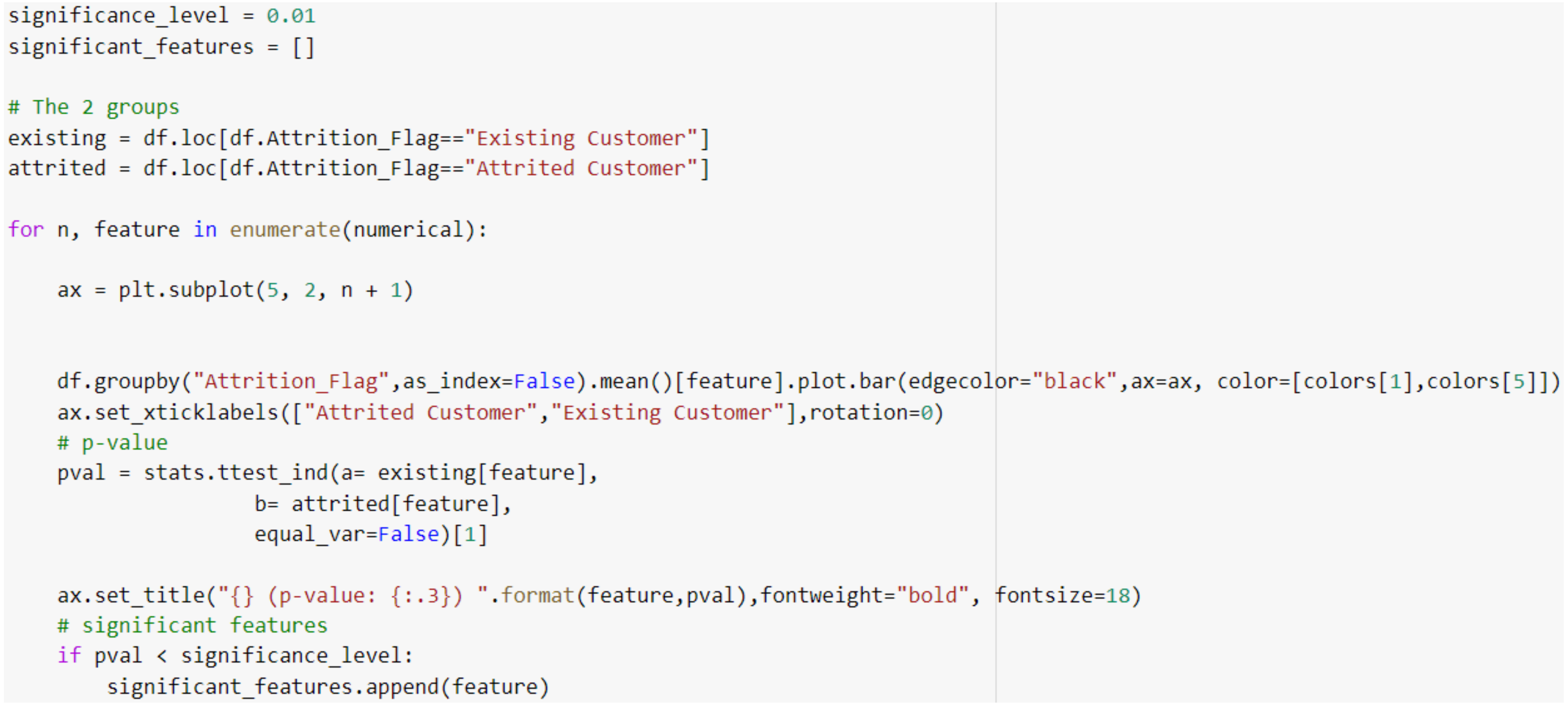




It can be seen that as months of inactivity grow, the percentage of clients leaving increases. We have only taken into account the first 4 months of inactivity since they represent almost 97% of the data.

T-Test on The Target Variable

A t-test is a statistical test that is used to compare the ***means of two groups***. It is often used in hypothesis testing to determine whether two groups are different from one another.



```
significance_level = 0.01
significant_features = []

# The 2 groups
existing = df.loc[df.Attrition_Flag=="Existing Customer"]
attrited = df.loc[df.Attrition_Flag=="Attrited Customer"]

for n, feature in enumerate(numerical):

    ax = plt.subplot(5, 2, n + 1)


    df.groupby("Attrition_Flag",as_index=False).mean()[feature].plot.bar(edgecolor="black",ax=ax, color=[colors[1],colors[5]])
    ax.set_xticklabels(["Attrited Customer","Existing Customer"],rotation=0)
    # p-value
    pval = stats.ttest_ind(a= existing[feature],
                    b= attrited[feature],
                    equal_var=False)[1]

    ax.set_title("{} (p-value: {:.3}) ".format(feature,pval),fontweight="bold", fontsize=18)
    # significant features
    if pval < significance_level:
        significant_features.append(feature)
```

Results

Significant Features (p-value < 0.01):

Total_Revolving_Bal

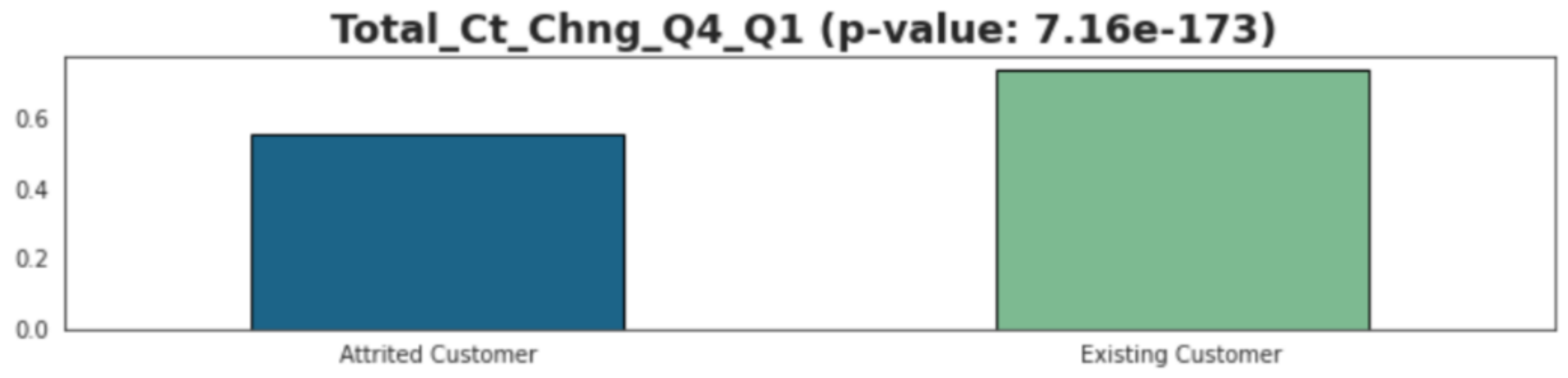
Total_Amt_Chng_Q4_Q1

Total_Trans_Amt

Total_Trans_Ct

Total_Ct_Chng_Q4_Q1

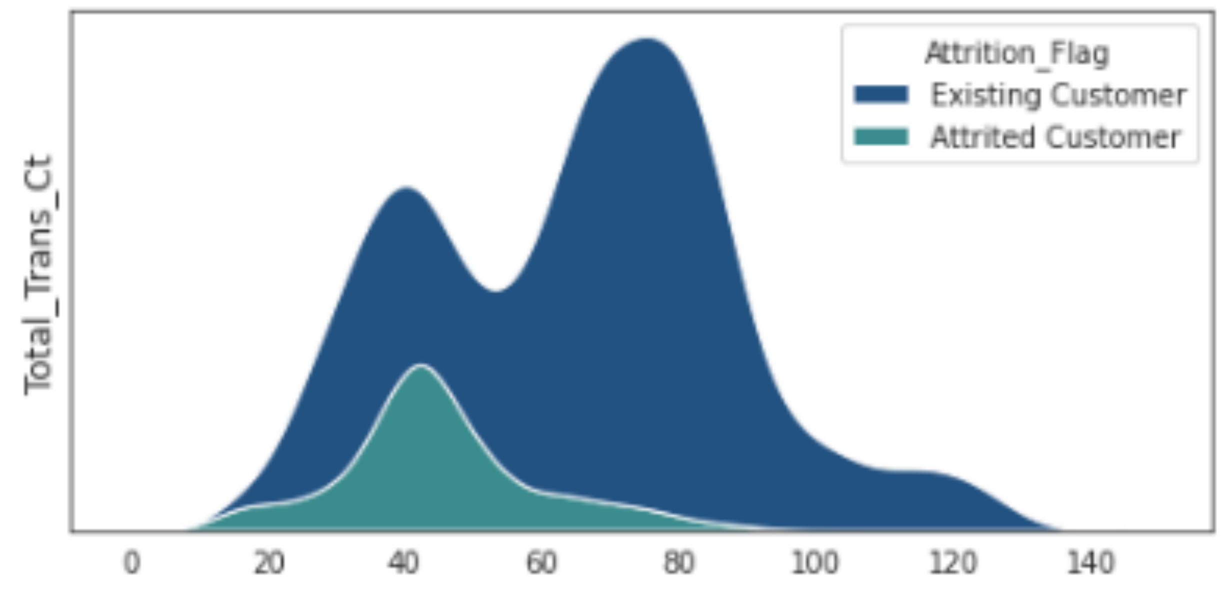
Avg_Utilization_Ratio

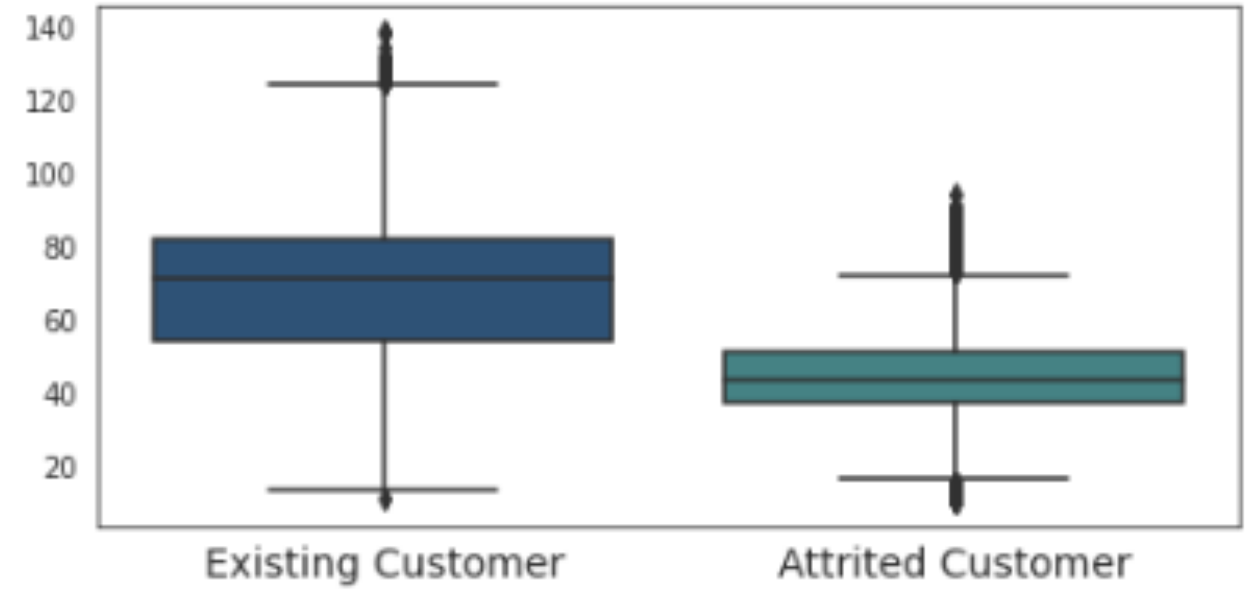


Variable with the lowest p-value

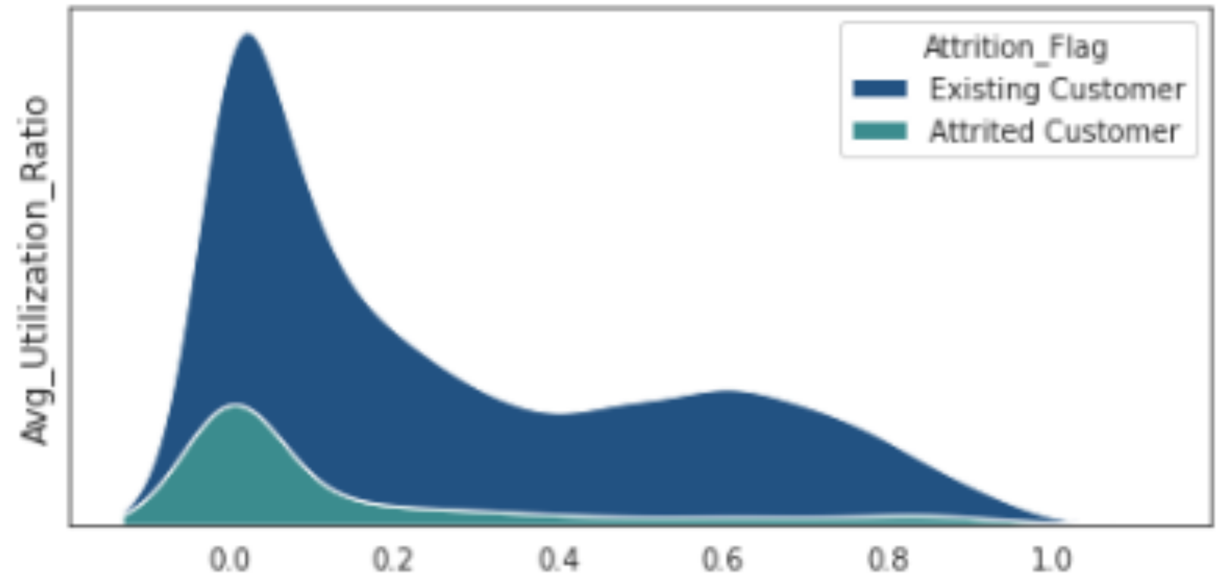
We can see that all the significant features are **related to the client activity levels**, except for *Total_Revolving_Bal* which is the total revolving balance on the credit card.

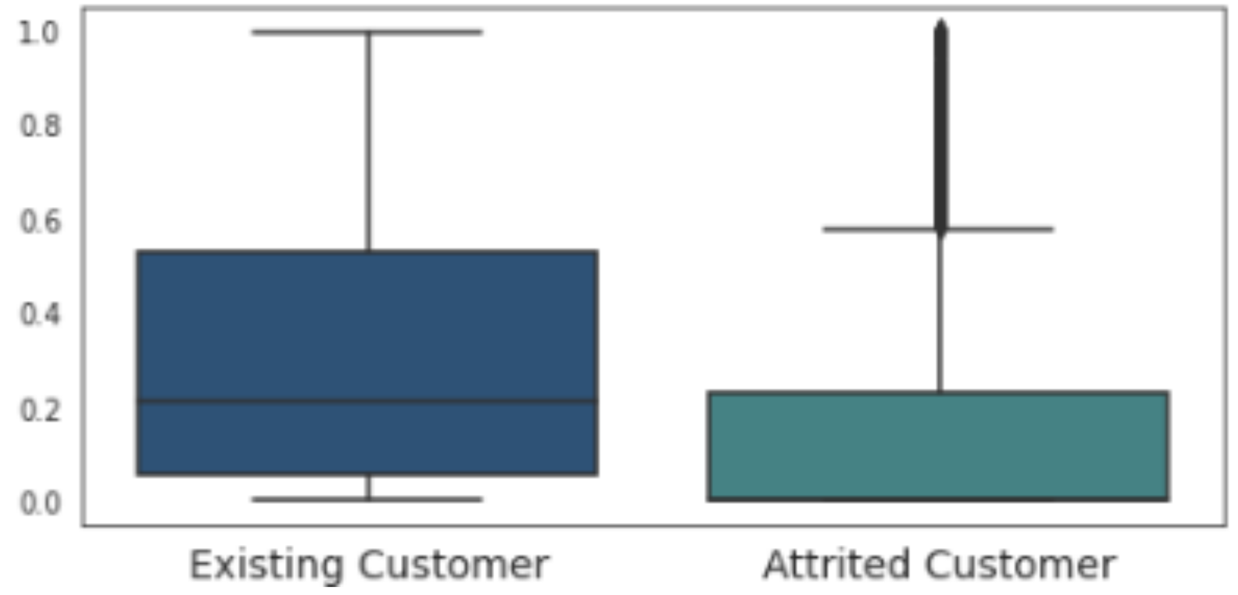
Let's try with the Median



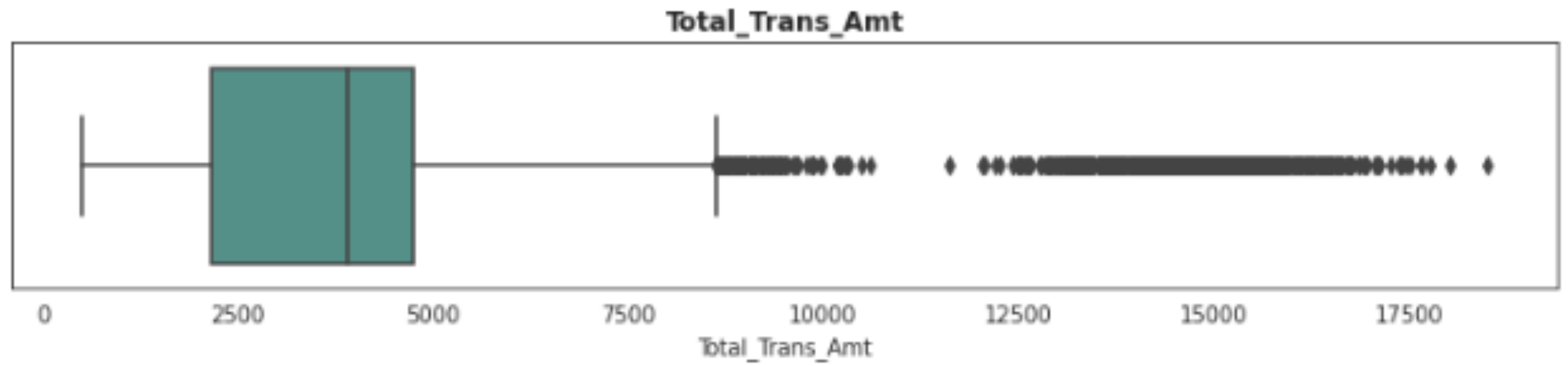


The previous results are confirmed even if we use the median instead of the mean, as can be seen in the two graphs used as examples.





Outliers



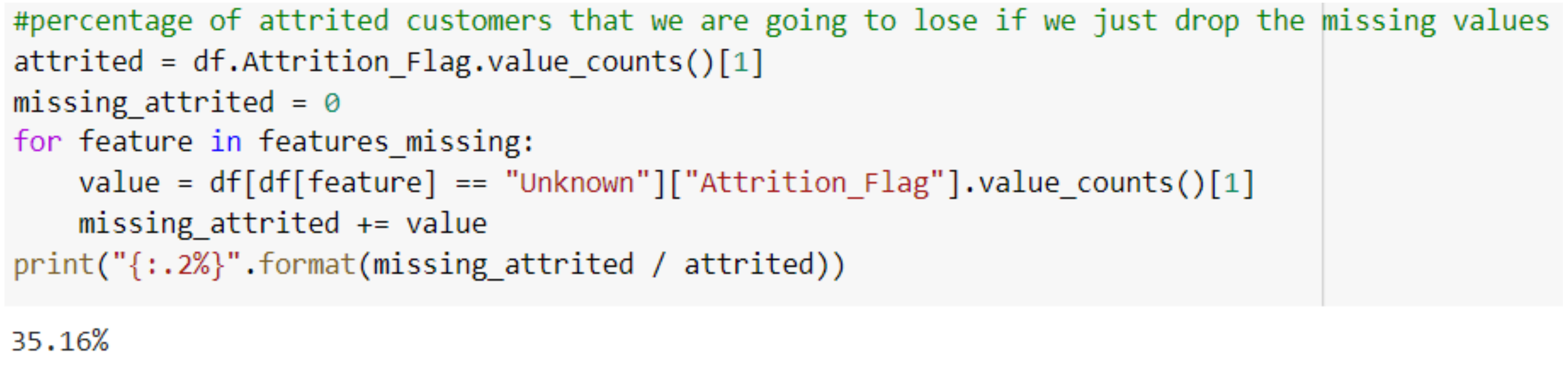
Some of the numerical variables present outliers. Removing them could lead to better performance of a predictive model, but in this case they are ***natural outliers*** (and ***not recording errors***) so they should not be removed. In the image it can be seen that some clients may exceed $17500 in total transactions amount.

What EDA Suggests

- The customers of the bank are ***relatively young with lower income levels***
- The most important features that help to predict whether a customer is going to leave are those related to the ***activity levels***
- Outliers ***shouldn't always be removed***, especially if they aren't due to recording errors

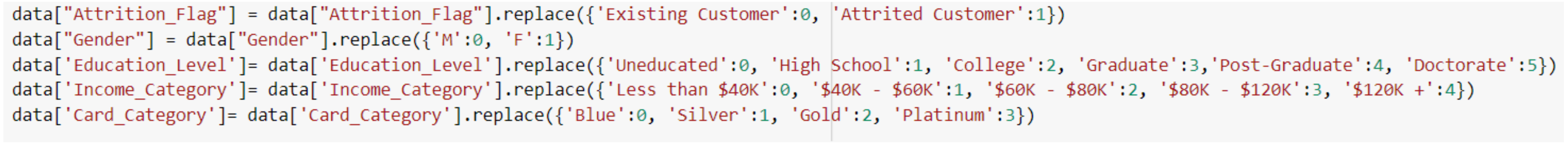
Data Preprocessing and Model Building

Label Encoding

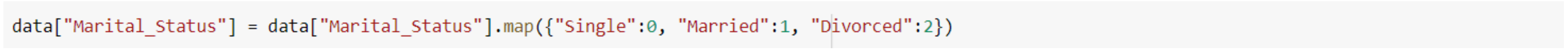


```python
#percentage of attrited customers that we are going to lose if we just drop the missing values
attrited = df.Attrition_Flag.value_counts()[1]
missing_attrited = 0
for feature in features_missing:
    value = df[df[feature] == "Unknown"]["Attrition_Flag"].value_counts()[1]
    missing_attrited += value
print("{:.2%}".format(missing_attrited / attrited))
```

```
35.16%
```



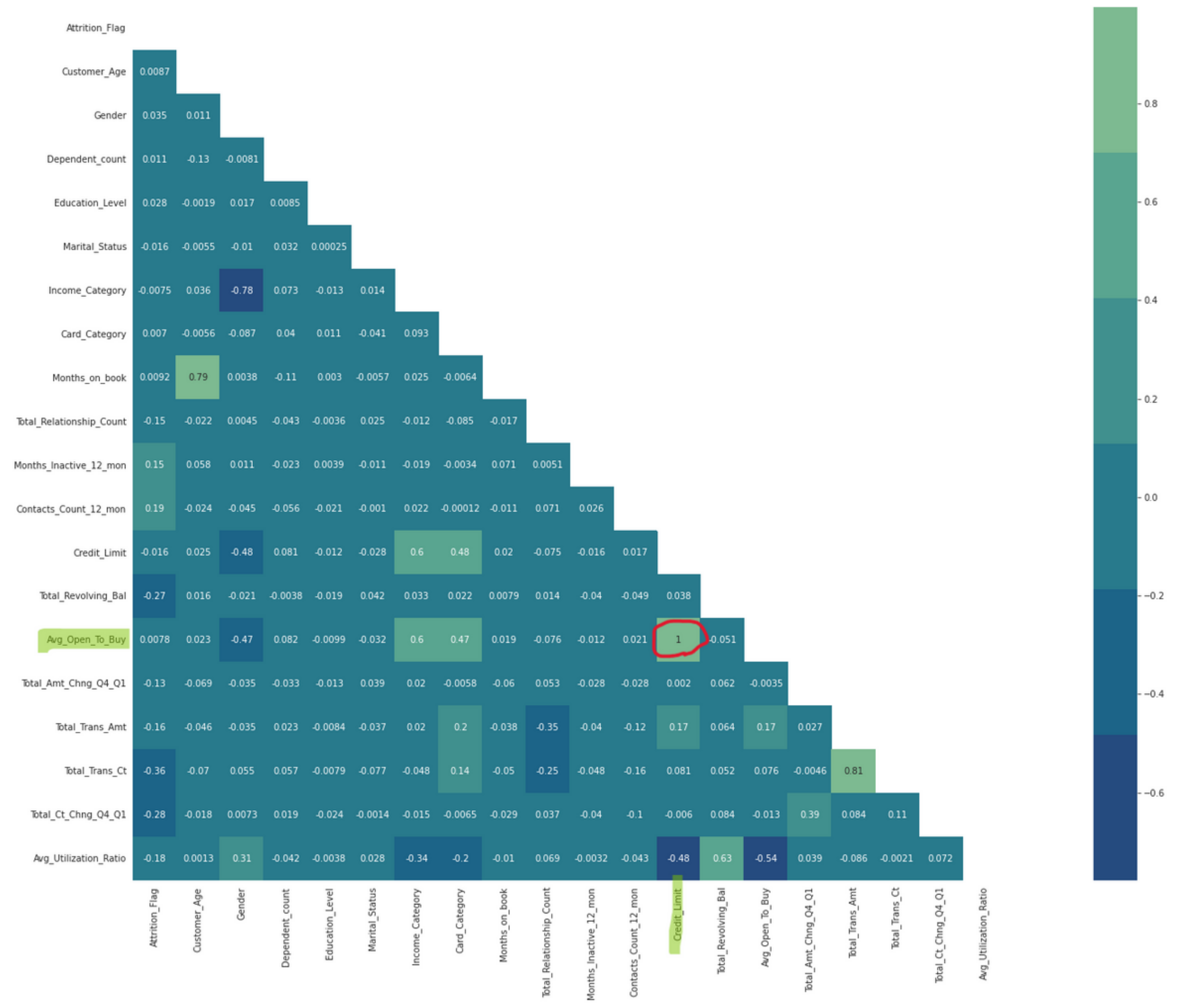
```python
data["Attrition_Flag"] = data["Attrition_Flag"].replace({'Existing Customer':0, 'Attrited Customer':1})
data["Gender"] = data["Gender"].replace({'M':0, 'F':1})
data['Education_Level']= data['Education_Level'].replace({'Uneducated':0, 'High School':1, 'College':2, 'Graduate':3,'Post-Graduate':4, 'Doctorate':5})
data['Income_Category']= data['Income_Category'].replace({'Less than $40K':0, '$40K - $60K':1, '$60K - $80K':2, '$80K - $120K':3, '$120K +':4})
data['Card_Category']= data['Card_Category'].replace({'Blue':0, 'Silver':1, 'Gold':2, 'Platinum':3})
```



```python
data["Marital_Status"] = data["Marital_Status"].map({"Single":0, "Married":1, "Divorced":2})
```

After seeing the low number of rows with the value *'Unknown'*, we decided to delete them. Then we have changed the value inside the categorical columns from strings to **numerical** in order to develop our model.

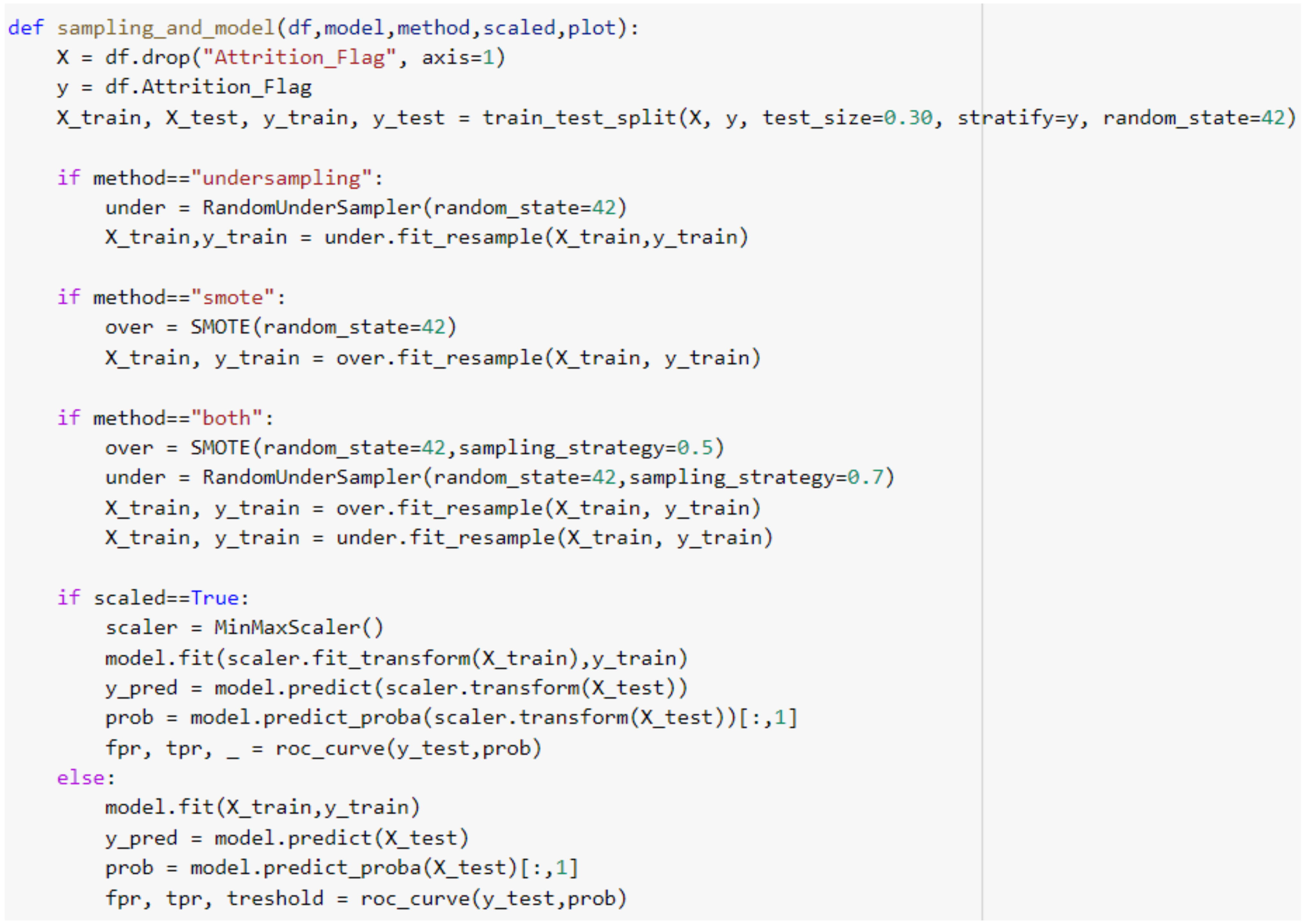
Correlation Matrix



	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
Attrition_Flag																				
Customer_Age	0.0087																			
Gender	0.035	0.011																		
Dependent_count	0.011	-0.13	-0.0081																	
Education_Level	0.028	-0.0019	0.017	0.0085																
Marital_Status	-0.016	-0.0055	-0.01	0.032	0.00025															
Income_Category	-0.0075	0.036	-0.78	0.073	-0.013	0.014														
Card_Category	0.007	-0.0056	-0.087	0.04	0.011	-0.041	0.093													
Months_on_book	0.0092	0.79	0.0038	-0.11	0.003	-0.0057	0.025	-0.0064												
Total_Relationship_Count	-0.15	-0.022	0.0045	-0.043	-0.0036	0.025	-0.012	-0.085	-0.017											
Months_Inactive_12_mon	0.15	0.058	0.011	-0.023	0.0039	-0.011	-0.019	-0.0034	0.071	0.0051										
Contacts_Count_12_mon	0.19	-0.024	-0.045	-0.056	-0.021	-0.001	0.022	-0.00012	-0.011	0.071	0.026									
Credit_Limit	-0.016	0.025	-0.48	0.081	-0.012	-0.028	0.6	0.48	0.02	-0.075	-0.016	0.017								
Total_Revolving_Bal	-0.27	0.016	-0.021	-0.0038	-0.019	0.042	0.033	0.022	0.0079	0.014	-0.04	-0.049	0.038							
Avg_Open_To_Buy	0.0078	0.023	-0.47	0.082	-0.0099	-0.032	0.6	0.47	0.019	-0.076	-0.012	0.021	1	-0.051						
Total_Amt_Chng_Q4_Q1	-0.13	-0.069	-0.035	-0.033	-0.013	0.039	0.02	-0.0058	-0.06	0.053	-0.028	-0.028	0.002	0.062	-0.0035					
Total_Trans_Amt	-0.16	-0.046	-0.035	0.023	-0.0084	-0.037	0.02	0.2	-0.038	-0.35	-0.04	-0.12	0.17	0.064	0.17	0.027				
Total_Trans_Ct	-0.36	-0.07	0.055	0.057	-0.0079	-0.077	-0.048	0.14	-0.05	-0.25	-0.048	-0.16	0.081	0.052	0.076	-0.0046	0.81			
Total_Ct_Chng_Q4_Q1	-0.28	-0.018	0.0073	0.019	-0.024	-0.0014	-0.015	-0.0065	-0.029	0.037	-0.04	-0.1	-0.006	0.084	-0.013	0.39	0.084	0.11		
Avg_Utilization_Ratio	-0.18	0.0013	0.31	-0.042	-0.0038	0.028	-0.34	-0.2	-0.01	0.069	-0.0032	-0.043	-0.48	0.63	-0.54	0.039	-0.086	-0.0021	0.072	

From the correlation matrix you can see that the variable *Avg_Open_To_Buy* is **highly correlated** with the variable *Credit_Limit* so we decided to drop it (no additional information from it).

Model Building



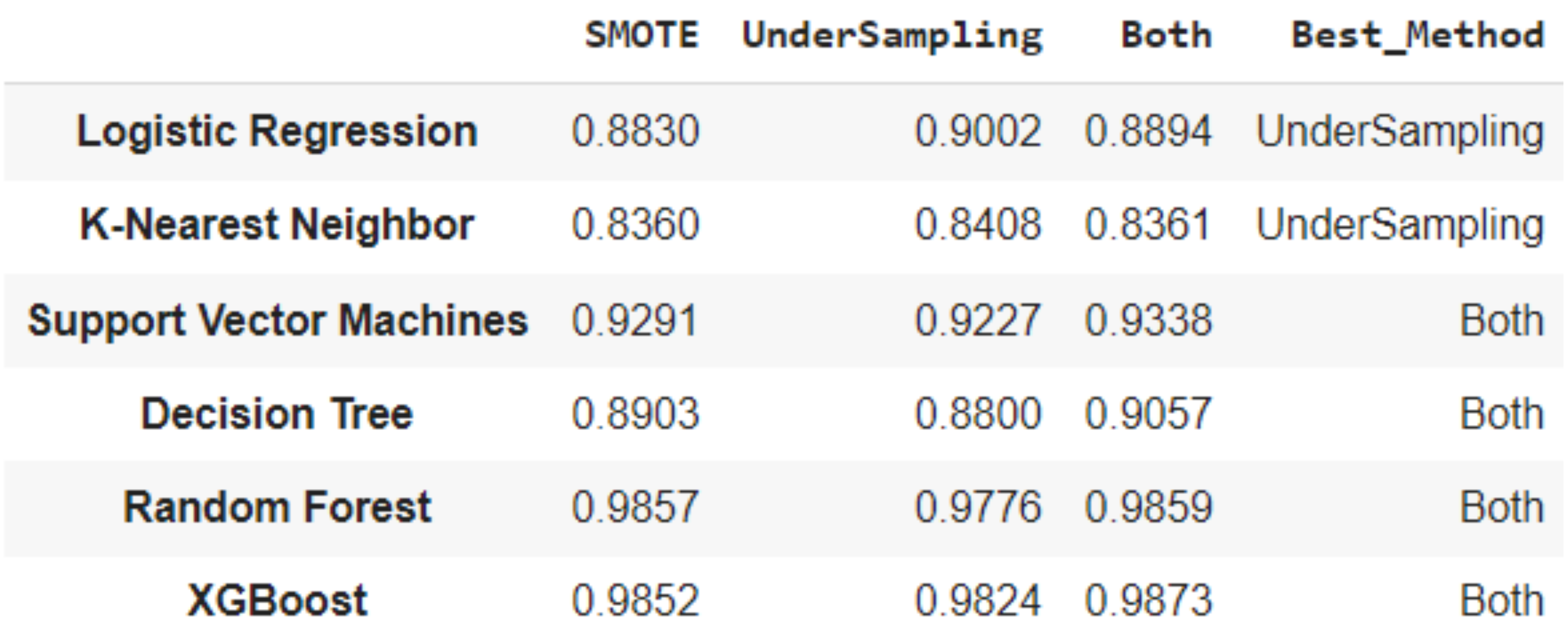
```
def sampling_and_model(df,model,method,scaled,plot):
    X = df.drop("Attrition_Flag", axis=1)
    y = df.Attrition_Flag
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, stratify=y, random_state=42)

    if method=="undersampling":
        under = RandomUnderSampler(random_state=42)
        X_train,y_train = under.fit_resample(X_train,y_train)

    if method=="smote":
        over = SMOTE(random_state=42)
        X_train, y_train = over.fit_resample(X_train, y_train)

    if method=="both":
        over = SMOTE(random_state=42,sampling_strategy=0.5)
        under = RandomUnderSampler(random_state=42,sampling_strategy=0.7)
        X_train, y_train = over.fit_resample(X_train, y_train)
        X_train, y_train = under.fit_resample(X_train, y_train)

    if scaled==True:
        scaler = MinMaxScaler()
        model.fit(scaler.fit_transform(X_train),y_train)
        y_pred = model.predict(scaler.transform(X_test))
        prob = model.predict_proba(scaler.transform(X_test))[:,1]
        fpr, tpr, _ = roc_curve(y_test,prob)
    else:
        model.fit(X_train,y_train)
        y_pred = model.predict(X_test)
        prob = model.predict_proba(X_test)[:,1]
        fpr, tpr, treshold = roc_curve(y_test,prob)
```

Our dataset was unbalanced and to solve this problem we defined a function where we tried 2 techniques:

- **Undersampling** that halves randomly observations from the majority class to reach the same number of the minority class
- **SMOTE** constructs new samples by observing the attributes of all samples and then alter the values just so that they stay within the range observed in the minority-class.

We have also tried to use the two methods together (*'both'*).

Models & Results

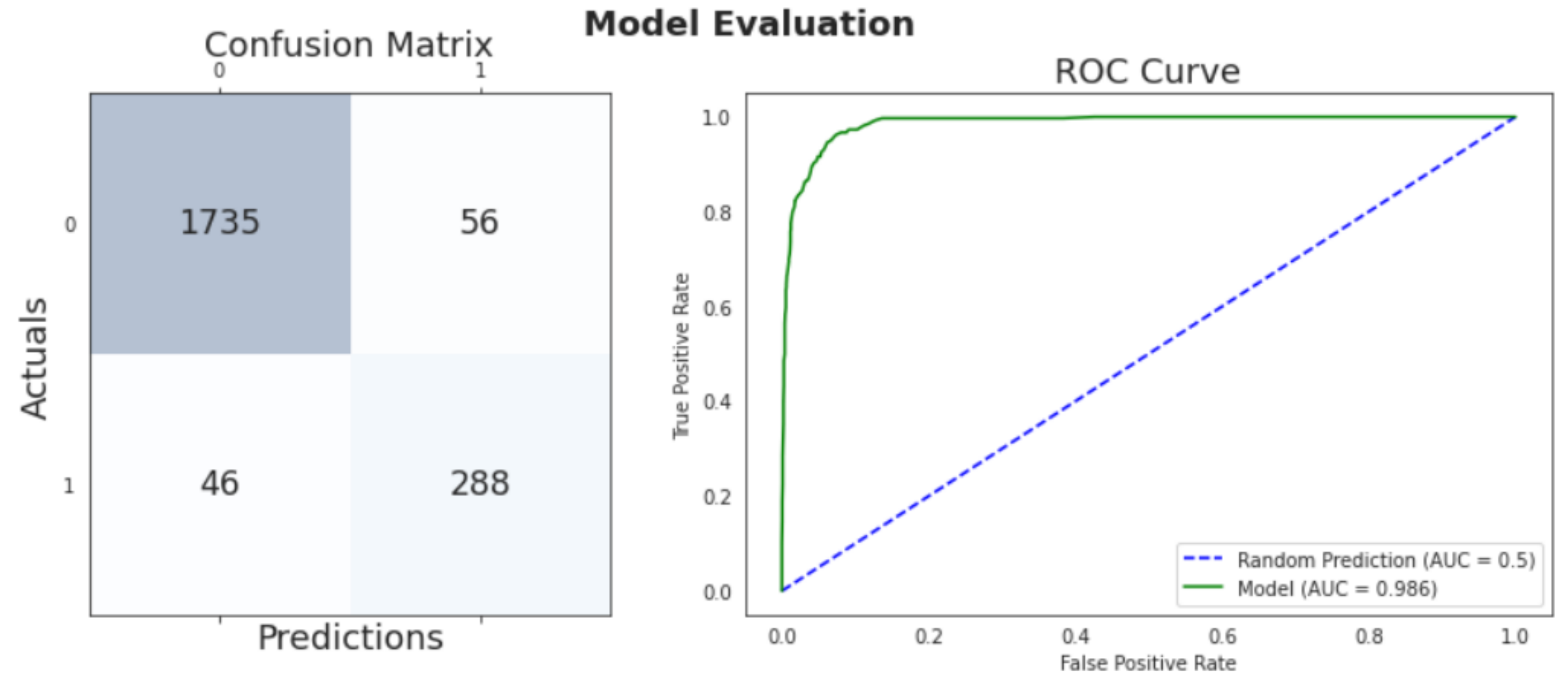


	SMOTE	UnderSampling	Both	Best_Method
Logistic Regression	0.8830	0.9002	0.8894	UnderSampling
K-Nearest Neighbor	0.8360	0.8408	0.8361	UnderSampling
Support Vector Machines	0.9291	0.9227	0.9338	Both
Decision Tree	0.8903	0.8800	0.9057	Both
Random Forest	0.9857	0.9776	0.9859	Both
XGBoost	0.9852	0.9824	0.9873	Both

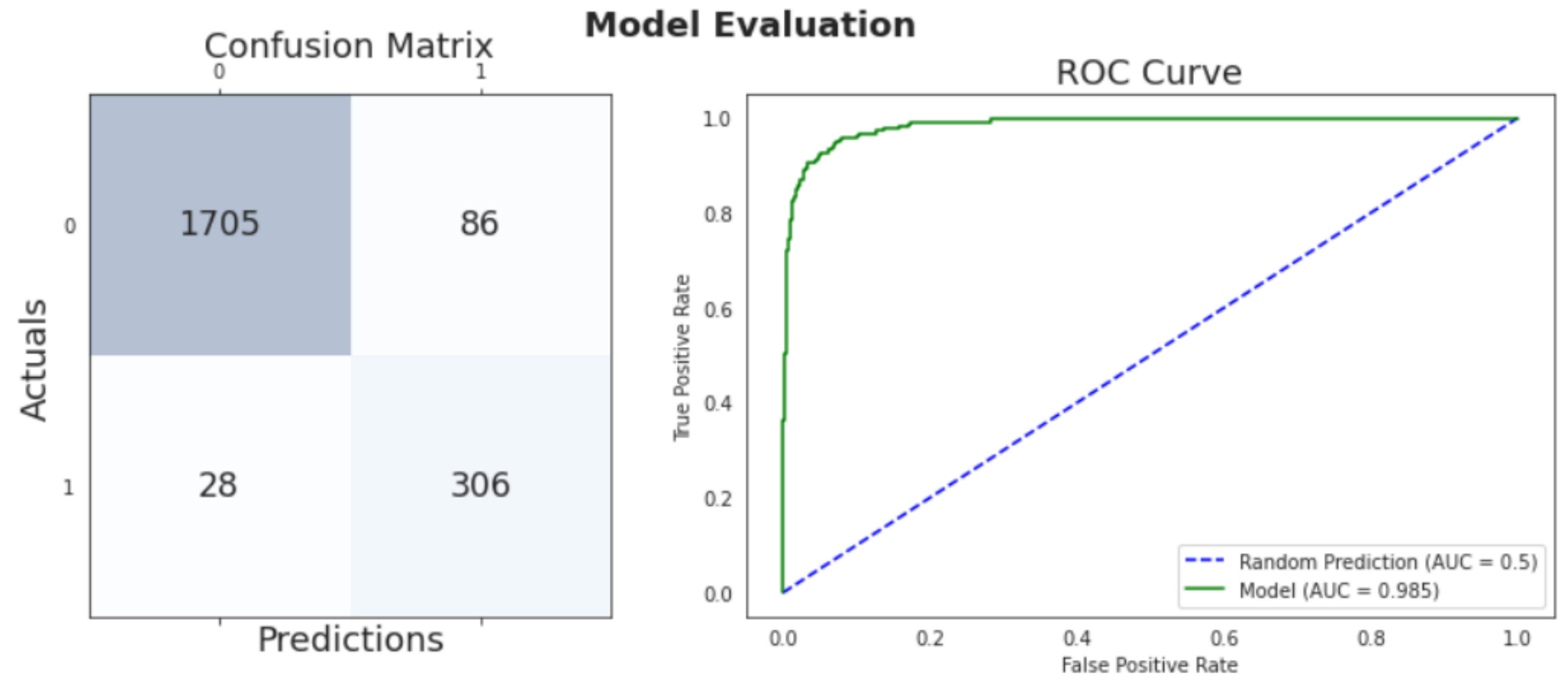
These are the models we tested with their respective results. We used the **AUC score** as metric to assess the goodness of the models, the two highest are for the **Random Forest** and **XGBoost**.

Model comparison

```
Random Forest
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1791
           1       0.84      0.86      0.85       334

    accuracy                           0.95      2125
   macro avg       0.91      0.92      0.91      2125
weighted avg       0.95      0.95      0.95      2125

AUC Score:  0.9857
```

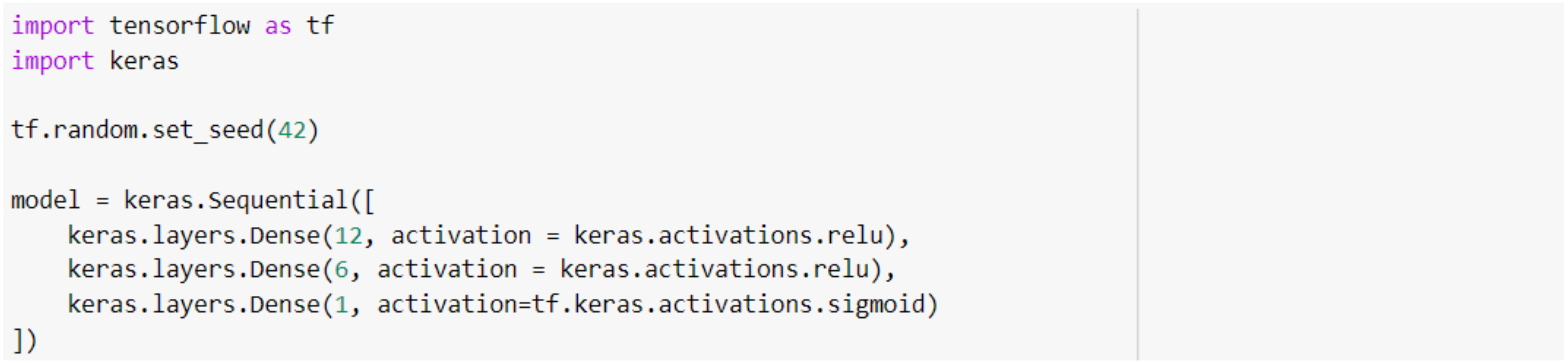


```
XGBoost
              precision    recall  f1-score   support

           0       0.98      0.95      0.97      1791
           1       0.78      0.92      0.84       334

    accuracy                           0.95      2125
   macro avg       0.88      0.93      0.91      2125
weighted avg       0.95      0.95      0.95      2125

AUC Score:  0.9852
```



To confirm the results we made a confusion matrix and a classification report. From these we can see how well the two models predict the churn for a client.

Neural Network

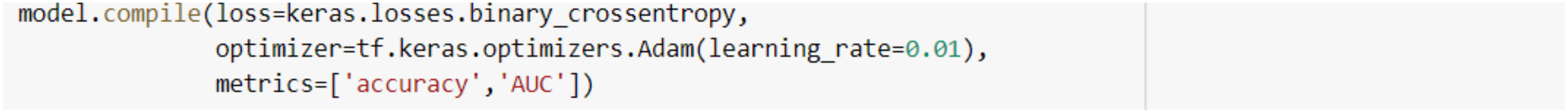


```python
import tensorflow as tf
import keras

tf.random.set_seed(42)

model = keras.Sequential([
    keras.layers.Dense(12, activation = keras.activations.relu),
    keras.layers.Dense(6, activation = keras.activations.relu),
    keras.layers.Dense(1, activation=tf.keras.activations.sigmoid)
])
```

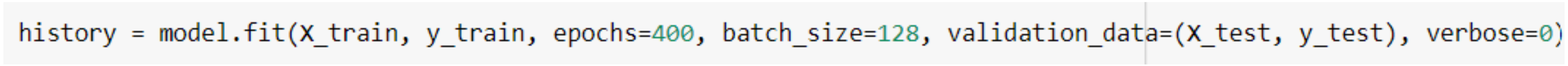
After splitting and scaling the dataset we defined a neural network by using the library Keras. We created a model with **three-layers**: the first layers of the model contains ***12 neurons*** that take the input from the data and applies the **ReLU** activation.

The second layer contains ***6 neurons*** that takes the input from the preceding layer and applies again a **ReLU**. The third layer has only ***one neuron*** that takes the input from the preceding layer, applies a **Sigmoid** activation and gives the classification output as 0 or 1.

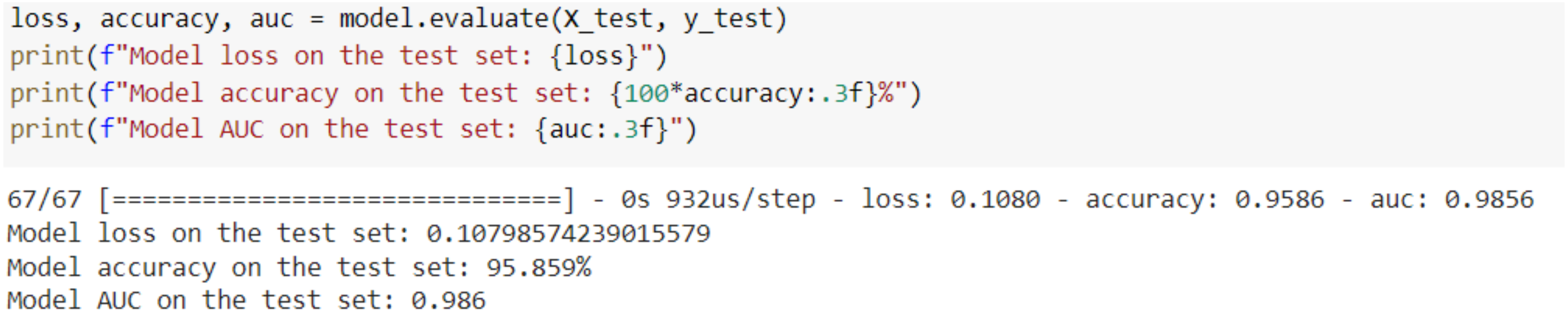
Later we compiled the model by using ***binary_crossentropy*** as loss and ***Adam*** as optimizer with a learning rate of 0.01


```
model.compile(loss=keras.losses.binary_crossentropy,
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
              metrics=['accuracy','AUC'])
```

Then we used X_train and y_train for training the model and run it for *400 epochs* with a *batch size* of *128.*

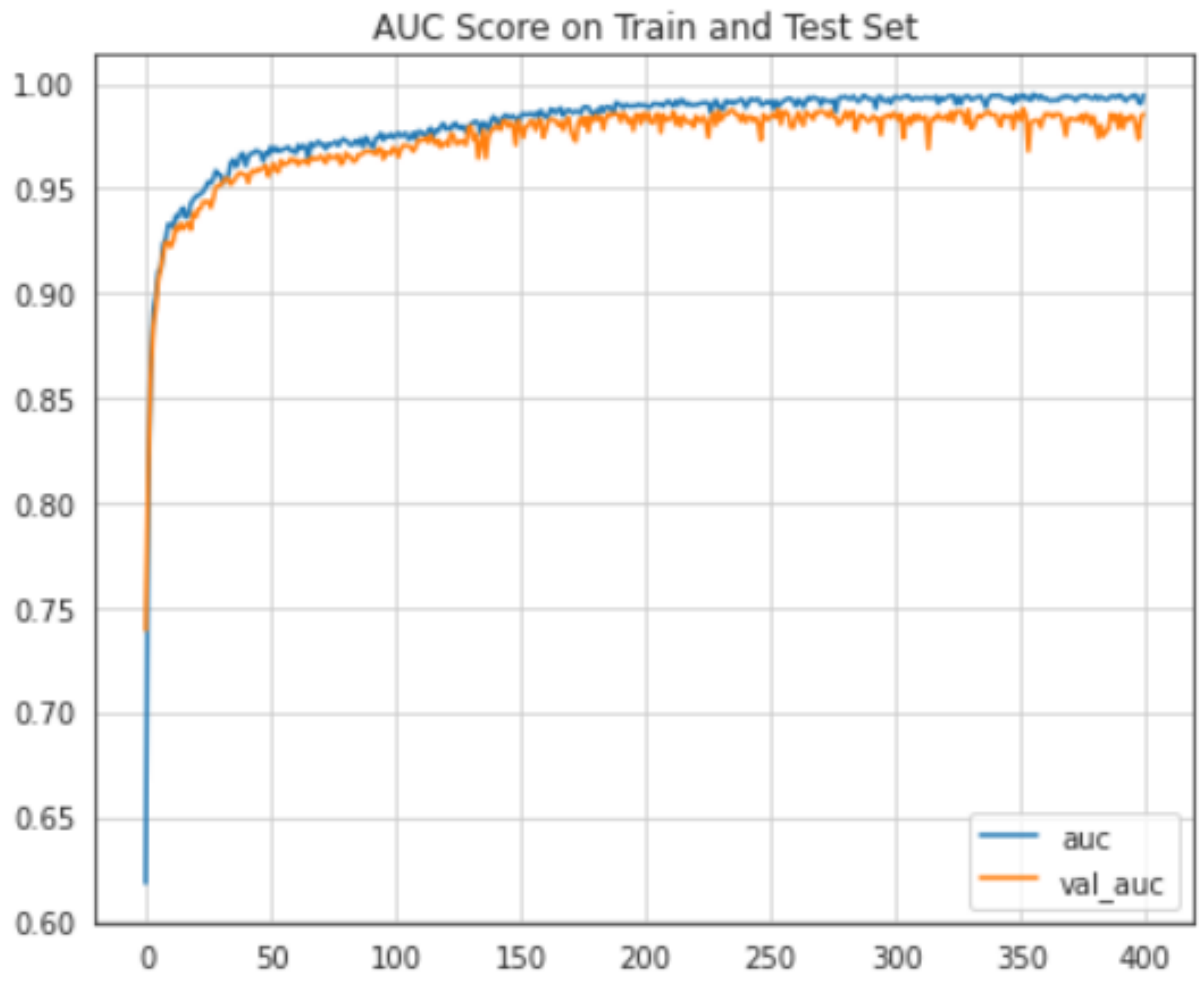

```
history = model.fit(X_train, y_train, epochs=400, batch_size=128, validation_data=(X_test, y_test), verbose=0)
```

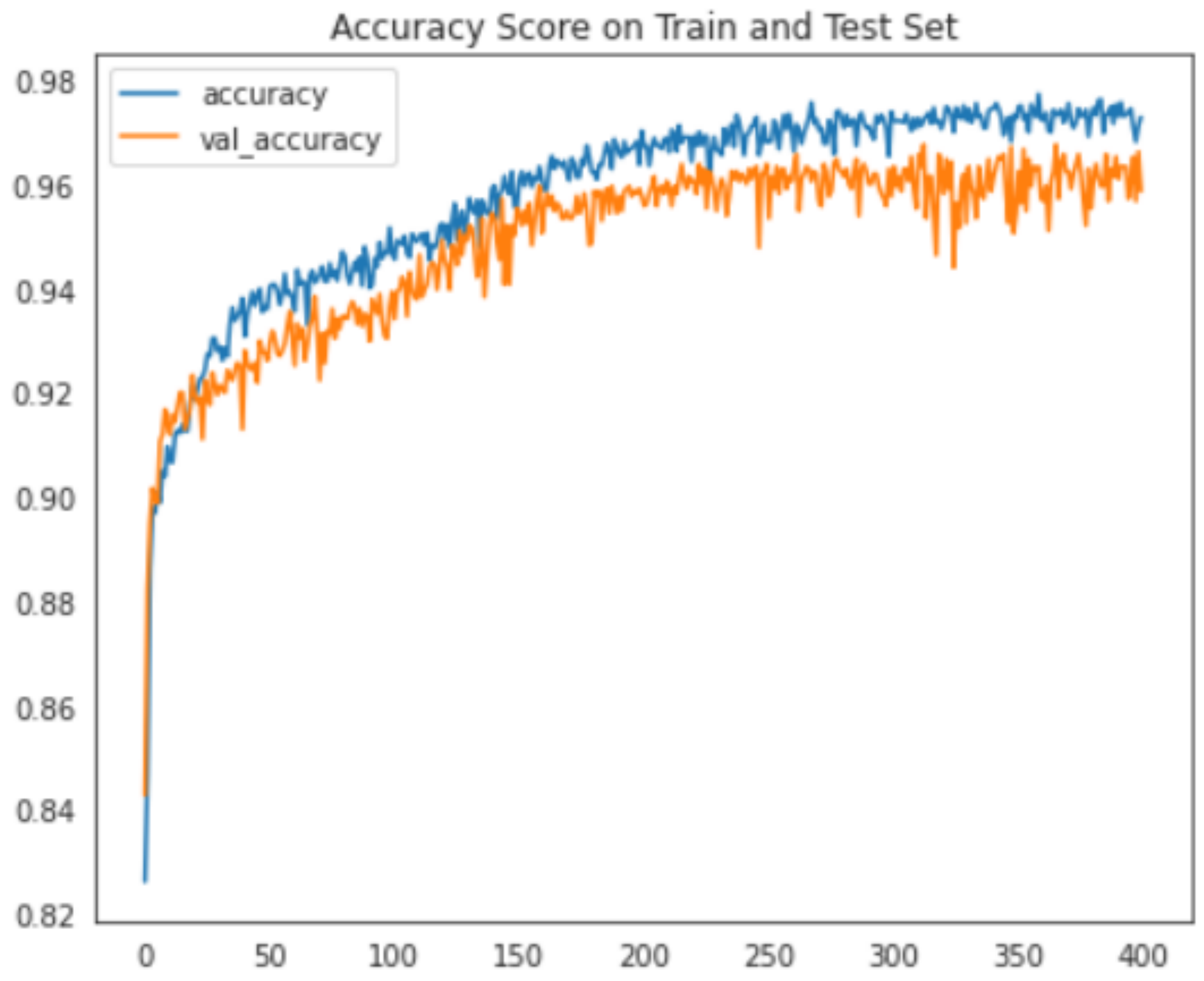
After the training, the model is evaluated on X_test and y_test.


```
loss, accuracy, auc = model.evaluate(X_test, y_test)
print(f"Model loss on the test set: {loss}")
print(f"Model accuracy on the test set: {100*accuracy:.3f}%")
print(f"Model AUC on the test set: {auc:.3f}")
```

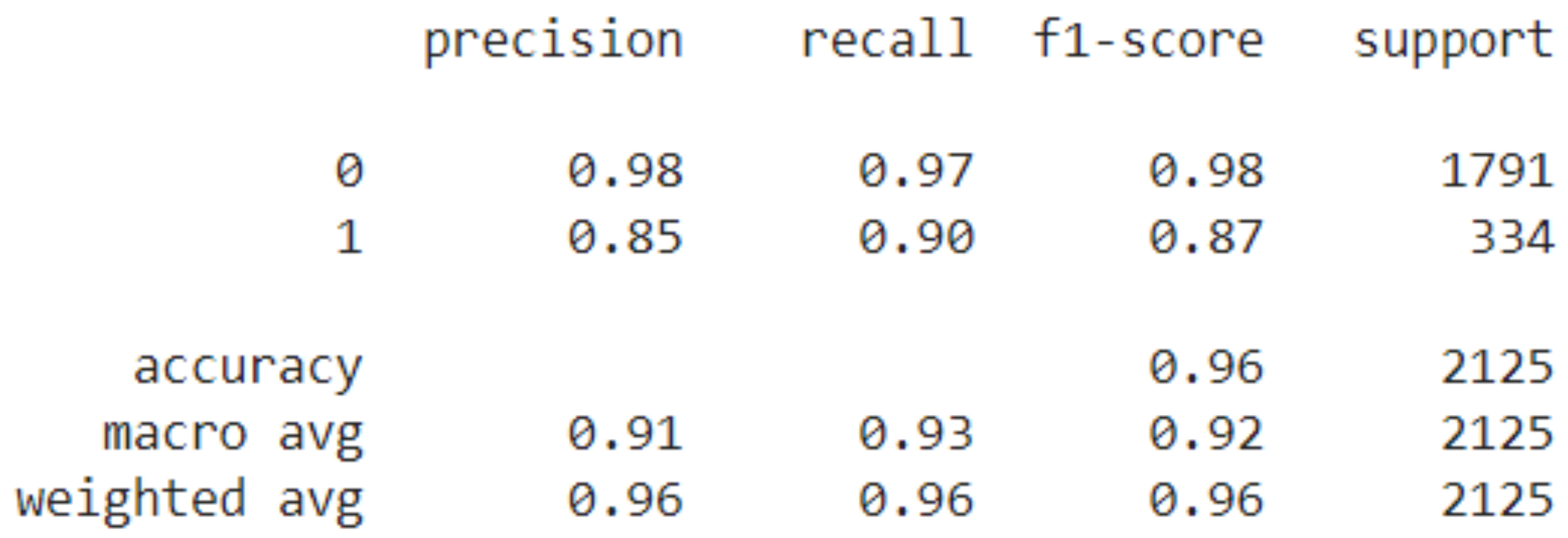
```
67/67 [==============================] - 0s 932us/step - loss: 0.1080 - accuracy: 0.9586 - auc: 0.9856
Model loss on the test set: 0.10798574239015579
Model accuracy on the test set: 95.859%
Model AUC on the test set: 0.986
```

The AUC is **0.9856** and this is a very good result, to better evaluate we also checked whether the neural network is overfitting or not thanks to the graph below but it seems to work well.

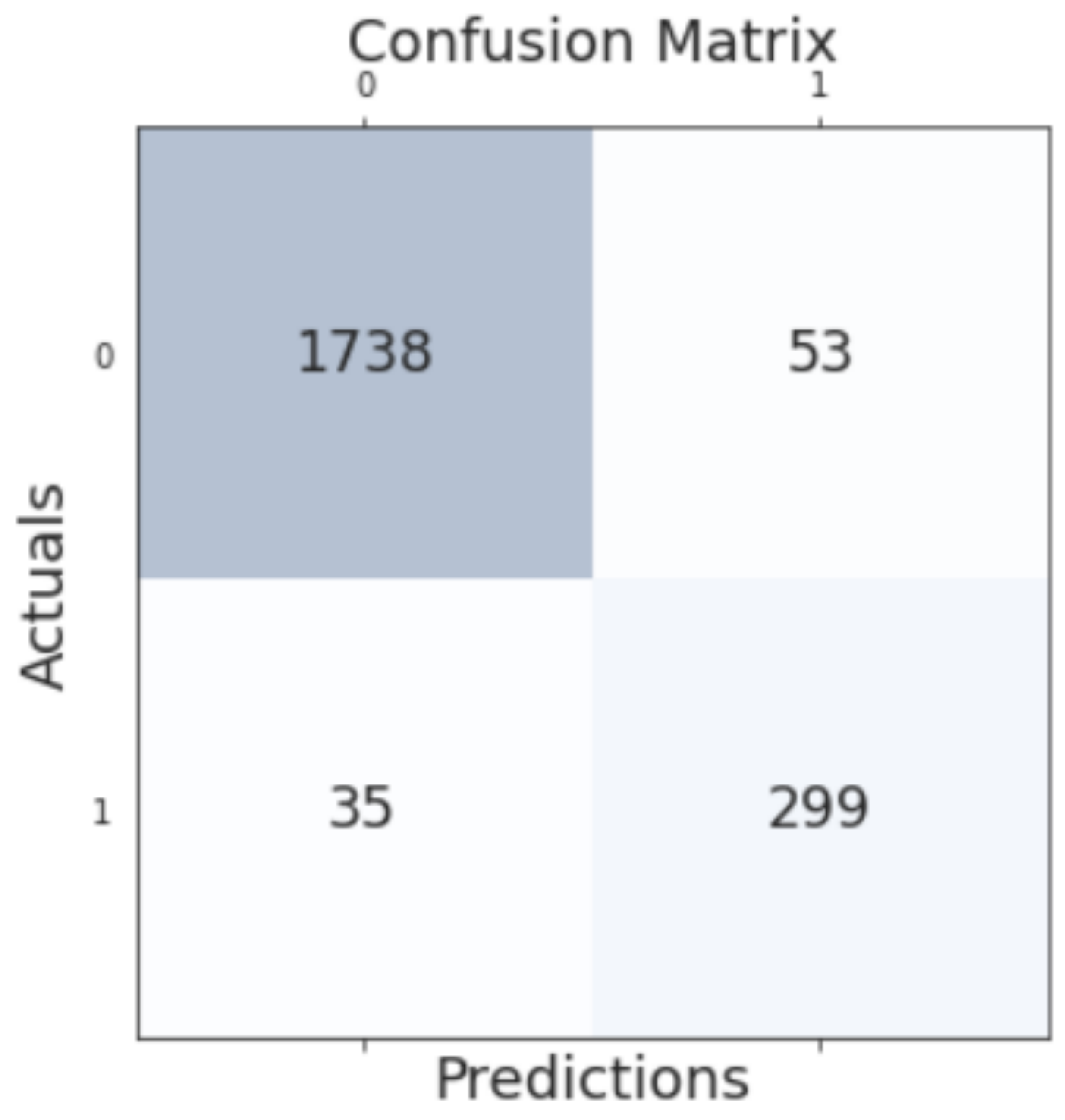




In the end we also plotted the classification report and confusion matrix to get a complete view of the neural network performance.



```
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1791
           1       0.85      0.90      0.87       334

    accuracy                           0.96      2125
   macro avg       0.91      0.93      0.92      2125
weighted avg       0.96      0.96      0.96      2125
```



Conclusions

The bank manager can now draw enough cues from the EDA to understand which types of customers are most likely to leave.
He can also try to predict whether a client will exit or not through one of the above models.