Churn Prediction

Objective:

Our objective is to build a machine learning model to predict whether the customer will churn or not in the next six months.

Importing the Relevant Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score, GridSearchCV
import warnings
warnings.filterwarnings("ignore")
```

Exploration and Feature Selection

To get initial insights and learn what story you can tell with the data, data exploration makes definitely sense. By using the python functions data.head() and “data.shape” we get a general overview of the dataset.

Reading the training dataset

```
train = pd.read_csv('C:/Users/admin/Downloads/train.csv')
```

```
train.head()
```

```
train.shape
```

The size of the train dataset was examined. It consists of 6650 observation units and 11 variables.

Reading the test dataset

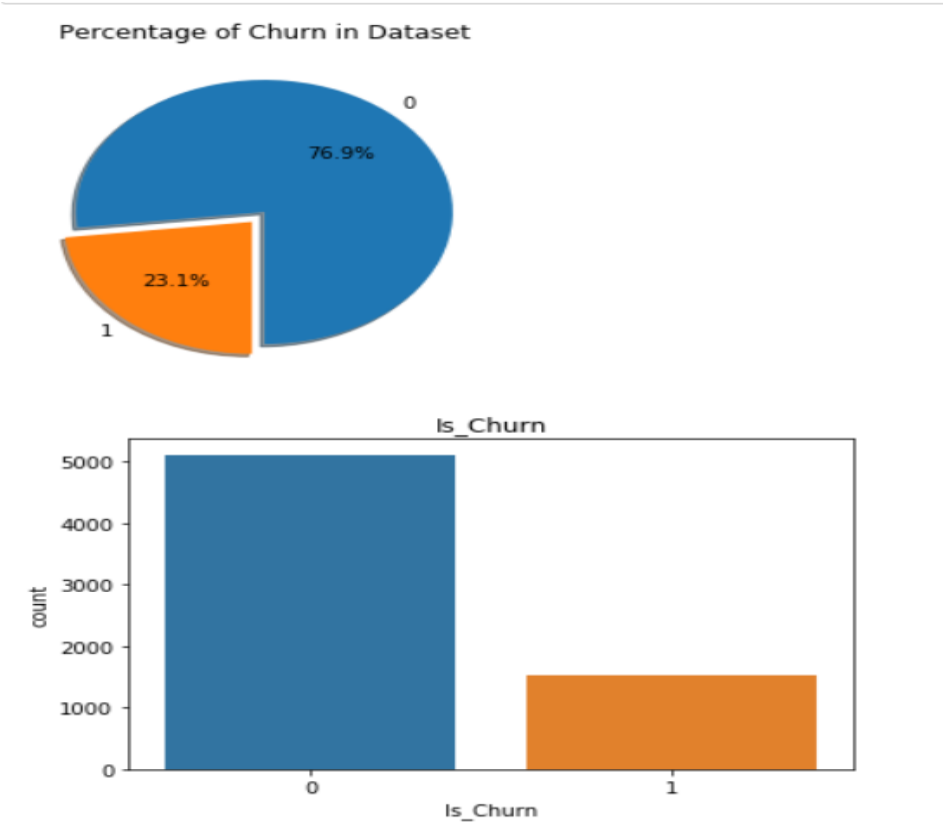
```
test = pd.read_csv('C:/Users/admin/Downloads/test.csv')
```

```
test.head()
```

```
test.shape
```

The size of the test data set was examined. It consists of 2851 observation units and 10 variables.

In detail we have a look at the target feature, the actual “Is_Churn”. Therefore we plot it accordingly and see that 23.1% Of the total amount of customer churn. This is important to know so we have the same proportion of Churned Customers to Non-Churned Customers in our training data.



```
# Data to plot

sizes = train['Is_Churn'].value_counts()

labels=0,1

explode = (0, 0.1)  # explode 1st slice

# Plot

plt.pie(sizes, explode=explode,autopct='%1.1f%%',labels=labels, shadow=True,
startangle=270,)

plt.title('Percentage of Churn in Dataset')

plt.show()

sns.countplot('Is_Churn',data=train)

plt.title('Is_Churn')

plt.show()
```

Data Preparation and Feature Engineering

Be aware that the better we prepare our data for the machine learning model, the better our prediction will be.

1.Dropping irrelevant data:

There may be data included that is not needed to improve our results. Best is that to identify by logic thinking or by creating a correlation matrix. In this data set we have the customerID for example. As it does not influence our predicted outcome, we drop the column with the pandas “drop()” function.

2.Missing Values:

Furthermore it is important to handle missing data. The values can be identified by the “.isnull()” function in pandas for example. After identifying the null values it depends on each case if it makes sense to fill the missing value for example with the mean, median or the mode, or in case there is enough training data drop the entry completely. There are no null values in our data set.To know the data types, structure and also to check the null values in the data we use the below functions.

3.Categorical data into numerical data

As we cannot calculate anything with string values, we have to convert these values intro numeric ones. A simple example in our dataset is the gender. By using the Pandas function “get_dummies()” two columns will replace the

gender column with “gender_Female” and “gender_Male”. Additionally to that we could use the “get_dummies()” function for all the categorical variables in the dataset.

4.Splitting the dataset

First our model needs to be trained, second our model needs to be tested. Therefore it is best to have two different dataset. As for now we only have one, it is very common to split the data accordingly. X is the data with the independent variables, Y is the data with the dependent variable. The test size variable determines in which ratio the data will be split. It is quite common to do this in a 80 Training / 20 Test ratio.

```
train.describe

train.dtypes

train.isnull().any()

#Feature Engineering

train["AgeScore"] = pd.qcut(train['Age'], 8, labels = [1, 2, 3, 4, 5, 6, 7, 8])

train["BalanceScore"] = pd.qcut(train['Balance'].rank(method="first"), 10, labels = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

test["AgeScore"] = pd.qcut(test['Age'], 8, labels = [1, 2, 3, 4, 5, 6, 7, 8])

test["BalanceScore"] = pd.qcut(test['Balance'].rank(method="first"), 10, labels = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

train.head()

#One Hot Encoding

train = pd.get_dummies(train, columns =["Gender"], drop_first = True)

train.head()

test = pd.get_dummies(test, columns =["Gender"], drop_first = True)

#Scaling

train = train.drop(["ID","Age","Income","Balance","Credit_Category","Product_Holdings"], axis = 1)

test = test.drop(["ID","Age","Income","Balance","Credit_Category","Product_Holdings"], axis = 1)

train.head()
```

Model Building

Here, we used some of the models like Logistic Regression, KNeighbors Classifier, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine , Gradient Boosting Classifier and noted the response.

Logistic Regression is one of the most used machine learning algorithm and mainly used when the dependent variable (here churn 1 or churn 0) is categorical. The independent variables in contrary can be categorical or numerical. With the trained model we can now predict if a customer churned or not for our test dataset. The results are saved in “y_pred” and afterwards the accuracy score and F1 score is measured. Logistic Regression performed well over the other models with Accuracy of 78.72%

```
LR Accuracy Score: (78.721805)
LR F1_Score: (0.027491)
KNN Accuracy Score: (74.887218)
KNN F1_Score: (0.204762)
CART Accuracy Score: (71.654135)
CART F1_Score: (0.238384)
RF Accuracy Score: (71.353383)
RF F1_Score: (0.274286)
SVM Accuracy Score: (78.796992)
SVM F1_Score: (0.034247)
XGB Accuracy Score: (78.270677)
XGB F1_Score: (0.099688)
```

```
#Modelling
models = []
models.append(('LR', LogisticRegression(random_state = 5160)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state = 5160)))
models.append(('RF', RandomForestClassifier(random_state = 5160)))
models.append(('SVM', SVC(gamma='auto', random_state = 5160)))
models.append(('XGB', GradientBoostingClassifier(random_state = 5160)))
# evaluate each model in turn
results = []
names = []
```

```
#Accuracy Score and F1_Score
for name, model in models:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)*100
        f1score=f1_score(y_test, y_pred)
        msg = "%s Accuracy Score: (%f)" % (name, accuracy)
        msg1 = "%s F1_Score: (%f)" % (name, f1score)
        print(msg)
        print(msg1)

submission = pd.read_csv('C:/Users/admin/Downloads/sample.csv')
final_predictions = model.predict(test)
submission['Is_Churn'] = final_predictions
#only positive predictions for the target variable
submission['Is_Churn'] = submission['Is_Churn'].apply(lambda x: 1 if x>0 else x)
submission.to_csv('my_submission.csv', index=False)
```