Allianz

March 2, 2023

1 Libraries used for the last part

[3]:
```
import xgboost as xgb
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
```

1.1 Definition of the data to be trained

[4]:
```
# Definition of X
X = pd.read_csv("Datathon_Allianz-main/Datathon/allianz.csv")

# Definition of the main treated in the first ipynb
main = pd.read_csv("Datathon_Allianz-main/Datathon/main_1.csv")

# Definition of the popensity to pay between 0 and 1
main['BOOLEAN_PROP_TO_PAY'] = main['PROPENSITY_TO_PAY'].astype(float) > 0.95
```

We defined that the *propensity_to_pay* recommended by this model, which can be definetely be improved, waas 0.95, taking into account that it is above the 95 percentile.

1.2 Description PROPENSITY_TO_PAY

[5]:
```
main['PROPENSITY_TO_PAY'].astype(float).describe()
```

[5]:
```
count    33110.000000
mean         0.887577
std          0.071992
min          0.000000
25%          0.833333
50%          0.868056
75%          0.953704
max          1.000000
Name: PROPENSITY_TO_PAY, dtype: float64
```

1.3 Description BOOLEAN_PROP_TO_PAY

[6]:
```
main['PROPENSITY_TO_PAY'].astype(float).describe()
```

[6]:
```
count    33110.000000
mean         0.887577
std          0.071992
min          0.000000
25%          0.833333
50%          0.868056
75%          0.953704
max          1.000000
Name: PROPENSITY_TO_PAY, dtype: float64
```

[7]:
```
# SEPARATE DATA WITH PERCENTAGE
X_train, X_test, y_train, y_test= train_test_split(X.astype(float),main.
 ↪BOOLEAN_PROP_TO_PAY.astype(float),test_size=0.8,random_state=1)
```

[8]:
```
# import algorithm
xgb_clf= xgb.XGBClassifier()


#fitting the model
le = LabelEncoder()
y_train = le.fit_transform(y_train)
# y_train

xgb_clf.fit(X_train, y_train)


## Predicting the model
xgb_predict= xgb_clf.predict(X_test)
xgb_predict
```

[8]:
```
array([0, 0, 1, …, 0, 0, 0])
```

[9]:
```
test = le.inverse_transform(xgb_predict)
```

[10]:
```
print(f'Model 1 XGboost Report for ALLIANZ \n␣
 ↪{classification_report(xgb_predict, y_test)}')
```

```
Model 1 XGboost Report for ALLIANZ
              precision    recall  f1-score   support

           0       0.91      0.76      0.83     23401
           1       0.19      0.44      0.26      3087

    accuracy                           0.72     26488
```

macro avg	0.55	0.60	0.55	26488
weighted avg	0.83	0.72	0.76	26488