XGBoost

```
print('XGBoost model - SMOTE RFM')

xgb_model = xgb.XGBClassifier(objective='binary:logistic', eval_metric='auc',
 learning_rate =0.3,    #change 0.01 >> 0.3
 n_estimators=100,
 max_depth=20,        #change 2 >> 20
 gamma=0.0,
 colsample_bytree=0.6)

predicted_y = []
expected_y = []

xgb_model_SMOTE_rfm = xgb_model.fit(X_SMOTE_rfm, y_SMOTE_rfm, early_stopping_rounds=5, eval_set=[(X_test_rfm.to_numpy(), y_test_rfm)])
predictions =  xgb_model_SMOTE_rfm.predict(X_SMOTE_rfm)
predicted_y.extend(predictions)
expected_y.extend(y_SMOTE_rfm)
report_train = classification_report(expected_y, predicted_y)
print('training set')
print(report_train)

predicted_y = []
expected_y = []
predictions = xgb_model_SMOTE_rfm.predict(X_test_rfm.to_numpy())
predicted_y.extend(predictions)
expected_y.extend(y_test_rfm)
report_test = classification_report(expected_y, predicted_y)
print('test set')
print(report_test)
```

XGBoost model - SMOTE RFM

```
[0]   validation_0-auc:0.562636
Will train until validation_0-auc hasn't improved in 5 rounds.
[1]   validation_0-auc:0.636721
[2]   validation_0-auc:0.639358
[3]   validation_0-auc:0.644874
[4]   validation_0-auc:0.663385
[5]   validation_0-auc:0.667483
[6]   validation_0-auc:0.662065
[7]   validation_0-auc:0.674141
[8]   validation_0-auc:0.681083
[9]   validation_0-auc:0.681651
[10]  validation_0-auc:0.68425
[11]  validation_0-auc:0.685515
[12]  validation_0-auc:0.687809
[13]  validation_0-auc:0.692853
[14]  validation_0-auc:0.697338
[15]  validation_0-auc:0.701752
[16]  validation_0-auc:0.705411
[17]  validation_0-auc:0.708047
[18]  validation_0-auc:0.710109
[19]  validation_0-auc:0.711633
[20]  validation_0-auc:0.711728
[21]  validation_0-auc:0.712254
[22]  validation_0-auc:0.708893
[23]  validation_0-auc:0.708511
[24]  validation_0-auc:0.709084
[25]  validation_0-auc:0.709794
[26]  validation_0-auc:0.710745
Stopping. Best iteration:
[21]  validation_0-auc:0.712254
```

training set

	precision	recall	f1-score	support
0	0.92	0.99	0.95	4389
1	0.98	0.92	0.95	4389
accuracy			0.95	8778
macro avg	0.95	0.95	0.95	8778
weighted avg	0.95	0.95	0.95	8778

test set

	precision	recall	f1-score	support
0	0.90	0.96	0.93	1848
1	0.23	0.10	0.14	218
accuracy			0.87	2066
macro avg	0.56	0.53	0.53	2066
weighted avg	0.82	0.87	0.85	2066

```
print('XGBoost model - SMOTE CLV')

xgb_model = xgb.XGBClassifier(objective='binary:logistic', eval_metric='auc',
 learning_rate =0.3,        #change 0.01 >> 0.3
 n_estimators=100,
 max_depth=20,              #change 2 >> 20
 gamma=0.0,
 colsample_bytree=0.6)

predicted_y = []
expected_y = []

xgb_model_SMOTE_clv = xgb_model.fit(X_SMOTE_clv, y_SMOTE_clv, early_stopping_rounds=5, eval_set=[(X_test_clv.to_numpy(), y_test_clv)])
predictions =  xgb_model_SMOTE_clv.predict(X_SMOTE_clv)
predicted_y.extend(predictions)
expected_y.extend(y_SMOTE_clv)
report_train = classification_report(expected_y, predicted_y)
print('training set')
print(report_train)

predicted_y = []
expected_y = []
predictions = xgb_model_SMOTE_clv.predict(X_test_clv.to_numpy())
predicted_y.extend(predictions)
expected_y.extend(y_test_clv)
report_test = classification_report(expected_y, predicted_y)
print('test set')
print(report_test)
```

XGBoost model - SMOTE CLV

```
[0]   validation_0-auc:0.616022
Will train until validation_0-auc hasn't improved in 5 rounds.
[1]   validation_0-auc:0.623804
[2]   validation_0-auc:0.627667
[3]   validation_0-auc:0.63077
[4]   validation_0-auc:0.634964
[5]   validation_0-auc:0.633452
[6]   validation_0-auc:0.643062
[7]   validation_0-auc:0.648257
[8]   validation_0-auc:0.652092
[9]   validation_0-auc:0.654402
[10]  validation_0-auc:0.655087
[11]  validation_0-auc:0.654912
[12]  validation_0-auc:0.652753
[13]  validation_0-auc:0.650833
[14]  validation_0-auc:0.651853
[15]  validation_0-auc:0.654077
Stopping. Best iteration:
[10]  validation_0-auc:0.655087
```

training set

	precision	recall	f1-score	support
0	0.98	0.99	0.99	4389
1	0.99	0.98	0.99	4389
accuracy			0.99	8778
macro avg	0.99	0.99	0.99	8778
weighted avg	0.99	0.99	0.99	8778

test set

	precision	recall	f1-score	support
0	0.90	0.92	0.91	1848
1	0.19	0.16	0.17	218
accuracy			0.84	2066
macro avg	0.55	0.54	0.54	2066
weighted avg	0.83	0.84	0.83	2066

```
## hyper parameter tuning - grid search

from sklearn.model_selection import KFold, GridSearchCV
from sklearn.metrics import accuracy_score, make_scorer
# Define our search space for grid search
search_space = [
  {
    'clf__n_estimators': [100, 300],
    'clf__learning_rate': [0.01, 0.1],
    'clf__max_depth': range(2, 5),
    'clf__colsample_bytree': [i/10.0 for i in range(4, 8)],     #change range (4,7) >> (4,8)
    'clf__gamma': [i/10.0 for i in range(3)],
    'fs__score_func': [chi2],
    'fs__k': [2],
  }
]
# Define cross validation
kfold = KFold(n_splits=5, random_state=60)      #Random state from 42 >> 60
# AUC and accuracy as score
scoring = {'AUC':'roc_auc', 'Accuracy':make_scorer(accuracy_score), 'F1 score': 'f1_micro'}
# Define grid search
grid = GridSearchCV(
  pipe,
  param_grid=search_space,
  cv=kfold,
  scoring=scoring,
  refit='AUC',
  verbose=1,
  n_jobs=-1
)

# Fit grid search
xgb_model_clv_GS = grid.fit(X_train_clv, y_train_clv)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:296: FutureWarning:
  FutureWarning
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
Fitting 5 folds for each of 144 candidates, totalling 720 fits
[Parallel(n_jobs=-1)]: Done  88 tasks      | elapsed:   15.5s
[Parallel(n_jobs=-1)]: Done 388 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 720 out of 720 | elapsed:  2.1min finished
```

[95]
```
predicted_y = []
expected_y = []
predictions = xgb_model_clv_GS.predict(X_test_clv)
print('Best AUC Score: {}'.format(xgb_model_clv_GS.best_score_))
print('Accuracy: {}'.format(accuracy_score(y_test_clv, predictions)))
print(confusion_matrix(y_test_clv,predictions))


predicted_y.extend(predictions)
expected_y.extend(y_test_clv)
report_test = classification_report(expected_y, predicted_y)
print('test set')
print(report_test)
```

```
Best AUC Score: 0.7089054675375633
Accuracy: 0.6079380445304937
[[1089  759]
 [  51  167]]
test set
              precision    recall  f1-score   support

           0       0.96      0.59      0.73      1848
           1       0.18      0.77      0.29       218

    accuracy                           0.61      2066
   macro avg       0.57      0.68      0.51      2066
weighted avg       0.87      0.61      0.68      2066
```