```
In [ ]: # import libraries
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        # import scalers
        from sklearn.preprocessing import StandardScaler
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.preprocessing import RobustScaler
        from sklearn.preprocessing import Normalizer
        # import feature selection
        from sklearn.feature selection import VarianceThreshold
        from sklearn.feature_selection import chi2
        from sklearn.feature selection import f classif
        from sklearn.feature_selection import mutual_info_classif
        from sklearn.feature selection import mutual info regression
        from sklearn.feature_selection import SelectKBest
        from sklearn.feature selection import SelectPercentile
        from sklearn.feature_selection import SelectFromModel
        from sklearn.feature_selection import RFE
        # import classificators
        from sklearn.linear model import LogisticRegression
        from sklearn.svm import SVC
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        # import regressors
        from sklearn.linear model import LinearRegression
        from sklearn.svm import SVR
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import RandomForestRegressor
        # import metrics
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import cross_val_score
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import mean_squared_error
        from sklearn.metrics import mean_absolute_error
        from sklearn.metrics import r2_score
        from sklearn.metrics import classification_report
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import roc_curve
        from sklearn.metrics import confusion_matrix
        # import library for unalanced data
        # Synthetic Minority Oversampling Technique (SMOTE)
        from imblearn.over_sampling import SMOTE
        # K-Nearest Neighbor OveRsampling (KNNOR)
        from knnor import data_augment
        # SMOTE + Tomek
        from imblearn.combine import SMOTETomek
        # SMOTE + ENN
        from imblearn.combine import SMOTEENN
        # random over sampler
```

```
from imblearn.over_sampling import RandomOverSampler

# import system
import os
import sys
```

#### Deal with unbalanced data

https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/

K-Nearest Neighbor OveRsampling approach: https://www.sciencedirect.com/science/article/pii/S156849462

### **Online learning:**

https://www.sciencedirect.com/topics/physics-and-astronomy/weight-vector

```
In [ ]: models classific = {
            'LogisticRegression': LogisticRegression(),
            'SVC': SVC(),
            'DecisionTreeClassifier': DecisionTreeClassifier(),
            'RandomForestClassifier': RandomForestClassifier(),
        }
In [ ]: def get_data(PATH,file_name):
            try:
                df = pd.read excel(PATH+file name, header=1)
                # fill empty space with _
                csv_filename = file_name.replace(" "," ")
                df.to_csv(PATH+csv_filename[:-5]+".csv", index=False)
                df = pd.read_csv(PATH+csv_filename[:-5]+".csv")
                return df
            except:
                print("Error: file not found")
                sys.exit(1)
In [ ]: df = get_data("../data/","default of credit card clients.xls")
        df.head()
```

Out[]:

#### ID LIMIT\_BAL SEX EDUCATION MARRIAGE AGE PAY\_0 PAY\_2 PAY\_3 PAY

0	1	20000	2	2	1	24	2	2	-1
1	2	120000	2	2	2	26	-1	2	0
2	3	90000	2	2	2	34	0	0	0
3	4	50000	2	2	1	37	0	0	0
4	5	50000	1	2	1	57	-1	0	-1

5 rows × 25 columns

```
In []: # make function looks nicer
        def data_summary(df, interactive=False):
            Prints a summary of the given DataFrame.
            Parameters:
            df (pd.DataFrame): The DataFrame to summarize.
            interactive (bool): If True, pauses after each summary part and clear
            Returns:
            dict: A dictionary containing various summary information of the Data
            hashtable = {
                "Data shape": df.shape,
                "Data columns": df.columns.to_list(),
                "Data types": df.dtypes.to_dict(),
                "Data describe": df.describe().to_string(),
                "Data null count": df.isnull().sum().to_dict(),
                "Data Count": df.count().to_dict()
            }
            for key, value in hashtable.items():
                print(f"{key}:\n{value}\n")
                if interactive:
                    input("Press Enter to continue...")
                    os.system('cls' if os.name == 'nt' else 'clear')
            return None
```

```
In []: # check if the data is clean enough
    def check_data(df):
        return df.isnull().sum()

# check if the the range of each column, not include the first row
    def check_range(df):
        for col in df.columns:
            print(col, df[col].unique())
```

```
In []: # check the correlation between each column
    corr = df.corr()
    plt.figure(figsize=(14, 14))
```

```
plt.title('Correlation Matrix')
sns.heatmap(corr, annot=True, vmin=-1, vmax=1, cmap='coolwarm')

Out[]: <AxesSubplot:title={'center':'Correlation Matrix'}>
```

```
Correlation Matrix
                                                                                                                                                                                                                                                    1.00
                                      ID - 1 0.0260.0180.0390.0290.0190.03±0.01±0.01±0.00±0.0220.020.0190.0180.0240.040.0170.0170.009700846.0386.0386.00780006550030.014
                                                         0.0250.22-0.11 0.14-0.27-0.3-0.29-0.27-0.25-0.24<mark>0.29 0.28 0.28 0.29 0.3 0.29</mark> 0.2 0.18 0.21 0.2 0.22 0.22 <mark>-</mark>0.15
                         LIMIT BAL -0.026
                                   SEX -0.0180.025 1 0.0140.0340.0940.0580.0740.0660.060.0550.0440.0340.0340.0250.0250.0250.0270.0470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.00470.004
                                                                                                                                                                                                                                                   - 0.75
                        EDUCATION 40.039-0.220.014 1 -0.14 0.18 0.11 0.12 0.11 0.110.0980.0820.0240.0190.0180.00065007660090.0370.03-0.040.0380.040.0370.028
                         0.0390.050.0530.050.0540.04<mark>90.0560.0540.0540.0510.0490.04</mark>80.0260.0220.0290.0210.0230.0190.014
                                  AGE -0.0190.14-0.0910.18-0.41 1
                                                                                                                                                                                                                                                   - 0.50
                                PAY_0 -0.0310.270.0580.11 0.02-0.039 1 0.67 0.57 0.54 0.51 0.47 0.19 0.19 0.18 0.18 0.18 0.18 0.0790.070.0710.0640.0580.0590.32
                                                                                                    0.77 0.66 0.62 0.58 0.23 0.24 0.22 0.22 0.22 0.22 0.08 0.05 0.05 0.04 0.03 0.03 70.03 70.26
                                PAY_3 -0.0180.290.0660.110.0330.0530.57 0.77 1 0.78 0.69 0.63 0.21 0.24 0.23 0.23 0.23 0.23 0.20.00180.0670.0530.0460.0360.0360.024
                                PAY_4-9.00270.27-0.06 0.110.033-0.05 0.54 0.66 0.78 1 0.82 0.72 0.2 0.23 0.24 0.25 0.24 0.240.009040019.0690.0430.0340.0270.22
                                                                                                                                                                                                                                                   - 0.25
                                PAY_5 -0.0220.250.05$0.0980.0360.0540.51 0.62 0.69 0.82 1 0.82 0.21 0.23 0.24 0.27 0.27 0.260.006010030200940.0580.0330.023 0.2
                                PAY_6 -0.02-0.240.0440.0820.0340.0490.47 0.58 0.63 0.72 0.82 1 0.21 0.23 0.24 0.27 0.29 0.290.003500520058.0190.0460.0250.19
                         BILL_AMT1 -0.0190.29-0.0340.0240.0230.0560.19 0.23 0.21 0.2 0.21 0.21 1 0.95 0.89 0.86 0.83 0.8 0.140.0990.16 0.16 0.17 0.18 -0.02
                         BILL_AMT2 -0.0180.28-0.03D.0190.0220.0540.19 0.24 0.24 0.23 0.23 0.23 0.95 1 0.93 0.89 0.86 0.83 0.28 0.1 0.15 0.15 0.16 0.17-0.014
                         BILL_AMT3 -0.0240.28-0.0250.0130.0250.0540.18 0.22 0.23 0.24 0.24 0.24 0.29 0.89 0.93 1 0.92 0.88 0.85 0.24 0.32 0.13 0.14 0.18 0.18-0.014
                         BILL AMT4 - 0.04 0.29 0.022000450230.0510.18 0.22 0.23 0.25 0.27 0.27 0.86 0.89 0.92 1 0.94 0.9 0.23 0.21 0.3 0.13 0.16 0.18 0.01
                                                                                                                                                                                                                                                   - -0.25
                         BILL AMT5 -0.017 0.3 -0.017.0076.025.0490.18 0.22 0.23 0.24 0.27 0.29 0.83 0.86 0.88 0.94 1 0.95 0.22 0.18 0.25 0.29 0.14 0.160.0068
                         BILL_AMT6 -0.0170.290.010.0090.0210.0480.18 0.22 0.22 0.24 0.26 0.29 0.8 0.83 0.85 0.9 0.95 1 0.2 0.17 0.23 0.25 0.31 0.120.0054
                          PAY_AMT1 9.00970.20.0002840330.0060.0260.0790.080.00193.009040061.00150.14 0.28 0.24 0.23 0.22 0.2 1 0.29 0.25 0.2 0.15 0.19-0.073
                                                                                                                                                                                                                                                   - -0 50
                          PAY_AMT2 6,00840.180.00140.030.0080.0220.070.0580.0670.00120003220050.099 0.1 0.32 0.21 0.18 0.17 0.29 1 0.24 0.18 0.18 0.16 0.059
                          PAY_AMT3 -0.0390.210.00860.040.0036.0290.0730.0560.0530.0690.009100580.16 0.15 0.13 0.3 0.25 0.23 0.25 0.24 1 0.22 0.16 0.16-0.056
                          PAY_AMT4 9.00780.2-0.0020.0380.0130.0210.0640.0440.0440.0430.0580.0190.16 0.15 0.14 0.13 0.29 0.25 0.2 0.18 0.22 1 0.15 0.16 0.057
                                                                                                                                                                                                                                                   - -0.75
                          PAY_AMT50;0006<mark>8.22</mark>0.00170.040.0010.02230.0580.0370.0360.0340.0380.0460.17 0.16 0.18 0.16 0.14 0.31 0.15 0.18 0.16 0.15 1 0.15-0.055
                          AMT1
```

```
In []: # write a function about one-hot encoding
    def one_hot_encoding(df, col_list):
        df = df.copy()
        for col in col_list:
            dummies = pd.get_dummies(df[col], prefix=col[:4])
            df = pd.concat([df, dummies], axis=1)
            df = df.drop(col, axis=1)
        return df
In []: def pre_processing_class(df,scaler):
    # remove uncessary columns
    df = df.drop(['ID'], axis=1)
```

X = df.drop(['default payment next month'], axis=1)

X = pd.DataFrame(scaler.fit\_transform(X), columns=X.columns)

col\_list = ["EDUCATION", "MARRIAGE"]
df = one\_hot\_encoding(df, col\_list)

# One-hot encoding

# scale the data

```
y = df['default payment next month']
return X,y
```

# Task1: Build a classification model that predicts whether or not a customer will default on their next payment

```
In [ ]: # scaler name
        scaler_standard = StandardScaler()
        scaler_min_max = MinMaxScaler()
        scaler_robust = RobustScaler()
        scaler norm = Normalizer()
        X, y = pre_processing_class(df, scaler_standard)
In [ ]: def train_test_split_class(X,y,random_state=33,test_size=0.2,oversample=F
            # split the data
            if cross val:
                pass
                # do something
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=t
            if oversample:
                X_train_oversample, y_train_oversample = oversampler.fit_resample
            else:
                X_train_oversample, y_train_oversample = [],[]
            return X_train, X_test, y_train, y_test,X_train_oversample, y_train_o
In [ ]: | def train_models(name, model, X_train, y_train):
            model.fit(X_train, y_train)
            print(name + ' trained.')
            print('Training accuracy: {:.2f}%'.format(accuracy_score(y_train, mod
        def evaluate_models(name, model, X_test, y_test):
            # accuracy
            print(name + ' Accuracy: {:.2f}%'.format(model.score(X_test, y_test)
            # ROC AUC
            if name != 'SVC':
                print(name + ' ROC AUC: {:.2f}%'.format(roc_auc_score(y_test, mod
            # confusion matrix
            y_pred = model.predict(X_test)
            cm = confusion_matrix(y_test, y_pred)
            recall = cm[1][1] / (cm[1][1] + cm[1][0])
            precision = cm[1][1] / (cm[1][1] + cm[0][1])
            specificity = cm[0][0] / (cm[0][0] + cm[0][1])
            print(name + ' Recall: {:.2f}%'.format(recall * 100))
            print(name + ' Precision: {:.2f}%'.format(precision * 100))
            print(name + ' Specificity: {:.2f}%'.format(specificity * 100))
            # confusion matrix report
            print(name + ' Confusion Matrix Report: \n', classification_report(y_
            if name != 'SVC':
                # ROC curve
                fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
```

```
plt.figure(figsize=(8, 6))
                plt.plot(fpr, tpr, label=name)
                plt.plot([0, 1], [0, 1], color='black', linestyle='--')
                plt.xlabel('False Positive Rate (Fall-Out)')
                plt.ylabel('True Positive Rate (Recall)')
                plt.title('ROC Curve')
                plt.legend()
                plt.show()
        # test overfitting
        def cross_validation(name, model, X, y):
            scores = cross val score(model, X, y, cv=5)
            print(name + ' Cross Validation Accuracy: {:.2f}%'.format(scores.mean
In [ ]: def model_start(model_name, X_train, y_train, X, y, X_test, y_test):
            train_models(model_name, models_classific[model_name], X_train, y_trai
            evaluate_models(model_name,models_classific[model_name], X_test, y_te
            cross_validation(model_name, models_classific[model_name], X, y)
        def train_different_sampler(X,y,modelname,random_state=33):
            sampler = {
                'SMOTE': SMOTE(random_state=random_state),
                'SMOTETomek': SMOTETomek(random state=random state),
                'SMOTEENN': SMOTEENN(random_state=random_state),
                'RandomOverSampler': RandomOverSampler(random state=random state)
                # 'Knnor': data_augment.KNNOR()
            }
            print("----")
            print("Training without oversampling")
            X_train, X_test, y_train, y_test, X_train_oversample, y_train_oversam
            model_start(modelname, X_train, y_train, X, y, X_test, y_test)
            print("----")
            print("Oversampler Using Different Sampler")
            print("----")
            for name, sampler in sampler.items():
                print("OverSampling with " + name)
                X_train, X_test, y_train, y_test, X_train_oversample, y_train_ove
                # model_name,X_train,y_train,X,y,X_test,y_test
                model_start(modelname,X_train_oversample,y_train_oversample,X,y,X
                print("-----
```

## **Logistic Regression**

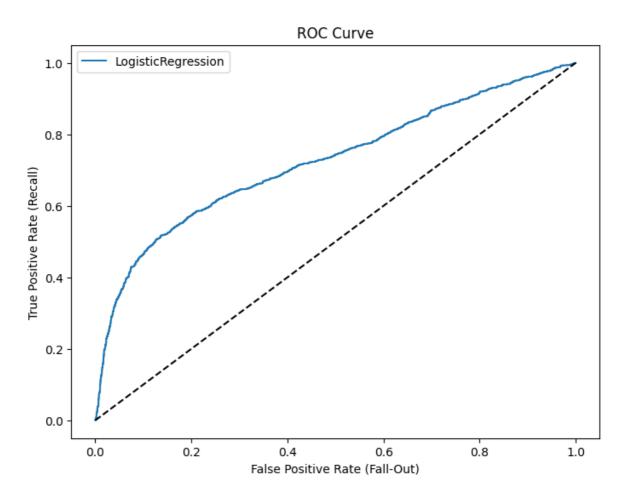
```
In [ ]: train_different_sampler(X,y,'LogisticRegression')
```

\_\_\_\_\_

Training without oversampling LogisticRegression trained. Training accuracy: 81.11%

LogisticRegression Accuracy: 81.22% LogisticRegression ROC AUC: 72.42% LogisticRegression Recall: 23.68% LogisticRegression Precision: 72.62% LogisticRegression Specificity: 97.48% LogisticRegression Confusion Matrix Report:

LOGISCICI	icg. c.	JJION CONTAGI	OII HACLEX	ricpor ci	
		precision	recall	f1-score	support
	0	0.82	0.97	0.89	4678
	1	0.73	0.24	0.36	1322
accur	acy			0.81	6000
macro	avg	0.77	0.61	0.62	6000
weighted	avg	0.80	0.81	0.77	6000



LogisticRegression Cross Validation Accuracy: 81.03%

-----

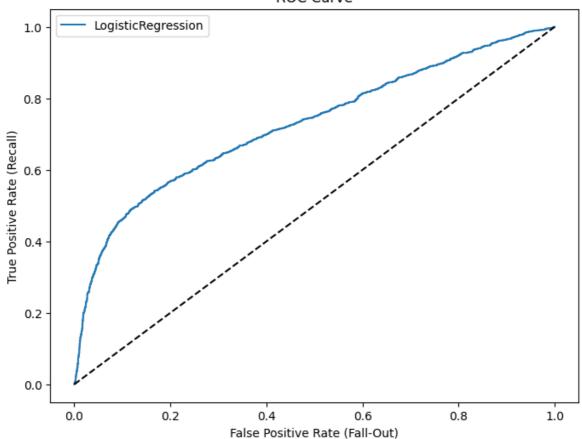
#### Oversampler Using Different Sampler

OverSampling with SMOTE

LogisticRegression trained. Training accuracy: 67.59%

LogisticRegression Accuracy: 66.88% LogisticRegression ROC AUC: 72.62% LogisticRegression Recall: 65.36% LogisticRegression Precision: 36.11% LogisticRegression Specificity: 67.32% LogisticRegression Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.87	0.67	0.76	4678
1	0.36	0.65	0.47	1322
accuracy			0.67	6000
macro avg	0.62	0.66	0.61	6000
weighted avg	0.76	0.67	0.70	6000



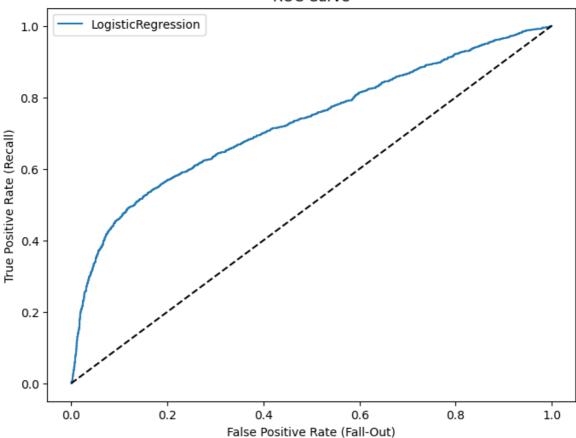
LogisticRegression Cross Validation Accuracy: 81.03%

-----

OverSampling with SMOTETomek LogisticRegression trained. Training accuracy: 68.05%

LogisticRegression Accuracy: 67.00% LogisticRegression ROC AUC: 72.63% LogisticRegression Recall: 65.28% LogisticRegression Precision: 36.20% LogisticRegression Specificity: 67.49% LogisticRegression Confusion Matrix Report:

3	precision	recall	f1-score	support
0	0.87	0.67	0.76	4678
1	0.36	0.65	0.47	1322
accuracy			0.67	6000
macro avg	0.62	0.66	0.61	6000
weighted avg	0.76	0.67	0.70	6000



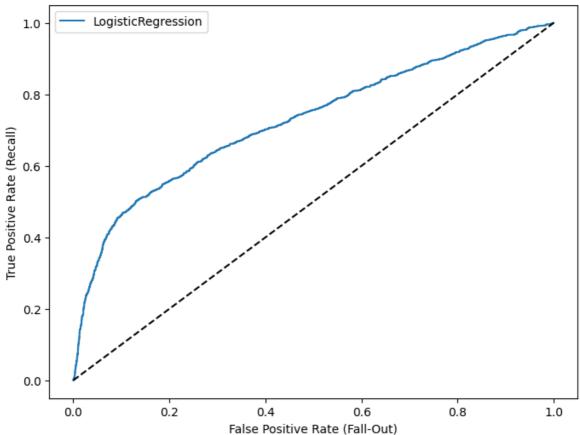
LogisticRegression Cross Validation Accuracy: 81.03%

-----

OverSampling with SMOTEENN LogisticRegression trained. Training accuracy: 71.31%

LogisticRegression Accuracy: 55.00% LogisticRegression ROC AUC: 72.63% LogisticRegression Recall: 75.95% LogisticRegression Precision: 29.65% LogisticRegression Specificity: 49.08% LogisticRegression Confusion Matrix Report:

5	precision	recall	f1-score	support
0	0.88	0.49	0.63	4678
1	0.30	0.76	0.43	1322
accuracy			0.55	6000
macro avg	0.59	0.63	0.53	6000
weighted avg	0.75	0.55	0.58	6000



LogisticRegression Cross Validation Accuracy: 81.03%

-----

OverSampling with RandomOverSampler

LogisticRegression trained. Training accuracy: 67.17%

LogisticRegression Accuracy: 67.53% LogisticRegression ROC AUC: 72.65% LogisticRegression Recall: 65.81% LogisticRegression Precision: 36.77% LogisticRegression Specificity: 68.02% LogisticRegression Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.88	0.68	0.77	4678
1	0.37	0.66	0.47	1322
accuracy			0.68	6000
macro avg	0.62	0.67	0.62	6000
weighted avg	0.76	0.68	0.70	6000

# **ROC Curve** LogisticRegression 1.0 0.8 True Positive Rate (Recall) 0.6 0.4 0.2 0.0 0.0 0.2 0.4 0.6 0.8 1.0 False Positive Rate (Fall-Out)

LogisticRegression Cross Validation Accuracy: 81.03%

## **SVM**

In [ ]: train\_different\_sampler(X,y,'SVC')

Training without oversampling

SVC trained.

Training accuracy: 82.43% SVC Accuracy: 82.27%

SVC Recall: 32.83% SVC Precision: 71.15% SVC Specificity: 96.24% SVC Confusion Matrix Report:

		precision	recall	f1-score	support
	0	0.84	0.96	0.89	4678
	1	0.71	0.33	0.45	1322
accur	асу			0.82	6000
macro	avg	0.77	0.65	0.67	6000
weighted	avg	0.81	0.82	0.80	6000

SVC Cross Validation Accuracy: 81.95%

Oversampler Using Different Sampler

\_\_\_\_\_

OverSampling with SMOTE

SVC trained.

Training accuracy: 72.77% SVC Accuracy: 77.45%

SVC Recall: 57.41% SVC Precision: 49.00% SVC Specificity: 83.11%

SVC Confusion Matrix Report:

	pr	ecision	recall	f1-score	support
	0	0.87	0.83	0.85	4678
	1	0.49	0.57	0.53	1322
accura	су			0.77	6000
macro a	vg	0.68	0.70	0.69	6000
weighted a	vg	0.79	0.77	0.78	6000

SVC Cross Validation Accuracy: 81.95%

OverSampling with SMOTETomek

SVC trained.

Training accuracy: 73.36%

SVC Accuracy: 77.40% SVC Recall: 57.49% SVC Precision: 48.91% SVC Specificity: 83.03% SVC Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.87	0.83	0.85	4678
1	0.49	0.57	0.53	1322
accuracy			0.77	6000
macro avg	0.68	0.70	0.69	6000
weighted avg	0.79	0.77	0.78	6000

SVC Cross Validation Accuracy: 81.95%

OverSampling with SMOTEENN

SVC trained.

Training accuracy: 83.50% SVC Accuracy: 66.22% SVC Recall: 72.09% SVC Precision: 36.50%

SVC Specificity: 64.56% SVC Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.89	0.65	0.75	4678
1	0.36	0.72	0.48	1322
accuracy			0.66	6000
macro avg	0.63	0.68	0.62	6000
weighted avg	0.78	0.66	0.69	6000

SVC Cross Validation Accuracy: 81.95%

\_\_\_\_\_\_

OverSampling with RandomOverSampler

SVC trained.

Training accuracy: 72.63% SVC Accuracy: 77.28% SVC Recall: 58.47% SVC Precision: 48.71% SVC Specificity: 82.60% SVC Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.88	0.83	0.85	4678
1	0.49	0.58	0.53	1322
accuracy			0.77	6000
macro avg	0.68	0.71	0.69	6000
weighted avg	0.79	0.77	0.78	6000

SVC Cross Validation Accuracy: 81.95%

\_\_\_\_\_

#### Random Forest model

In [ ]: train\_different\_sampler(X,y,'DecisionTreeClassifier')

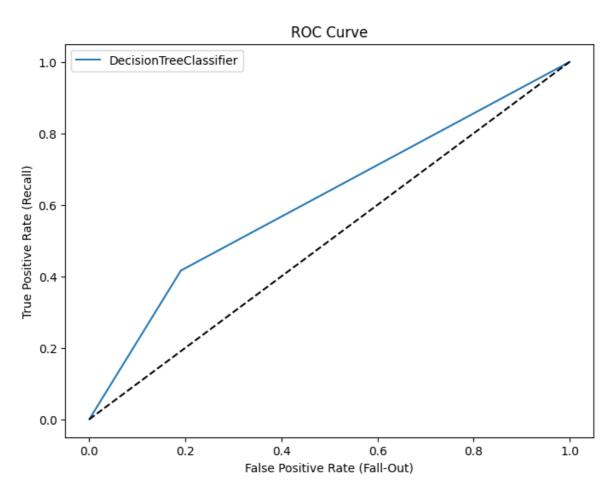
\_\_\_\_\_

Training without oversampling DecisionTreeClassifier trained.

Training accuracy: 99.93%

DecisionTreeClassifier Accuracy: 72.25%
DecisionTreeClassifier ROC AUC: 61.27%
DecisionTreeClassifier Recall: 41.75%
DecisionTreeClassifier Precision: 38.15%
DecisionTreeClassifier Specificity: 80.87%
DecisionTreeClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.83	0.81	0.82	4678
1	0.38	0.42	0.40	1322
accuracy			0.72	6000
macro avg	0.61	0.61	0.61	6000
weighted avg	0.73	0.72	0.73	6000



#### DecisionTreeClassifier Cross Validation Accuracy: 72.19%

#### Oversampler Using Different Sampler

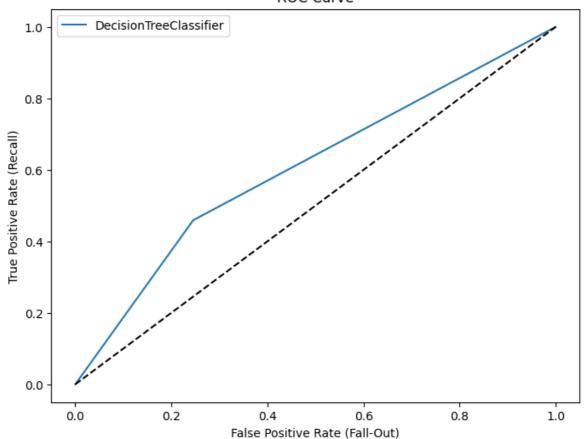
OverSampling with SMOTE

DecisionTreeClassifier trained.

Training accuracy: 99.96%

DecisionTreeClassifier Accuracy: 68.93% DecisionTreeClassifier ROC AUC: 60.68% DecisionTreeClassifier Recall: 45.99% DecisionTreeClassifier Precision: 34.58% DecisionTreeClassifier Specificity: 75.42% DecisionTreeClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.83	0.75	0.79	4678
1	0.35	0.46	0.39	1322
accuracy			0.69	6000
macro avg	0.59	0.61	0.59	6000
weighted avg	0.72	0.69	0.70	6000



#### DecisionTreeClassifier Cross Validation Accuracy: 72.31%

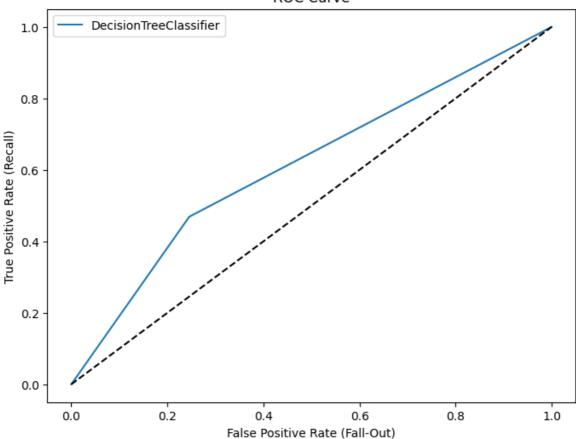
-----

OverSampling with SMOTETomek DecisionTreeClassifier trained. Training accuracy: 99.96%

DecisionTreeClassifier Accuracy: 69.15% DecisionTreeClassifier ROC AUC: 61.16% DecisionTreeClassifier Recall: 46.90%

DecisionTreeClassifier Precision: 35.05% DecisionTreeClassifier Specificity: 75.44% DecisionTreeClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.83	0.75	0.79	4678
1	0.35	0.47	0.40	1322
accuracy			0.69	6000
macro avg	0.59	0.61	0.60	6000
weighted avg	0.73	0.69	0.71	6000



#### DecisionTreeClassifier Cross Validation Accuracy: 72.29%

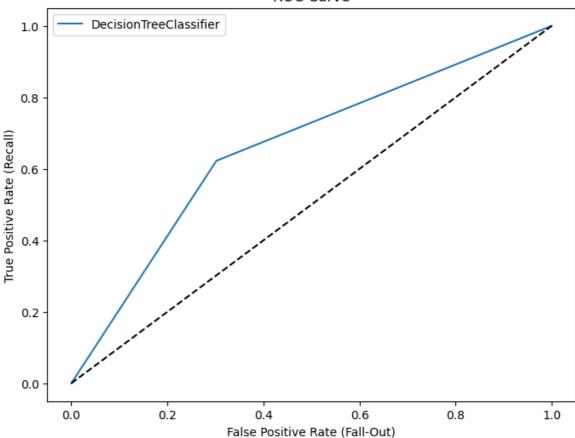
-----

OverSampling with SMOTEENN DecisionTreeClassifier trained.

Training accuracy: 100.00%

DecisionTreeClassifier Accuracy: 68.13%
DecisionTreeClassifier ROC AUC: 66.02%
DecisionTreeClassifier Recall: 62.25%
DecisionTreeClassifier Precision: 36.81%
DecisionTreeClassifier Specificity: 69.79%
DecisionTreeClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.87	0.70	0.77	4678
1	0.37	0.62	0.46	1322
accuracy			0.68	6000
macro avg	0.62	0.66	0.62	6000
weighted avg	0.76	0.68	0.71	6000



#### DecisionTreeClassifier Cross Validation Accuracy: 72.43%

\_\_\_\_\_

OverSampling with RandomOverSampler DecisionTreeClassifier trained.

Training accuracy: 99.95%

DecisionTreeClassifier Accuracy: 73.05%
DecisionTreeClassifier ROC AUC: 61.09%
DecisionTreeClassifier Recall: 39.71%
DecisionTreeClassifier Precision: 39.03%
DecisionTreeClassifier Specificity: 82.47%
DecisionTreeClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.83	0.82	0.83	4678
1	0.39	0.40	0.39	1322
accuracy			0.73	6000
macro avg	0.61	0.61	0.61	6000
weighted avg	0.73	0.73	0.73	6000

# **ROC Curve** DecisionTreeClassifier 1.0 0.8 True Positive Rate (Recall) 0.6 0.4 0.2 0.0 0.0 0.2 0.4 0.6 0.8 1.0 False Positive Rate (Fall-Out)

DecisionTreeClassifier Cross Validation Accuracy: 72.28%

In [ ]: train\_different\_sampler(X,y,'RandomForestClassifier')

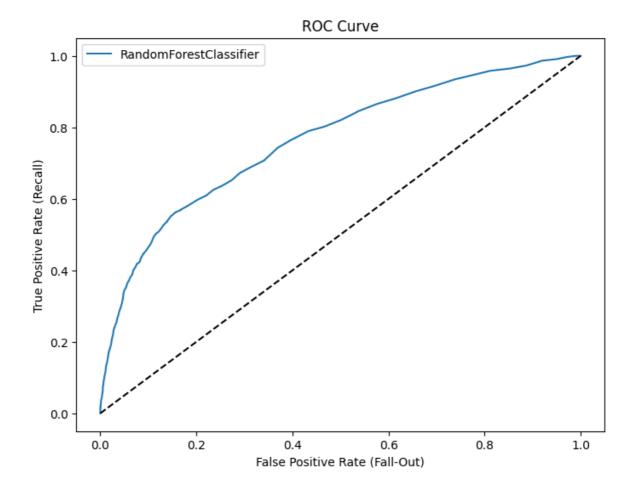
\_\_\_\_\_

Training without oversampling RandomForestClassifier trained.

Training accuracy: 99.93%

RandomForestClassifier Accuracy: 81.60%
RandomForestClassifier ROC AUC: 76.43%
RandomForestClassifier Recall: 36.38%
RandomForestClassifier Precision: 64.65%
RandomForestClassifier Specificity: 94.38%
RandomForestClassifier Confusion Matrix Report:

		precision	recall	f1-score	support
	0	0.84	0.94	0.89	4678
	1	0.65	0.36	0.47	1322
accur	асу			0.82	6000
macro	avg	0.74	0.65	0.68	6000
weighted	avg	0.80	0.82	0.80	6000



#### RandomForestClassifier Cross Validation Accuracy: 81.51%

\_\_\_\_\_

#### Oversampler Using Different Sampler

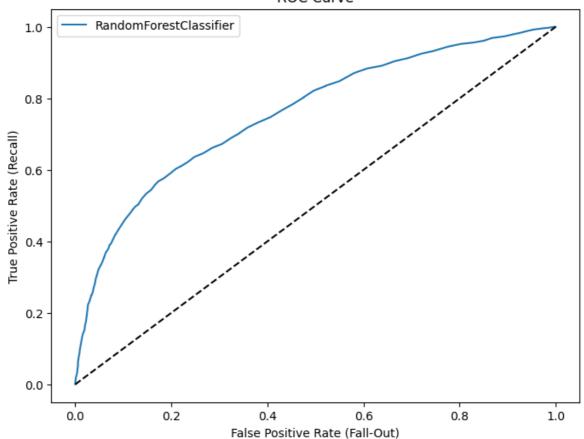
OverSampling with SMOTE

RandomForestClassifier trained.

Training accuracy: 99.96%

RandomForestClassifier Accuracy: 79.77%
RandomForestClassifier ROC AUC: 75.85%
RandomForestClassifier Recall: 47.28%
RandomForestClassifier Precision: 54.73%
RandomForestClassifier Specificity: 88.95%
RandomForestClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.86	0.89	0.87	4678
1	0.55	0.47	0.51	1322
accuracy			0.80	6000
macro avg	0.70	0.68	0.69	6000
weighted avg	0.79	0.80	0.79	6000



#### RandomForestClassifier Cross Validation Accuracy: 81.36%

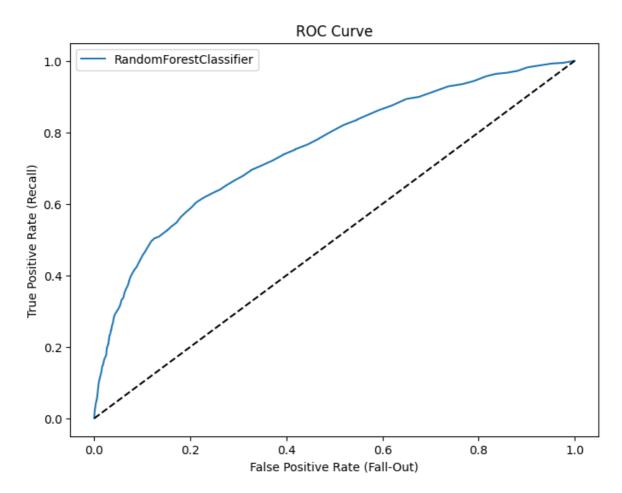
-----

OverSampling with SMOTETomek RandomForestClassifier trained.

Training accuracy: 99.96%

RandomForestClassifier Accuracy: 79.90%
RandomForestClassifier ROC AUC: 75.38%
RandomForestClassifier Recall: 47.43%
RandomForestClassifier Precision: 55.10%
RandomForestClassifier Specificity: 89.08%
RandomForestClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.86	0.89	0.87	4678
1	0.55	0.47	0.51	1322
accuracy			0.80	6000
macro avg weighted avg	0.70 0.79	0.68 0.80	0.69 0.79	6000 6000



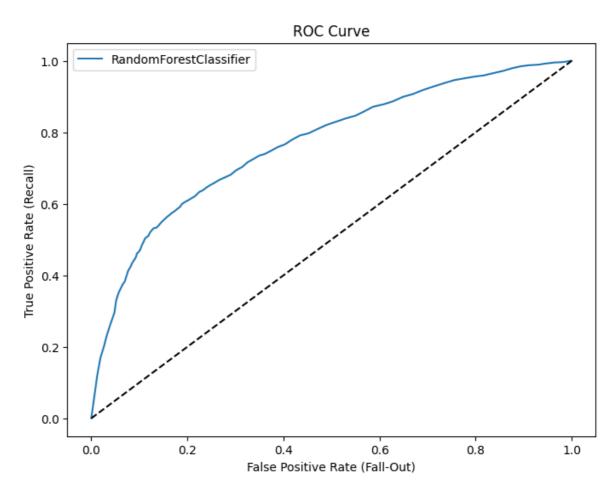
#### RandomForestClassifier Cross Validation Accuracy: 81.50%

-----

OverSampling with SMOTEENN RandomForestClassifier trained. Training accuracy: 100.00%

RandomForestClassifier Accuracy: 75.27%
RandomForestClassifier ROC AUC: 76.77%
RandomForestClassifier Recall: 61.57%
RandomForestClassifier Precision: 45.47%
RandomForestClassifier Specificity: 79.14%
RandomForestClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.88	0.79	0.83	4678
1	0.45	0.62	0.52	1322
accuracy			0.75	6000
macro avg weighted avg	0.67 0.79	0.70 0.75	0.68 0.76	6000 6000



#### RandomForestClassifier Cross Validation Accuracy: 81.49%

\_\_\_\_\_

OverSampling with RandomOverSampler RandomForestClassifier trained. Training accuracy: 99.95%

RandomForestClassifier Accuracy: 80.88%
RandomForestClassifier ROC AUC: 76.80%
RandomForestClassifier Recall: 42.97%
RandomForestClassifier Precision: 59.11%
RandomForestClassifier Specificity: 91.60%
RandomForestClassifier Confusion Matrix Report:

	precision	recall	f1-score	support
0	0.85	0.92	0.88	4678
1	0.59	0.43	0.50	1322
accuracy			0.81	6000
macro avg	0.72	0.67	0.69	6000
weighted avg	0.79	0.81	0.80	6000

# **ROC Curve** RandomForestClassifier 1.0 0.8 True Positive Rate (Recall) 0.6 0.4 0.2 0.0 0.2 0.0 0.4 0.6 0.8 1.0 False Positive Rate (Fall-Out)

RandomForestClassifier Cross Validation Accuracy: 81.59%

Build a regression model that predicts a customer's limit balance if they were a new customer to the bank with only X0, X2-5 available to you as data.

```
In [ ]: def pre_processing_regre(df,scaler):
    # remove uncessary columns
    df = df.drop(['ID'], axis=1)
```

```
# One-hot encoding
            col_list = ["EDUCATION", "MARRIAGE"]
            df = one_hot_encoding(df, col_list)
            # scale the data
            X = df.drop(['LIMIT_BAL'], axis=1)
            X = pd.DataFrame(scaler.fit transform(X), columns=X.columns)
            y = df['LIMIT BAL']
            return X, y
        X, y = pre_processing_regre(df, scaler_standard)
In [ ]: models_regre = {
            'LinearRegression': LinearRegression(),
            'SVR': SVR(),
             'DecisionTreeRegressor': DecisionTreeRegressor(),
            'RandomForestRegressor': RandomForestRegressor(),
In [ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
        def train_models(name, model, X_train, y_train):
            model.fit(X_train, y_train)
            print(name + ' trained.')
        def evaluate_models(name, model, X_test, y_test):
            print(name + ' Accuracy: {:.2f}%'.format(model.score(X_test, y_test)
```

## **Linear Regression**

```
In []: train_models("linear_regression", models_regre["LinearRegression"], X_trai
    evaluate_models("linear_regression", models_regre["LinearRegression"], X_t
    linear_regression trained.
    linear_regression Accuracy: 36.10%
```

#### Random Forest

```
In []: train_models("SVR", models_regre["SVR"], X_train, y_train)
        evaluate_models("SVR", models_regre["SVR"], X_test, y_test)

SVR trained.
    SVR Accuracy: -4.24%

In []: train_models("DecisionTreeRegressor", models_regre["DecisionTreeRegressor"
        evaluate_models("DecisionTreeRegressor", models_regre["DecisionTreeRegress
        DecisionTreeRegressor trained.
        DecisionTreeRegressor Accuracy: -6.16%

In []: train_models("RandomForestRegressor", models_regre["RandomForestRegressor"
        evaluate_models("RandomForestRegressor", models_regre["RandomForestRegress
        RandomForestRegressor trained.
        RandomForestRegressor Accuracy: 46.90%
```

#### **ANN** model

In []: