



Gr5

TAIWANESE BANKRUPTCY PREDICTION

Taiwanese Bankruptcy Prediction Data Set

SUPER TEAM



6487030
Talaborn Wutthicharoenkit



6487032
Thanawat Rattanakwamdee



6487063
Suchanan Manmark

Dataset ของเราคืออะไร

Company Bankruptcy Prediction

class 0 คือไม่ล้มละลาย และ 1 คือล้มละลาย

96 feature

Import

```
[ ] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import make_scorer, f1_score, precision_score, recall_score, accuracy_score
from sklearn.datasets import make_classification
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.utils.class_weight import compute_sample_weight
from xgboost import XGBClassifier
from fast_ml.utilities import display_all
from fast_ml.feature_selection import get_duplicate_features
from imblearn.over_sampling import SMOTE
```

Clean Data

01

Duplicate

02

Check Null and Zero

03

Class 0 and 1 balance

Duplicate Index

ข้อมูลที่มีความสัมพันธ์กัน(ไปในทางเดียวกัน)

	Desc	feature1	feature2
0	Duplicate Index	Debt ratio %	Net worth/Assets
1	Duplicate Index	Total income/Total expense	Net Income to Stockholder's Equity
2	Duplicate Index	Total expense/Assets	Cash Flow to Total Assets
3	Duplicate Index	Total expense/Assets	Current Liability to Equity
4	Duplicate Index	Total expense/Assets	Cash Flow to Sales
5	Duplicate Index	Total expense/Assets	Working capital Turnover Rate
6	Duplicate Index	Total income/Total expense	Equity to Liability
7	Duplicate Index	Total income/Total expense	Liability to Equity
8	Duplicate Index	Total income/Total expense	No-credit Interval
9	Duplicate Index	Total income/Total expense	Cash Flow to Sales

Duplicate Index

```
[ ] # drop these duplicate features from dataset
print('Shape of Dataset before dropping the duplicate index features: ', data.shape)
data.drop(columns = duplicate_index_features_list, inplace=True)
print('Shape of Dataset after dropping the duplicate index features: ', data.shape)
```



Shape of Dataset before dropping the duplicate index features: (6819, 94)
Shape of Dataset after dropping the duplicate index features: (6819, 69)

จาก 96 เหลือ 69 Feature

Check Null and Zero

```
[ ] checkNull = data.isnull().sum()  
for key,value in checkNull.iteritems():  
    print(key,"",value)
```

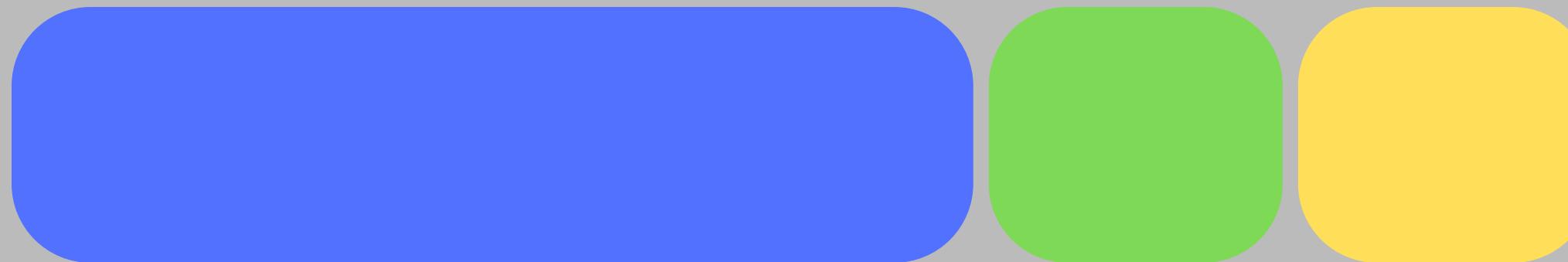


Bankrupt? , 0
ROA(C) before interest and depreciation before interest ,
ROA(A) before interest and % after tax , 0
ROA(B) before interest and depreciation after tax , 0
Operating Gross Margin , 0
Realized Sales Gross Margin , 0
Operating Profit Rate , 0
Pre-tax net Interest Rate , 0
After-tax net Interest Rate , 0
Non-industry income and expenditure/revenue , 0
Continuous interest rate (after tax) , 0
Operating Expense Rate , 0
Research and development expense rate , 0
Cash flow rate , 0
Interest-bearing debt interest rate , 0
Tax rate (A) , 0
Net Value Per Share (B) , 0
Net Value Per Share (A) , 0
Net Value Per Share (C) , 0

```
checkZero = data.eq(0).sum()  
for key,value in checkZero.iteritems():  
    print(key,"",value)  
  
Research and development expense rate , 1424  
Cash flow rate , 1  
Interest-bearing debt interest rate , 891  
Tax rate (A) , 2568  
Net Value Per Share (B) , 1  
Net Value Per Share (A) , 1  
Net Value Per Share (C) , 1  
Persistent EPS in the Last Four Seasons , 1  
Cash Flow Per Share , 1  
Revenue Per Share (Yuan ¥) , 2  
Operating Profit Per Share (Yuan ¥) , 1  
Per Share Net profit before tax (Yuan ¥) , 1  
Realized Sales Gross Profit Growth Rate , 1  
Operating Profit Growth Rate , 1  
After-tax Net Profit Growth Rate , 1  
Regular Net Profit Growth Rate , 1  
Continuous Net Profit Growth Rate , 1  
Total Asset Growth Rate , 1  
Net Value Growth Rate , 1  
Total Asset Return Growth Rate Ratio , 1  
Cash Reinvestment % , 1  
Current Ratio , 1  
Quick Ratio , 1  
Interest Expense Ratio , 1  
Total debt/Total net worth , 1  
Debt ratio % , 1  
Long-term fund suitability ratio (A) , 1  
Borrowing dependency , 1  
Contingent liabilities/Net worth , 1  
Operating profit/Paid-in capital , 1  
Net profit before tax/Paid-in capital , 1  
Inventory and accounts receivable/Net value , 1  
Total Asset Turnover , 8
```

แบ่งข้อมูลเพื่อ train, valid และ test model

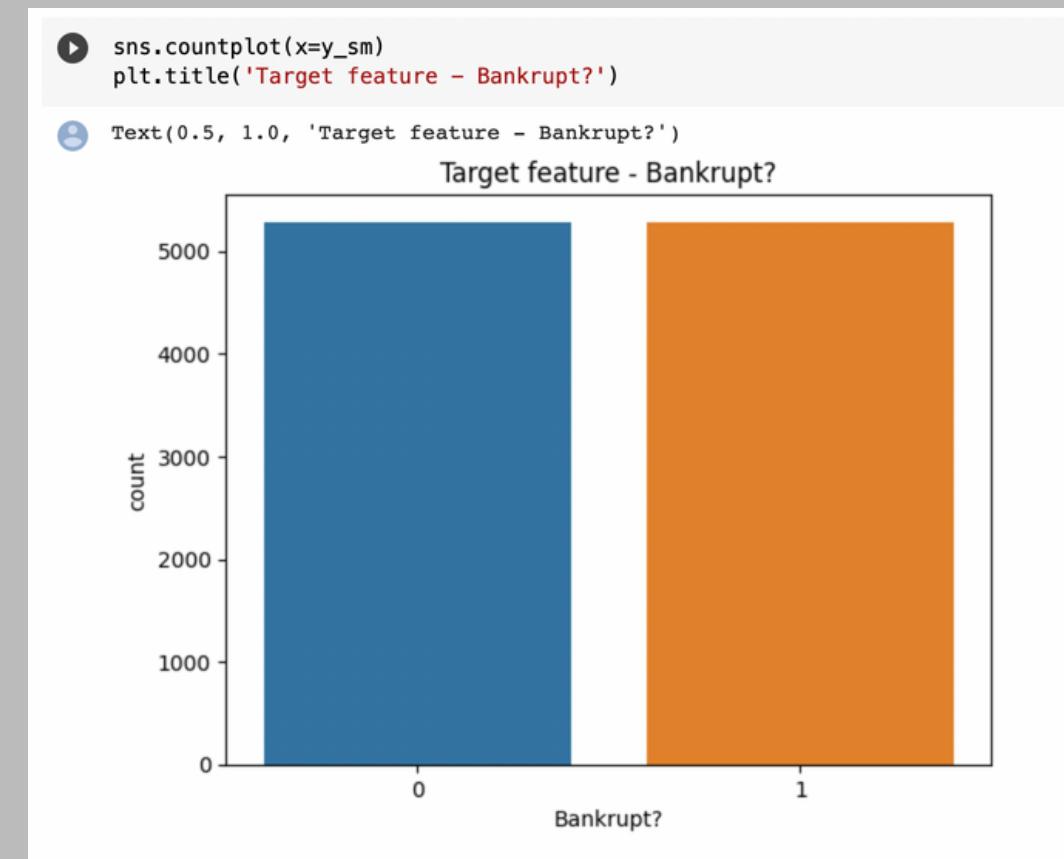
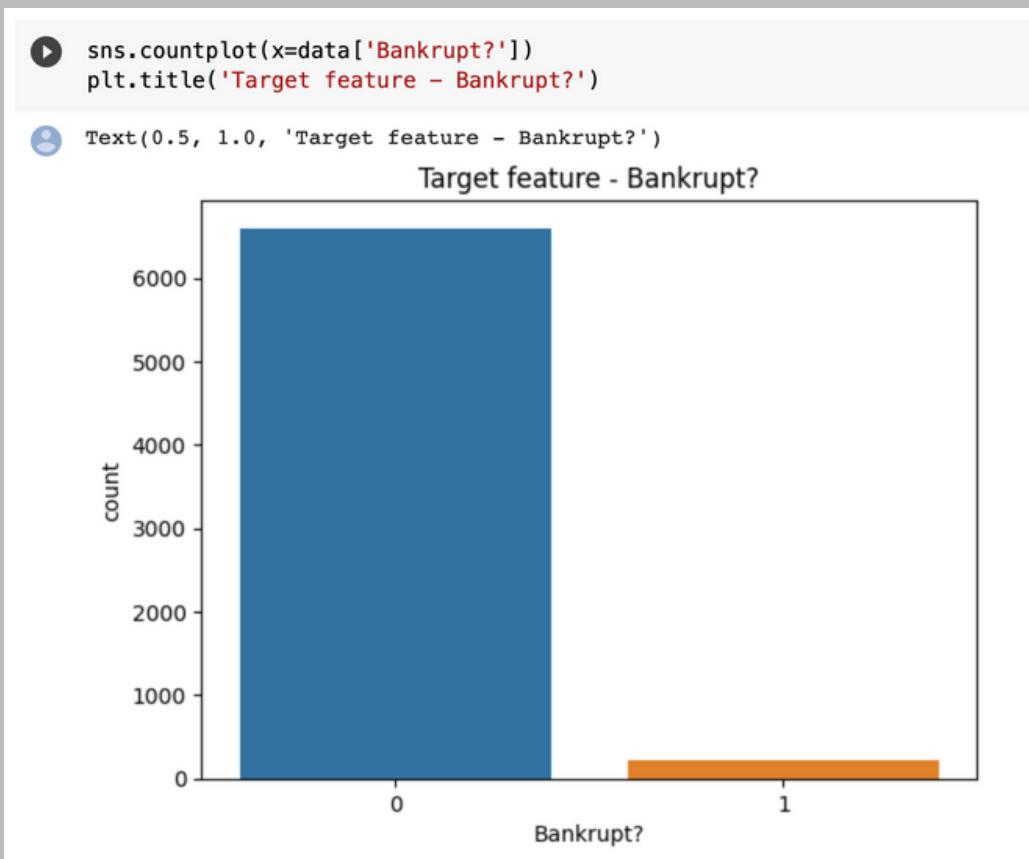
```
▶ X_Will_SM, X_test, y_Will_SM, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
X_test, X_valid, y_test, y_valid = train_test_split(X_test, y_test, test_size=0.5, random_state=1)
```



อัตราส่วน 60 : 20 : 20

Class 0 and 1 balance ໄລຍ້ smote data

@ train data



X_sm, y_sm = sm.fit_resample(X_Will_SM, y_Will_SM)

print('balance of 1 and 0 classes:')
y_sm.value_counts()

balance of 1 and 0 classes:
0 5285
1 5285
Name: Bankrupt?, dtype: int64



Correlation

Use Correlation Coefficient to select feature

```
▶ corr = X_sm.corr()
print(corr)

Bankrupt          Bankrupt \
1.000000
ROA(C) before interest and depreciation before... -0.585940
ROA(A) before interest and % after tax       -0.569592
ROA(B) before interest and depreciation after tax -0.580041
Operating Gross Margin                  -0.332981
...
Liability-Assets Flag                 ...
Total assets to GNP price            0.024618
Gross Profit to Sales              0.053761
Degree of Financial Leverage (DFL)   -0.332987
Net Income Flag                      0.028264
                                         NaN

                                         ROA(C) before interest and depreciation before interest \
Bankrupt          -0.585940
ROA(C) before interest and depreciation before... 1.000000
ROA(A) before interest and % after tax       0.939364
ROA(B) before interest and depreciation after tax 0.989079
Operating Gross Margin                  0.377527
...
Liability-Assets Flag                 ...
Total assets to GNP price            -0.071774
Gross Profit to Sales              -0.132407
Degree of Financial Leverage (DFL)   0.377533
Net Income Flag                      0.000964
                                         NaN

                                         ROA(A) before interest and % after tax \

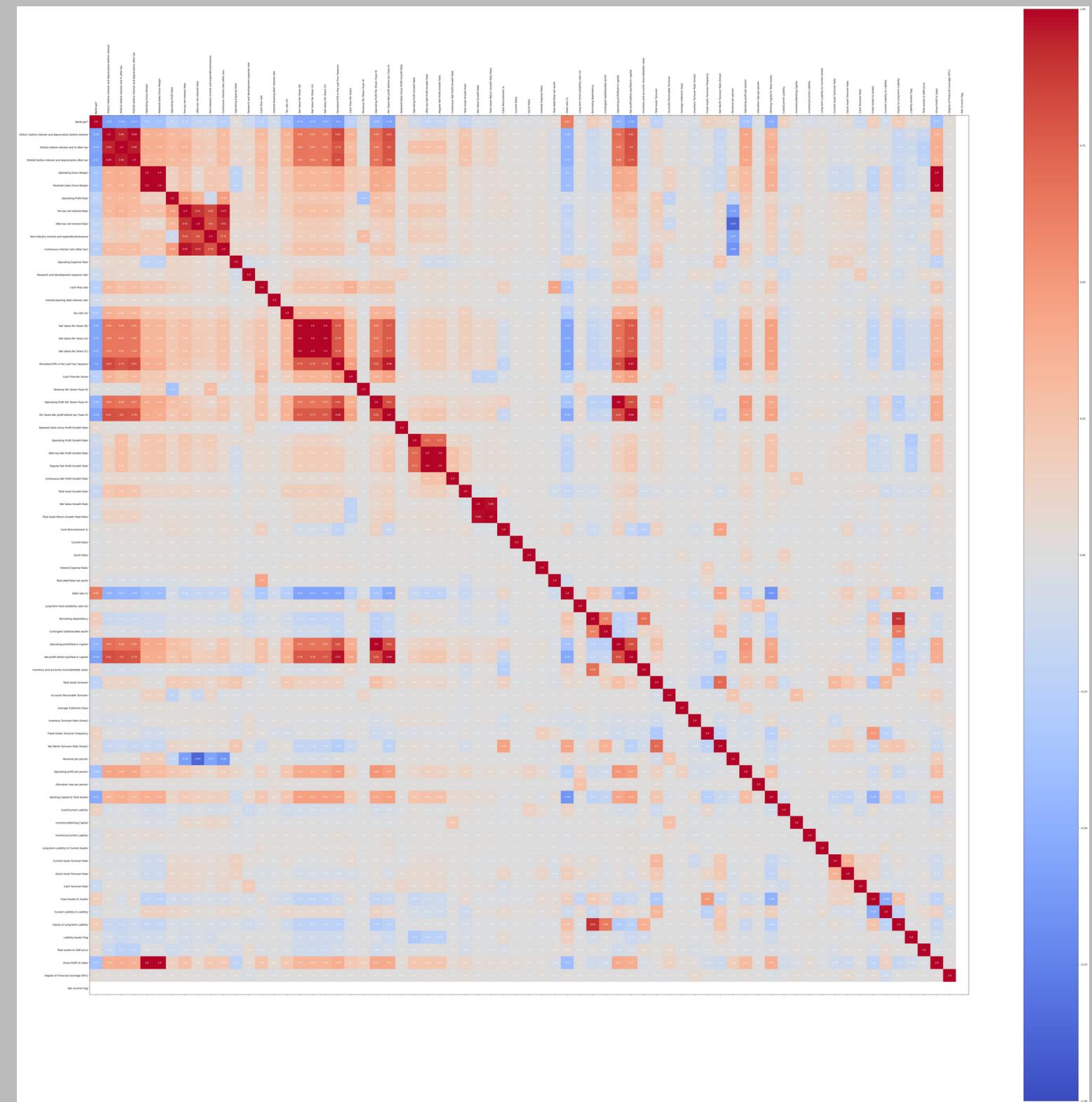
```

Graph

```
crm = corr

fig, ax = plt.subplots(figsize=(80, 80))
cax = ax.matshow(crm,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(data.columns),1)
ax.set_xticks(ticks)
ax.set_xticklabels(data.columns)
plt.xticks(rotation = 90)
ax.set_yticklabels(data.columns)
ax.set_yticks(ticks)

for i in range(data.shape[1]):
    for j in range(data.shape[1]):
        text = ax.text(j, i, round(crm.iloc[i][j],2),
                      ha="center", va="center", color="w")
plt.show()
```



Use CSV to analyze correlation

```
[ ] X_sm.corr().to_csv("correlation.csv")
```

top correlation

- "Debt ratio %",
- "Borrowing dependency",
- "Equity to Long-term Liability",
- "Fixed Assets to Assets",

```
[ ] X_sm =X_sm.drop(columns=['Bankrupt'])
```

	Bankrupt	Bankrupt ▾
Debt ratio %	0.62	1
Borrowing dependency	0.19	
Equity to Long-term Liability	0.17	
Fixed Assets to Assets	0.16	
Fixed Assets Turnover Frequency	0.13	
Inventory and accounts receivable/Net value	0.13	
Cash/Current Liability	0.1	
Net Worth Turnover Rate (times)	0.09	
Contingent liabilities/Net worth	0.07	

Feature

- "Debt ratio %"
- "Borrowing dependency"
- "Equity to Long-term Liability"
- "Fixed Assets to Assets"

Model

LogisticRegression

DecisionTreeClassifier

RandomForestClassifier

XGBoostClassifier

Support Vector Classification

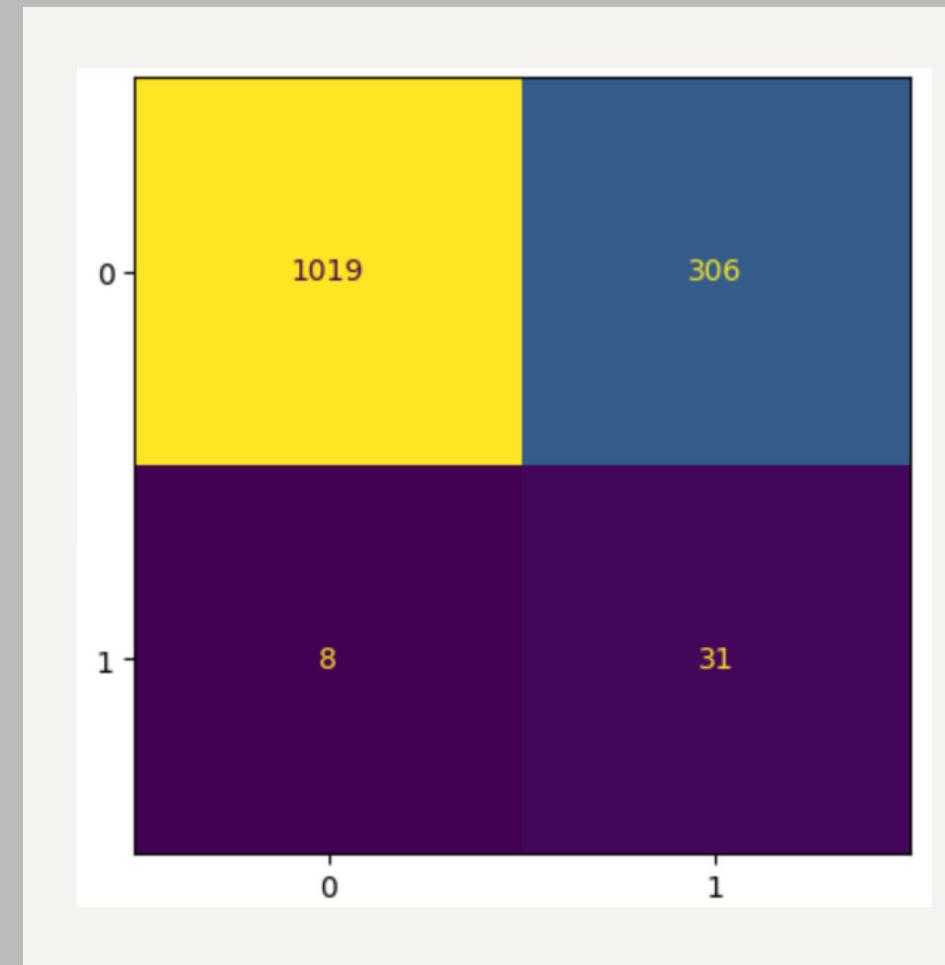
Model

Use GridSearchCV to find best parameter

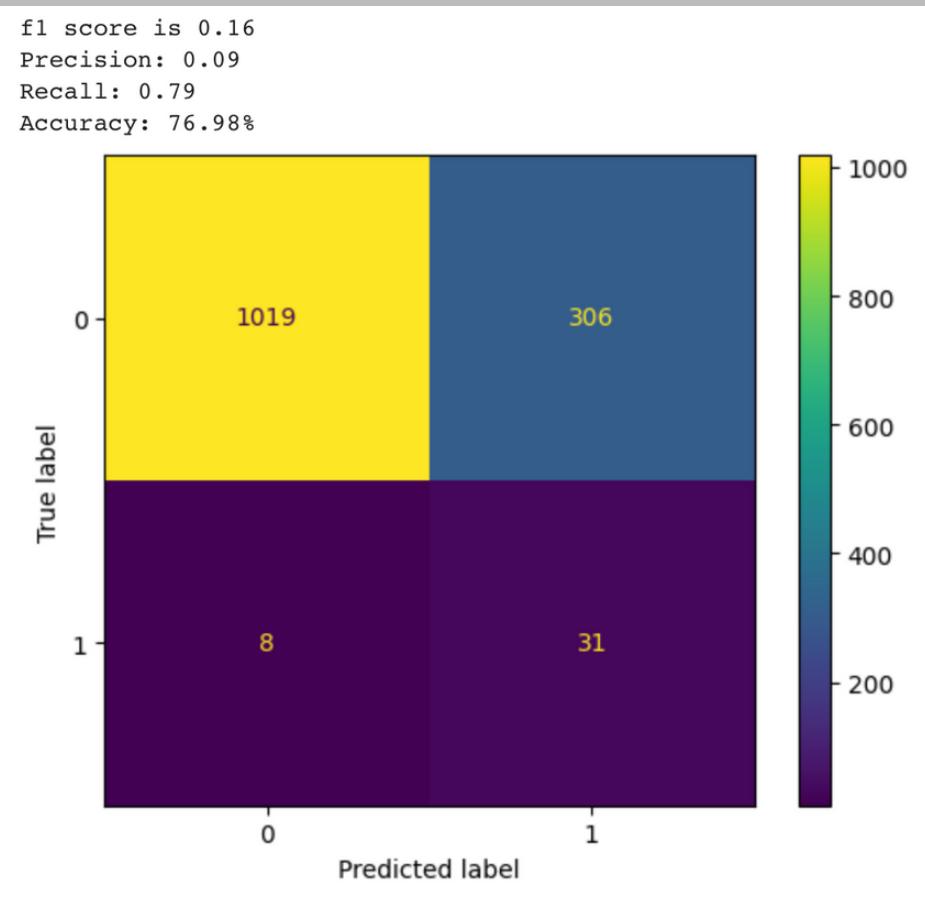
```
param= {  
    'n_estimators': [10, 20, 30],  
    'max_depth': [None, 1,2,5,6,10,12,15,18,20],  
    'min_samples_split': [2, 5, 7],  
    'class_weight': ['balanced', None],  
    'random_state' :[1]  
}  
  
rf = RandomForestClassifier()  
model = GridSearchCV(estimator=rf, param_grid=param)  
model.fit(X_train, y_train)  
  
print("best parameters: ",model.best_params_)  
  
model = model.best_estimator_  
y_pred = model.predict(X_test)  
  
print("f1 score is %.2f" % f1_score(y_test, y_pred))  
print("Precision: %.2f" % precision_score(y_test, y_pred))  
print("Recall: %.2f" % recall_score(y_test, y_pred))  
acc = accuracy_score(y_test, y_pred)  
print("Accuracy: %.2f%%" % (accuracy * 100.0))  
cm = confusion_matrix(y_test, y_pred, labels=model.classes_)  
disp = ConfusionMatrixDisplay(confusion_matrix=cm,  
                               display_labels=model.classes_)  
disp.plot()  
  
plt.show()
```

Confusion matrix

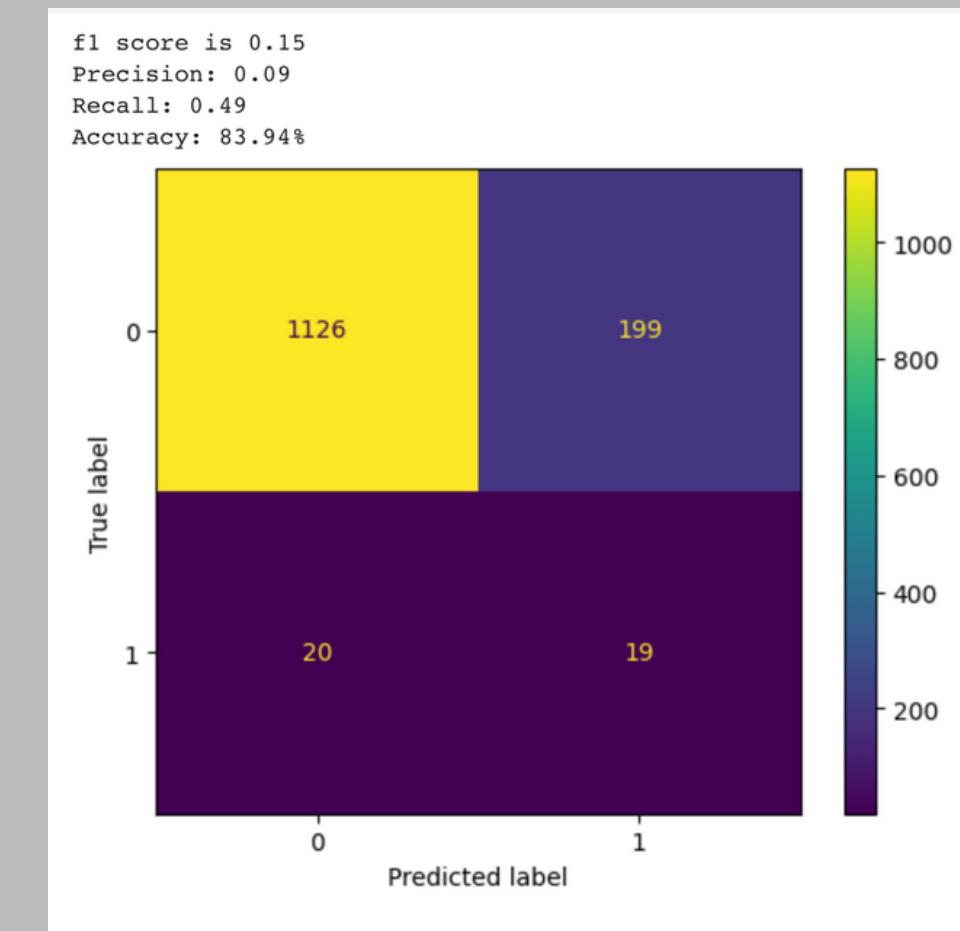
Use confusion matrix to measure performance.



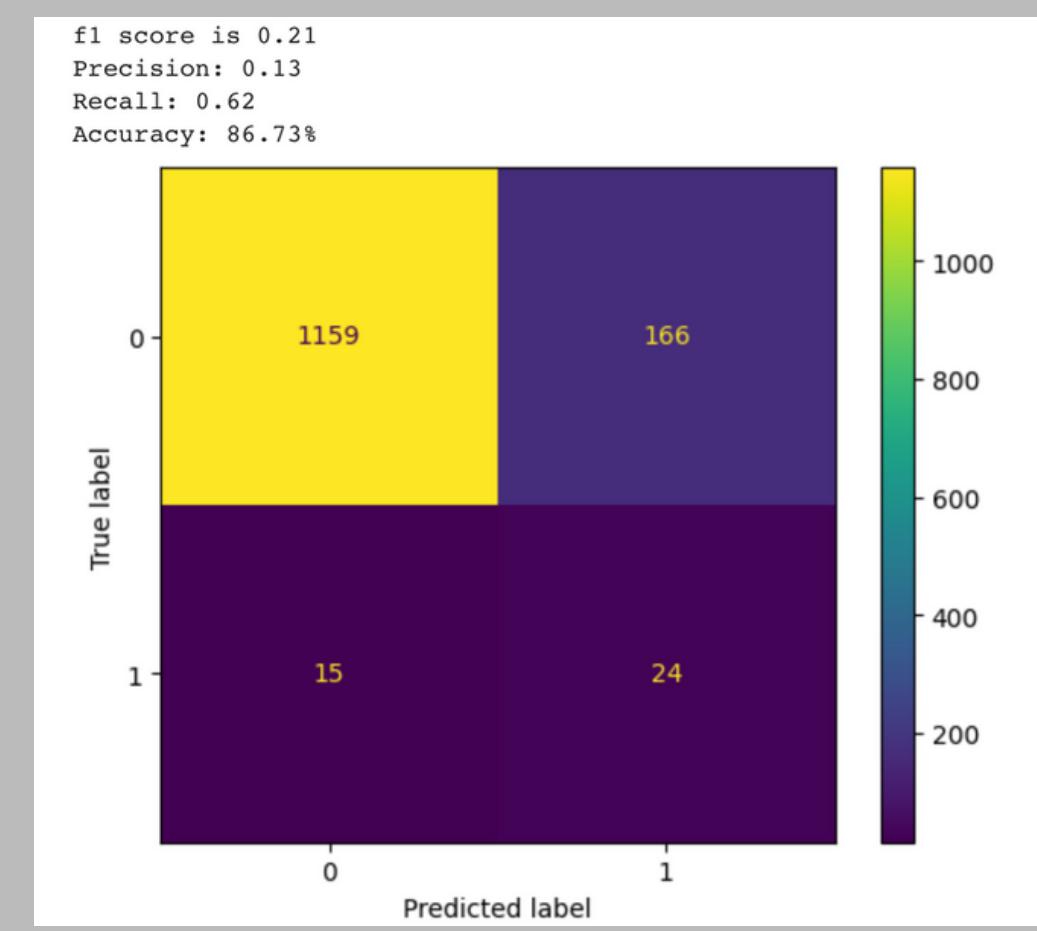
Performance matrix (Recall) and (Accuracy)
good when false positive is less important and classes are approximately balance



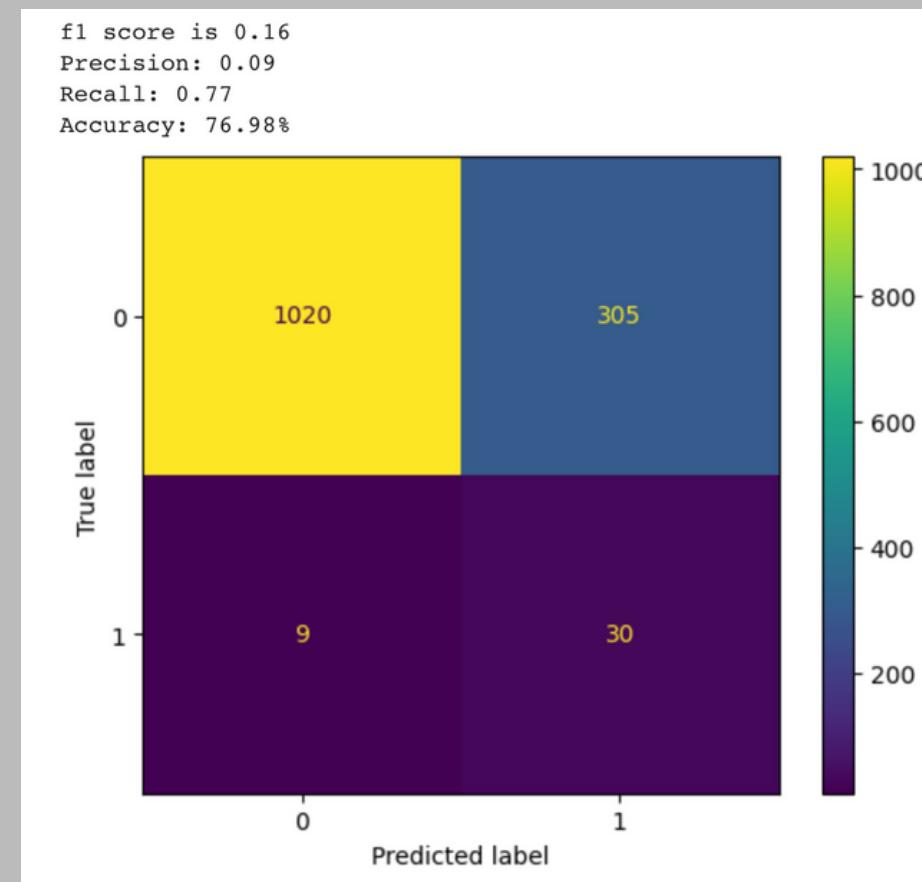
LogisticRegression



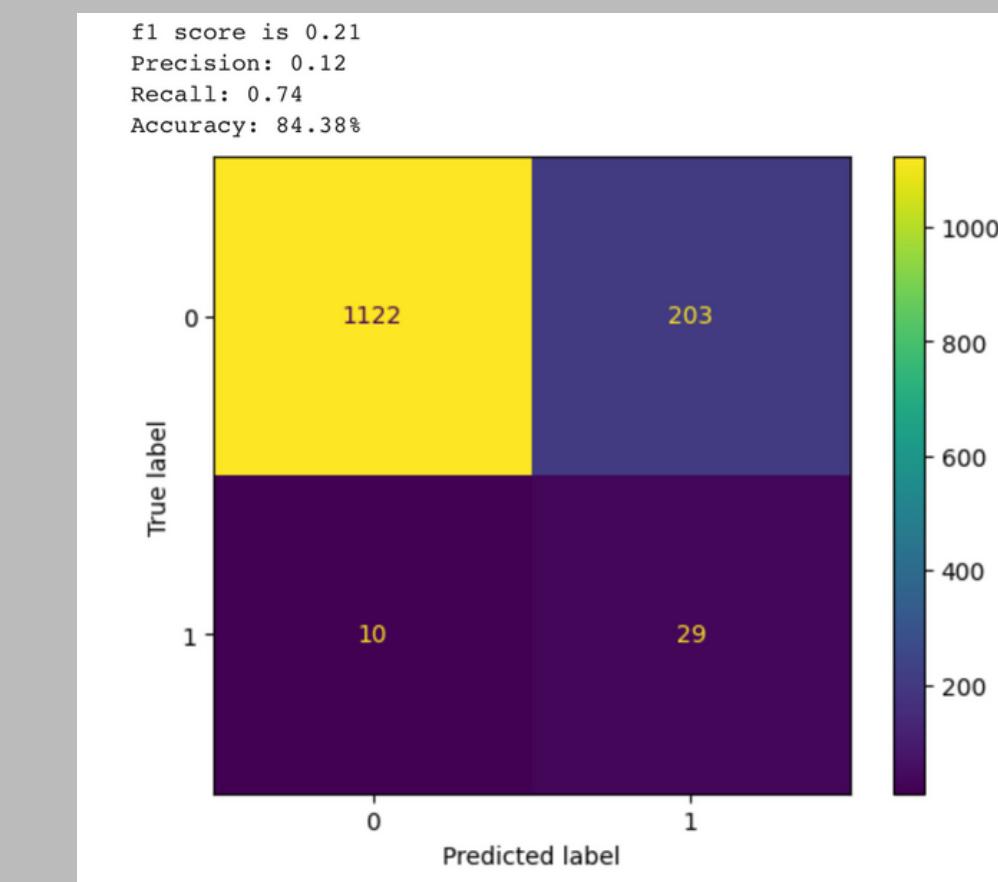
DecisionTreeClassifier



RandomForestClassifier

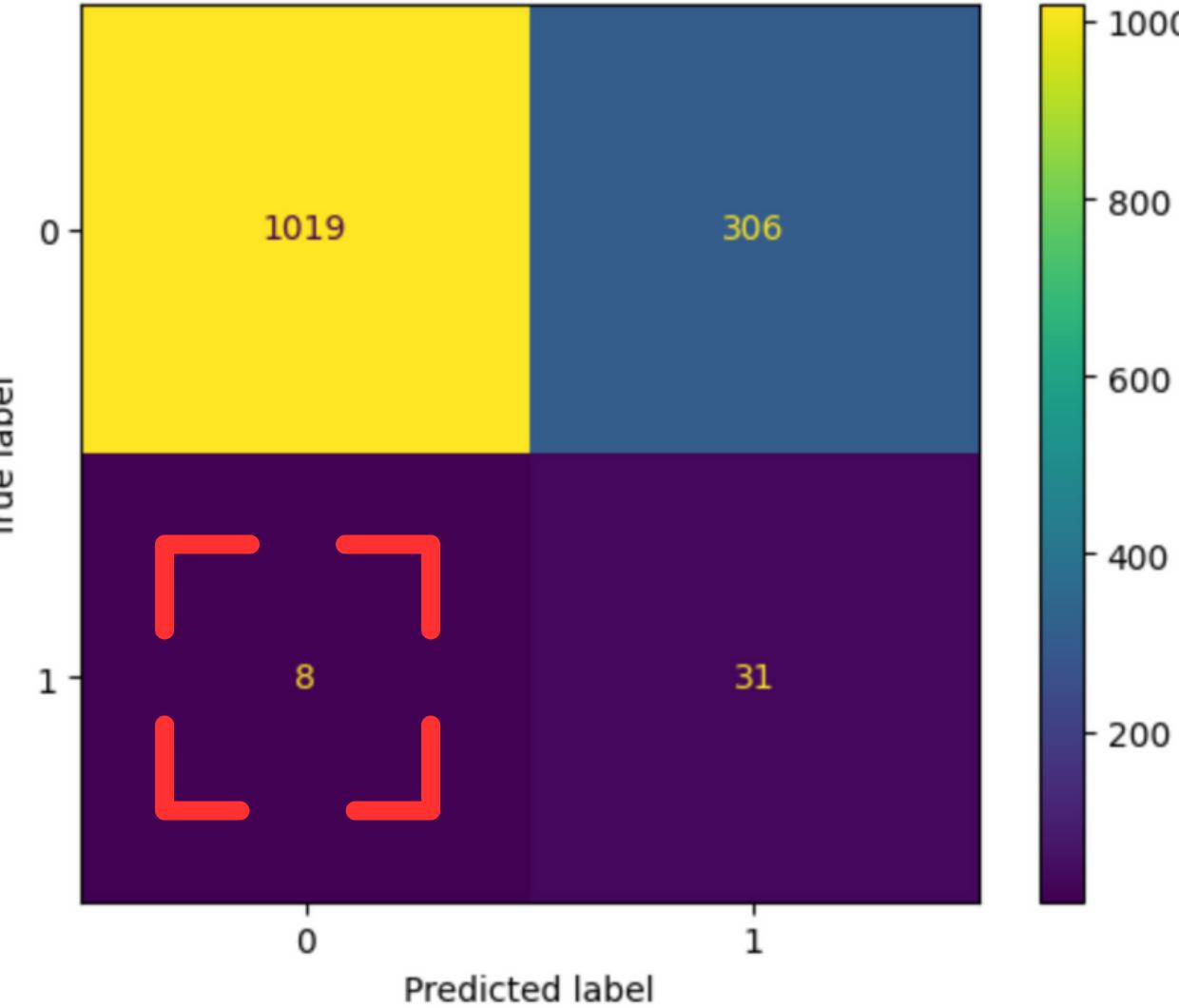


XGBoostClassifier



Support Vector Classification

Recall: 0.79
Accuracy: 76.98%



LogisticRegression

Thank You
