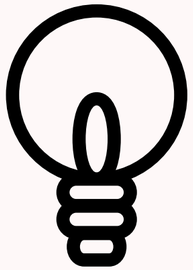
รู้แล้วต้องช็อค !!!

เมื่อบริษัทของคุณกำลังจะล้มละลาย...



จุดประสงค์

- เพื่อทำนายว่าบริษัทกำลังจะล้มละลายหรือไม่
- เพื่อหาโมเดลที่ดีที่สุดในการทำ Binary Classification
- เป็นตัวช่วยในการประกอบตัดสินใจของนักลงทุนที่จะลงทุนบริษัทนั้นๆ

แหล่งข้อมูลที่ใช้

- Data set ของ UCI ชื่อ Polish companies bankruptcy data Data Set
- Data set ของ Kaggle ชื่อ Company Bankruptcy Prediction

การนำเข้าข้อมูล

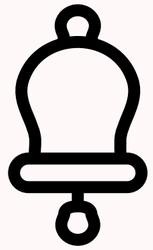
UCI

ข้อมูลของ UCL นั้นสามารถโหลดจากเว็บไซต์ได้เลย

KAGGLE

ข้อมูลของ Kaggle นั้นได้ทำการดึงข้อมูลผ่าน API

ข้อมูลใน DATA SET



- ตัวแปร Independent variable (X) เก็บข้อมูลอัตราส่วนทางการเงิน
- ตัวแปร Target (y) จะบอกว่าบริษัทนั้นล้มละลายหรือไม่

UCI

ข้อมูลมีทั้งหมด 10503 ROWS 65 COLUMNS
ประกอบด้วย
ตัวแปร X 64 COLUMNS ตัวแปร Y 1 COLUMN

ตัวอย่างข้อมูลใน Data set

- Attr1: net profit / total assets
- Attr2: total liabilities / total assets
- Attr3: working capital / total assets
- Attr4: current assets / short-term liabilities
- Attr6: retained earnings / total assets
- Attr7: EBIT / total assets

  ...
- Class

KAGGLE

ข้อมูลมีทั้งหมด 6819 ROWS 96 COLUMNS
ประกอบด้วย
ตัวแปร X 95 COLUMNS ตัวแปร Y 1 COLUMN

ตัวอย่างข้อมูลใน Data set

- ROA(C)
- ROA(A)
- ROA(B)
- Operating Gross Margin
- Realized Sales Gross Marginperating

  ...

- Bankrupt

วิเคราะห์เชิงสถิติ

UCI

	count	mean	std	min	25%	50%	75%	max
Attr1	10503.0	0.052844	0.647797	-1.769200e+01	0.000686	0.043034	0.123805	52.652
Attr2	10503.0	0.619911	6.427041	0.000000e+00	0.253955	0.464140	0.689330	480.730
Attr3	10503.0	0.095490	6.420056	-4.797300e+02	0.017461	0.198560	0.419545	17.708
Attr4	10485.0	9.980499	523.691951	2.080200e-03	1.040100	1.605600	2.959500	53433.000
Attr5	10478.0	-1347.662372	118580.569222	-1.190300e+07	-52.070750	1.579300	56.084000	685440.000
...	...	...	...	...	...	...	...	...
Attr60	9911.0	571.336309	37159.672255	0.000000e+00	5.533150	9.952100	20.936000	3660200.000
Attr61	10486.0	13.935361	83.704103	-6.590300e+00	4.486075	6.677300	10.587500	4470.400
Attr62	10460.0	135.536989	25991.162023	-2.336500e+06	40.737000	70.664000	118.220000	1073500.000
Attr63	10485.0	9.095149	31.419096	-1.562200e-04	3.062800	5.139200	8.882600	1974.500
Attr64	10275.0	35.766800	428.298315	-1.023000e-04	2.023350	4.059300	9.682750	21499.000

64 rows × 8 columns

dtypes: float64(64), object(1)

```
b'0'    10008
b'1'      495
Name: class,
```

KAGGLE

	count	mean	std	min	25%	50%	75%	max
Bankrupt?	6819.0	0.032263	0.176710	0.0	0.000000	0.000000	0.000000	1.0
ROA(C) before interest and depreciation before interest	6819.0	0.505180	0.060686	0.0	0.476527	0.502706	0.535563	1.0
ROA(A) before interest and % after tax	6819.0	0.558625	0.065620	0.0	0.535543	0.559802	0.589157	1.0
ROA(B) before interest and depreciation after tax	6819.0	0.553589	0.061595	0.0	0.527277	0.552278	0.584105	1.0
Operating Gross Margin	6819.0	0.607948	0.016934	0.0	0.600445	0.605997	0.613914	1.0
...	...	...	...	...	...	...	...	...
Liability to Equity	6819.0	0.280365	0.014463	0.0	0.276944	0.278778	0.281449	1.0
Degree of Financial Leverage (DFL)	6819.0	0.027541	0.015668	0.0	0.026791	0.026808	0.026913	1.0
Interest Coverage Ratio (Interest expense to EBIT)	6819.0	0.565358	0.013214	0.0	0.565158	0.565252	0.565725	1.0
Net Income Flag	6819.0	1.000000	0.000000	1.0	1.000000	1.000000	1.000000	1.0
Equity to Liability	6819.0	0.047578	0.050014	0.0	0.024477	0.033798	0.052838	1.0

96 rows × 8 columns

dtypes: float64(93), int64(3)

```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

1. Bankrupt?,
2. Liability-Assets Flag,
3. Net Income Flag

DATA PREPARATION

CLEAN DATA

- ลบค่า Null
- ลบค่า Outlier
- ลบแถวที่มีค่าซ้ำกัน

FEATURE SELECTION

- Multicollinearity
- RFE

IMBALANCED DATA

- SMOTE

DATA STANDARDIZATION

- StandardScaler

FEATURE SELECTION

- **Multicollinearity**

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	Attr11	Attr12	Attr13	Attr14	Attr15	Attr16	Attr17
Attr1	1.0000	0.3193	0.3622	0.1114	0.0175	0.2343	0.9881	0.0353	0.1303	0.3233	0.9489	0.3889	0.3977	0.9881	0.0109	0.4269	0.0354
Attr2	0.3193	1.0000	0.6798	0.2545	0.0369	0.6090	0.3175	0.1421	0.0788	0.9940	0.2620	0.1408	0.1715	0.3175	0.0082	0.2152	0.1445
Attr3	0.3622	0.6798	1.0000	0.3738	0.0539	0.3548	0.3671	0.1030	0.0324	0.6844	0.3270	0.1422	0.1437	0.3671	0.0021	0.2190	0.1037
Attr4	0.1114	0.2545	0.3738	1.0000	0.0206	0.0482	0.1148	0.4390	0.0932	0.2572	0.1000	0.3334	0.1347	0.1148	0.0222	0.4125	0.4407
Attr5	0.0175	0.0369	0.0539	0.0206	1.0000	0.0241	0.0173	0.0087	0.0260	0.0363	0.0179	0.0230	0.0040	0.0173	0.0049	0.0179	0.0089
Attr6	0.2343	0.6090	0.3548	0.0482	0.0241	1.0000	0.2345	0.0335	0.0786	0.6006	0.1973	0.0710	0.1485	0.2345	0.0235	0.1039	0.0350
Attr7	0.9881	0.3175	0.3671	0.1148	0.0173	0.2345	1.0000	0.0356	0.1376	0.3212	0.9634	0.3961	0.3976	1.0000	0.0070	0.4364	0.0357
Attr8	0.0353	0.1421	0.1030	0.4390	0.0087	0.0335	0.0356	1.0000	0.0653	0.1430	0.0257	0.1372	0.0451	0.0356	0.0133	0.2468	0.9994
Attr9	0.1303	0.0788	0.0324	0.0932	0.0260	0.0786	0.1376	0.0653	1.0000	0.0602	0.1820	0.0214	0.0103	0.1376	0.0071	0.0294	0.0674
Attr10	0.3233	0.9940	0.6844	0.2572	0.0363	0.6006	0.3212	0.1430	0.0602	1.0000	0.2691	0.1423	0.1710	0.3212	0.0107	0.2160	0.1436
Attr11	0.9489	0.2620	0.3270	0.1000	0.0179	0.1973	0.9634	0.0257	0.1820	0.2691	1.0000	0.3845	0.3674	0.9634	0.0061	0.4212	0.0254
Attr12	0.3889	0.1408	0.1422	0.3334	0.0230	0.0710	0.3961	0.1372	0.0214	0.1423	0.3845	1.0000	0.4211	0.3961	0.0125	0.6475	0.1384
Attr13	0.3977	0.1715	0.1437	0.1347	0.0040	0.1485	0.3976	0.0451	0.0103	0.1710	0.3674	0.4211	1.0000	0.3976	0.0106	0.3617	0.0461
Attr14	0.9881	0.3175	0.3671	0.1148	0.0173	0.2345	1.0000	0.0356	0.1376	0.3212	0.9634	0.3961	0.3976	1.0000	0.0070	0.4364	0.0357
Attr15	0.0109	0.0082	0.0021	0.0222	0.0049	0.0235	0.0070	0.0133	0.0071	0.0107	0.0061	0.0125	0.0106	0.0070	1.0000	0.0260	0.0136
Attr16	0.4269	0.2152	0.2190	0.4125	0.0179	0.1039	0.4364	0.2468	0.0294	0.2160	0.4212	0.6475	0.3617	0.4364	0.0260	1.0000	0.2493
Attr17	0.0354	0.1445	0.1037	0.4407	0.0089	0.0350	0.0357	0.9994	0.0674	0.1436	0.0254	0.1384	0.0461	0.0357	0.0136	0.2493	1.0000
Attr18	0.9881	0.3175	0.3671	0.1148	0.0173	0.2345	1.0000	0.0356	0.1376	0.3212	0.9634	0.3961	0.3976	1.0000	0.0070	0.4364	0.0357
Attr19	0.4160	0.1723	0.1563	0.1213	0.0113	0.1614	0.4180	0.0282	0.0369	0.1720	0.3886	0.4223	0.9694	0.4180	0.0183	0.3214	0.0289

Pearson correlation

Attribute ที่ไม่ได้ใช้งาน

- Attr1
- Attr2
- Attr7
- Attr11
- Attr8
- Attr14
- Attr13
- Attr18
- Attr10
- Attr16
- Attr19
- Attr22
- Attr23
- Attr43
- Attr4
- Attr40
- Attr42
- Attr38
- Attr32
- Attr28
- Attr53
- Attr33

FEATURE SELECTION

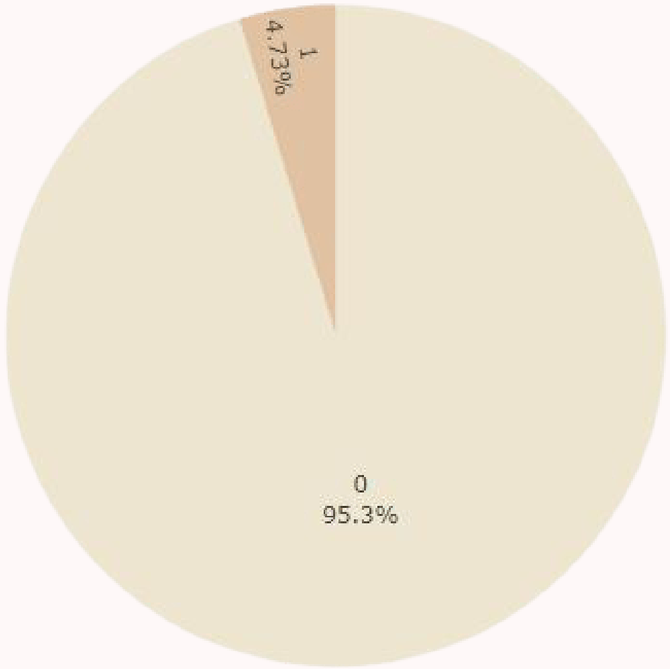
- **RFE**

```
def feature_selec_rfe(X, y):
  '''Function นี้ไว้ทำการเลือก Attribute ที่ดีที่สุดจากค่า Parameter ที่กำหนดไว้ โดยใช้โมเดล RandomForest'''
  #ทำการแบ่งข้อมูลไว้เพื่อใช้ในการ Train และ Test
  X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                      stratify = y,
                                                      test_size = 0.2)
  #สร้างโมเดล RandomForest เพื่อใช้ในการทำนาย
  estimator = RandomForestClassifier(n_estimators=100, random_state=42)
  #เรียกใช้ RFE และทำการกำหนดค่า n_features
  selector = RFE(estimator, n_features_to_select=10)
  selector = selector.fit(X_train, y_train.values.ravel())

  temp = pd.Series(selector.support_,index = X.columns)
  selected_features_rfe = temp[temp==True].index

  X = X.drop(X.columns.difference(selected_features_rfe), axis=1)
  return X
```

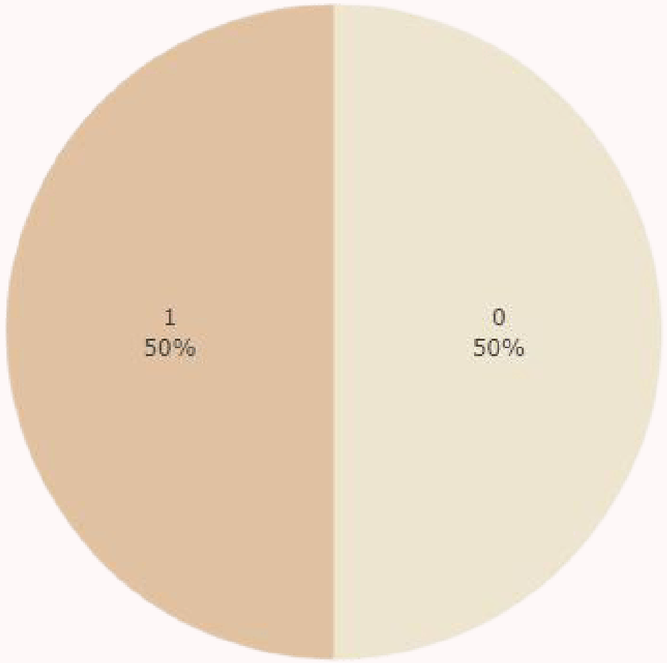
```
X = feature_selec_rfe(X, y)
```

- Attr9
- Attr24
- Attr26
- Attr27
- Attr29
- Attr34
- Attr35
- Attr46
- Attr56
- Attr58

- **SMOTE**

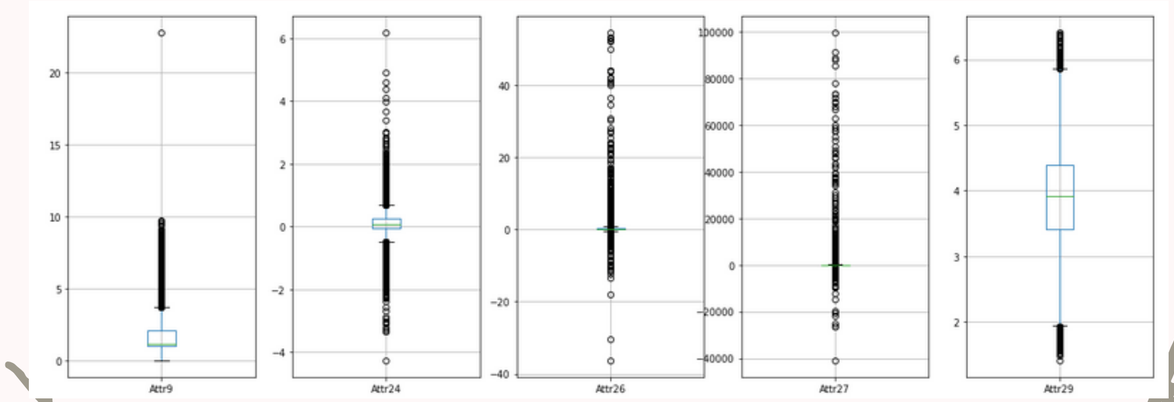


Imbalanced Dataset



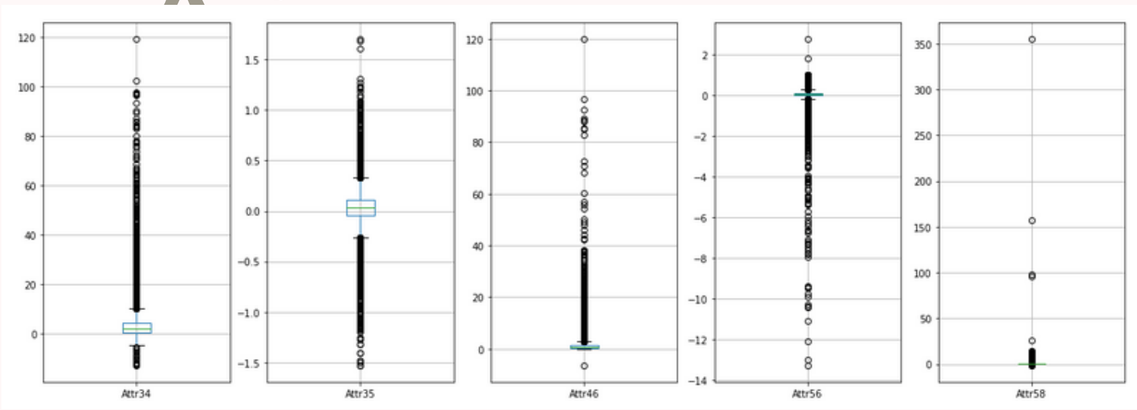
SMOTE Dataset

- **Data Standardization**




```
def sd_scaler(X):
  '''Function นี้ทำการ Normalize ข้อมูลด้วย StandardScaler'''
  scaler = StandardScaler()
  scaled = scaler.fit_transform(X)
  return X
```

```
X = sd_scaler(X)
```



Boxplots

วิธีการทำการทดลอง

MODEL ที่ในการทำนาย

- Decision Tree
- K-Nearest Neighbor
- Logistic Regression
- Neural network
- Random Forest
- Gradient Boosted Trees
- XGBoost Classifier
- AdaBoost Classifier

วัดประสิทธิภาพ MODEL

- Accuracy
- ROC Curve
- Confusion matrix
- Classification report

HYPERPARAMETER OPTIMIZATION

- RandomizedSearchCV

FEATURE SELECTION

แบ่งข้อมูล

```
1 X.shape
```

```
(18906, 10)
```

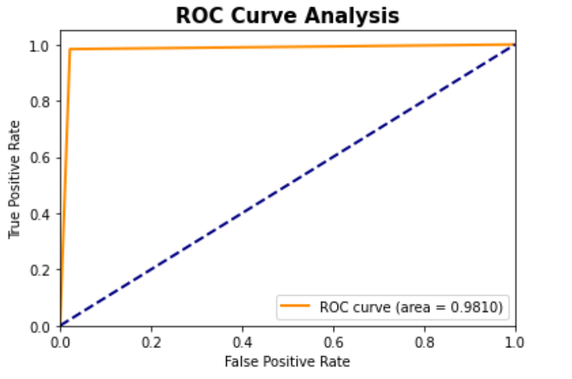
```
1 X_train, X_test, y_train, y_test = train_test_split(X,y,
2                                                     stratify = y,
3                                                     test_size = 0.2)
```

RANDOM FOREST

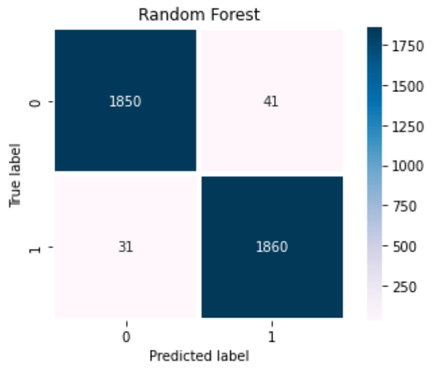
```
1 rdf = RandomForestClassifier()
2 rdf.fit(X_train, y_train.values.ravel())
3 y_pred = rdf.predict(X_test)
```

```
1 train_test_score(rdf, X_train, y_train, y_test)
```

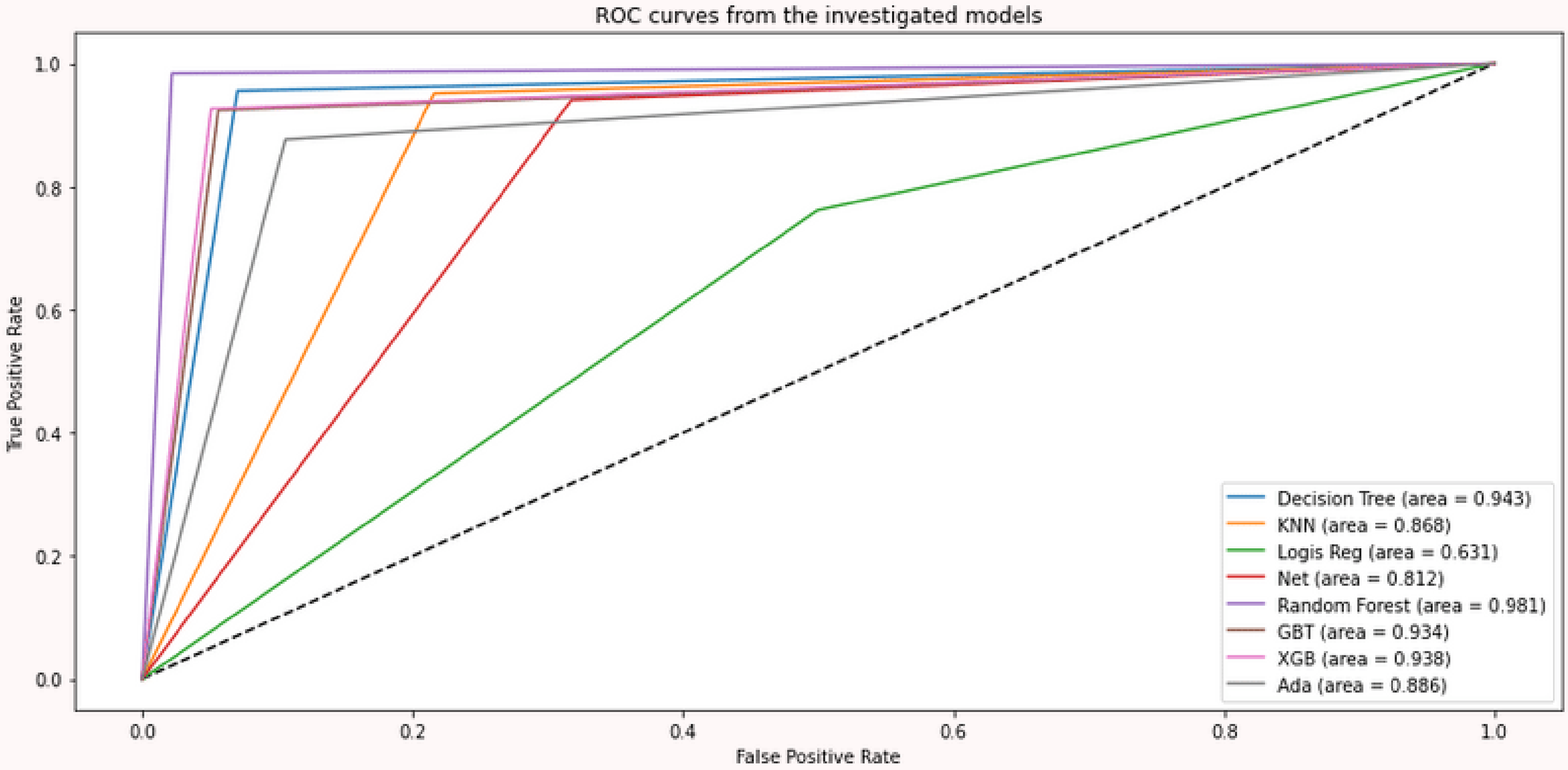
Training Accuracy: 1.0
Test Accuracy: 0.981
Bias: 0.0
Variance: 0.019



	precision	recall	f1-score	support
0	0.98	0.98	0.98	1891
1	0.98	0.98	0.98	1891
accuracy			0.98	3782
macro avg	0.98	0.98	0.98	3782
weighted avg	0.98	0.98	0.98	3782



ROC curve comparison



RANDOM FOREST

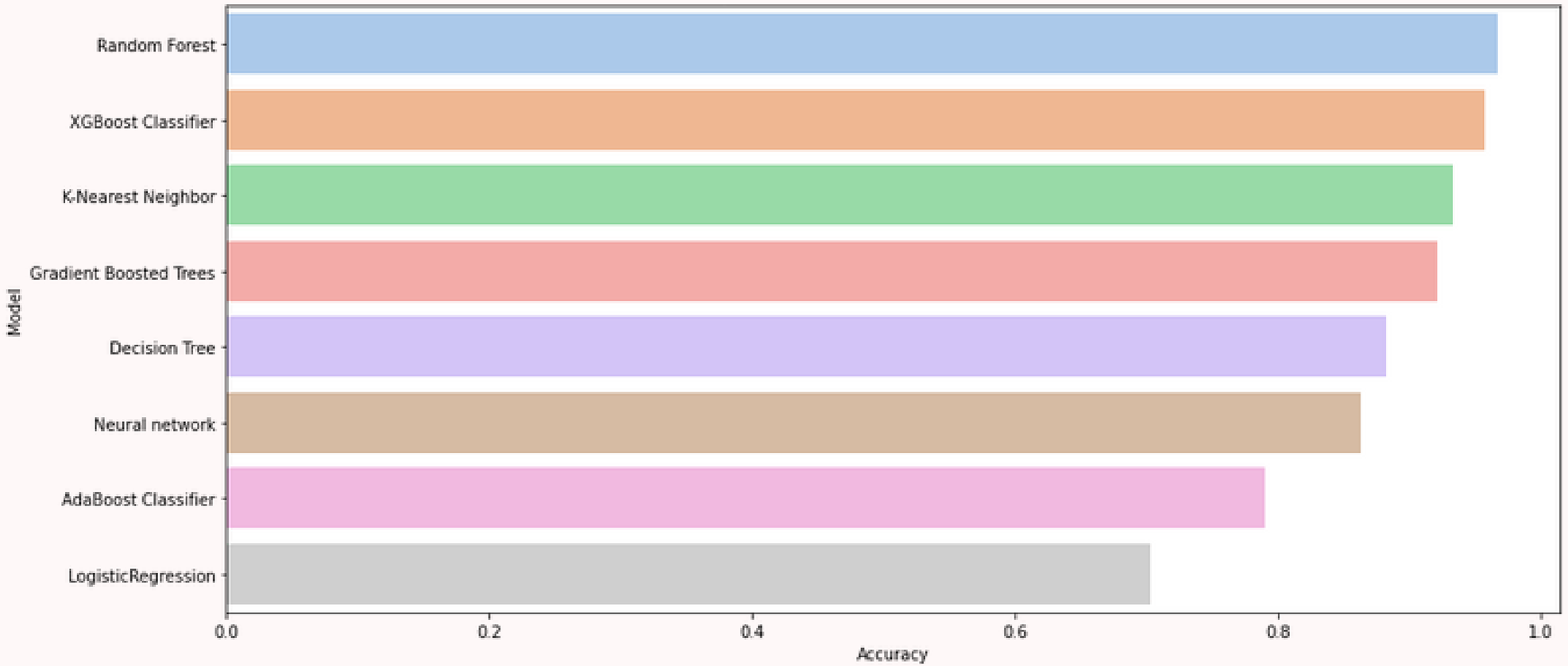
```
1 model.fit(X, y.values.ravel())
2 acc_score = cross_val_score(model, X, y, cv=5)
3 print(acc_score)
4 print('Mean cross-validation accuracy: %.3f (%.3f)' %(mean(acc_score), std(acc_score)))
```

```
[0.96774194 0.96905581 0.96693996 0.95900555 0.97064269]
Mean cross-validation accuracy: 0.967 (0.004)
```

```
1 est = RandomForestClassifier()
2 param = {'max_depth':[3,5,10,None],
3          'n_estimators':[100,200,300,400,500],
4          'max_features':[1,2,3,4,5],
5          'criterion':['gini', 'entropy'],
6          'bootstrap':[True,False],
7          'min_samples_leaf':[1,2,3,4,5]
8          }
9 model = hypertuning(est,param, X, y.values.ravel())
```

```
best estimator : RandomForestClassifier(bootstrap=False, max_features=5, min_samples_leaf=2,
                                        n_estimators=300)
Best parameters : {'n_estimators': 300, 'min_samples_leaf': 2, 'max_features': 5, 'max_depth': None, 'criterion': 'gini', 'bootstrap': False}
```

RANKING MODEL WITH FEATURE SELECTION



NON FEATURE SELECTION

แบ่งข้อมูล

```
1 X_copy.shape
```

```
(18906, 64)
```

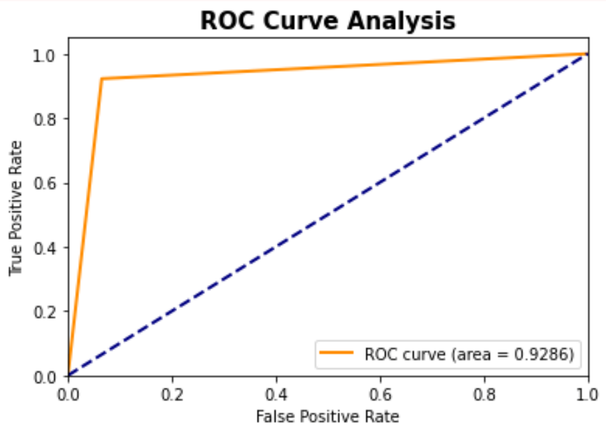
```
1 X_train, X_test, y_train, y_test = train_test_split(X_copy,
2                                                     y_copy,
3                                                     stratify = y_copy,
4                                                     test_size = 0.2)
```

XGBOOST

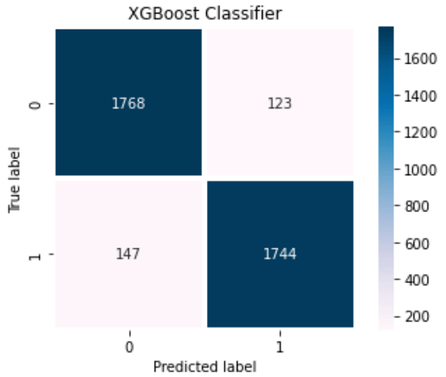
```
1 xgb = XGBClassifier()
2 xgb.fit(X_train, y_train.values.ravel())
3 y_pred = xgb.predict(X_test)
```

```
1 train_test_score(xgb, X_train, y_train, y_test)
```

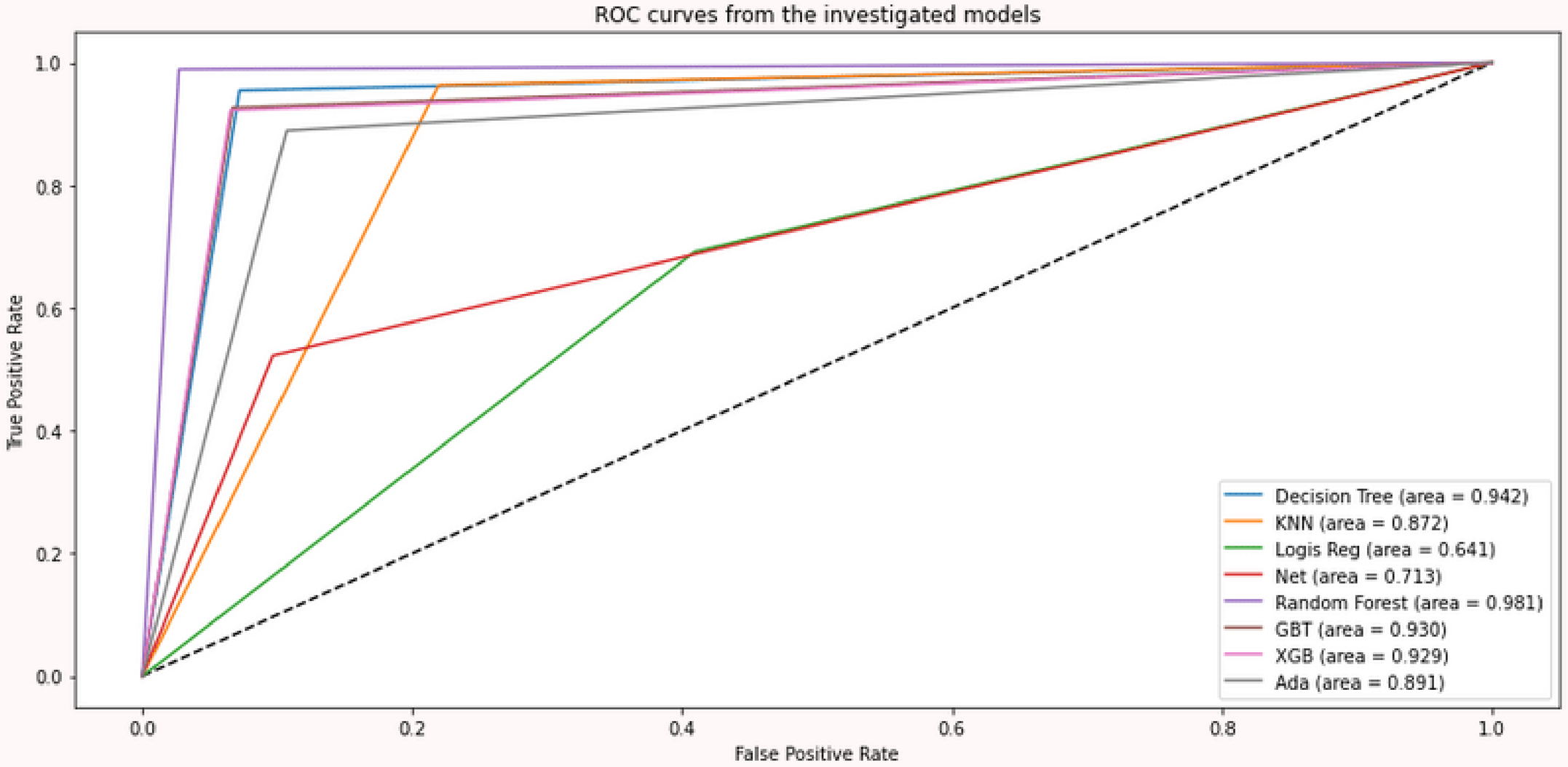
```
Training Accuracy: 0.948
Test Accuracy: 0.929
Bias: 0.052
Variance: 0.019
```



	precision	recall	f1-score	support
0	0.92	0.93	0.93	1891
1	0.93	0.92	0.93	1891
accuracy			0.93	3782
macro avg	0.93	0.93	0.93	3782
weighted avg	0.93	0.93	0.93	3782



ROC curve comparison



XGBOOST

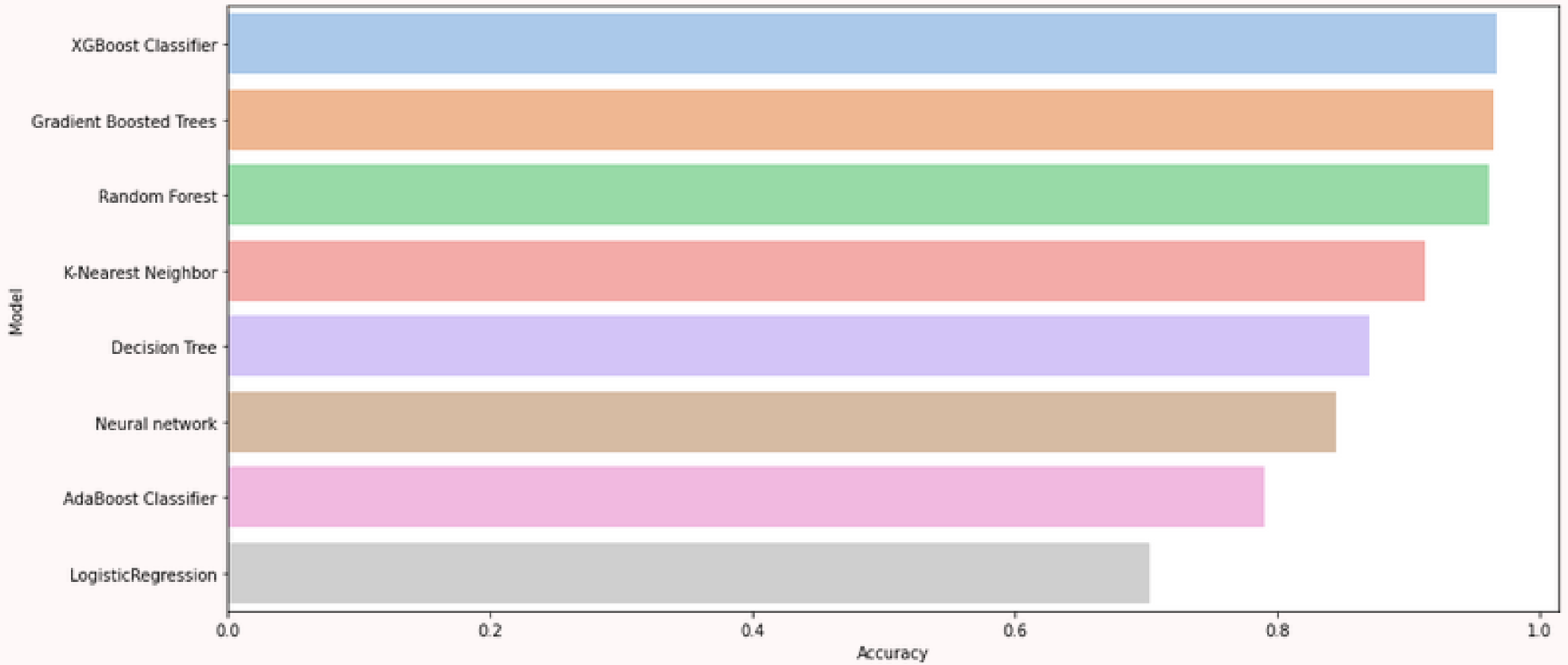
```
est = XGBClassifier()
param = {'learning_rate':[0.001,0.01,0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 1, 10],
         'max_depth':[1,3,5,9,15,20,50,100],
          "min_child_weight":[ 1, 3, 5, 7 ],
         "gamma":[ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
         "colsample_bytree":[ 0.3, 0.4, 0.5 , 0.7 ] }
model = hypertuning(est,param, X, y.values.ravel())
```

```
best estimator : XGBClassifier(colsample_bytree=0.7, gamma=0.3, learning_rate=0.25, max_depth=20)
Best parameters : {'min_child_weight': 1, 'max_depth': 20, 'learning_rate': 0.25, 'gamma': 0.3, 'colsample_bytree': 0.7}
```

```
model.fit(X_copy, y_copy.values.ravel())
acc_score = cross_val_score(model, X, y, cv=5)
print(acc_score)
print('Mean cross-validation accuracy: %.3f (%.3f)' %(mean(acc_score), std(acc_score)))
```

```
[0.97329455 0.97037821 0.966411   0.95583179 0.96826236]
Mean cross-validation accuracy: 0.967 (0.006)
```

RANKING MODEL WITHOUT FEATURE SELECTION



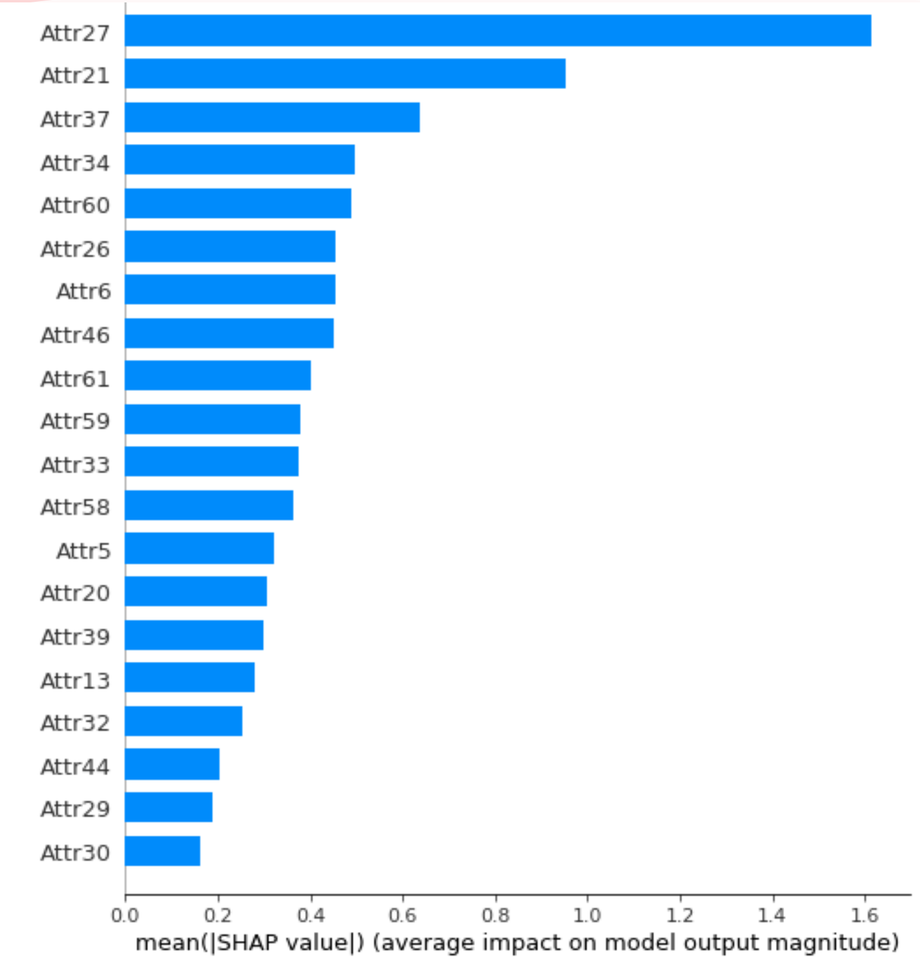
SHAP (SHapley Additive exPlanations)

```
model = XGBClassifier(min_child_weight=5,
                      max_depth=50,
                      learning_rate=0.2,
                      gamma=0.3,
                      colsample_bytree=0.7)
model.fit(X_copy, y_copy.values.ravel())
```

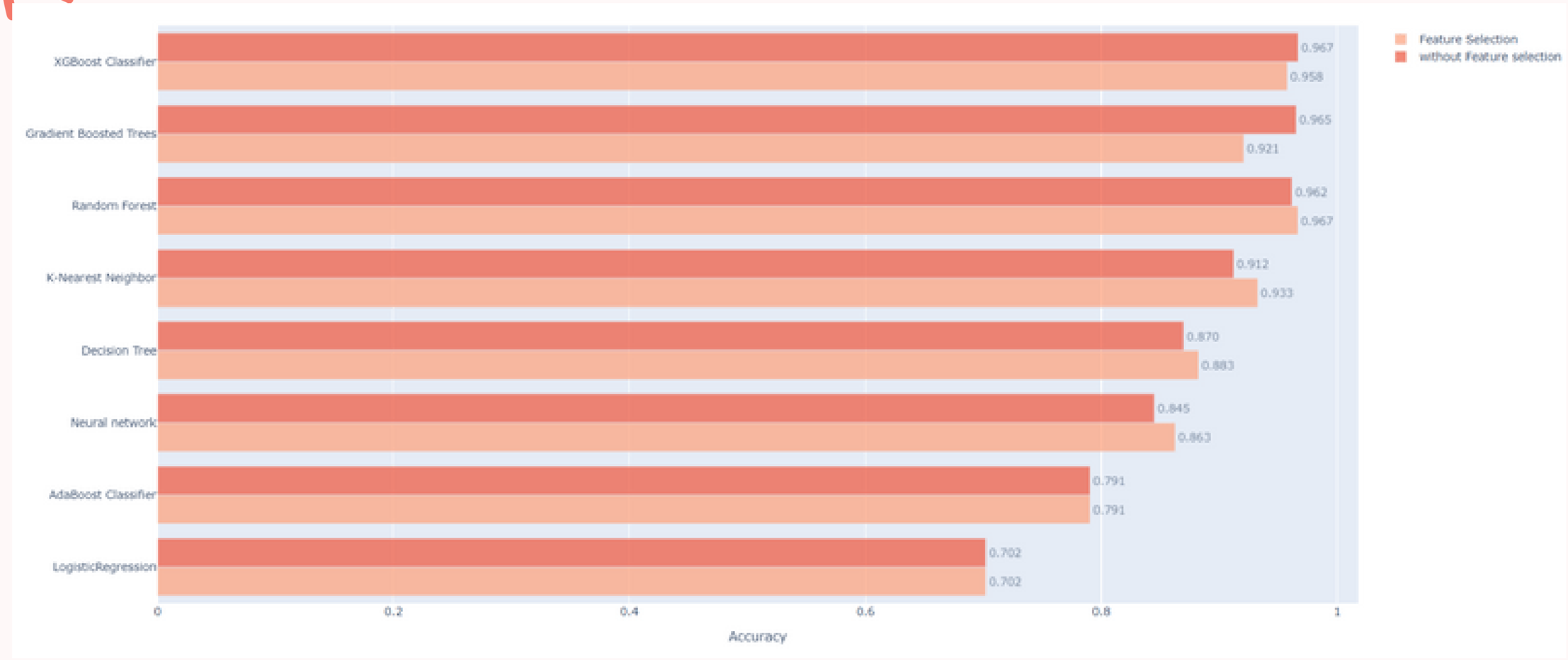
```
XGBClassifier(colsample_bytree=0.7, gamma=0.3, learning_rate=0.2, max_depth=50,
              min_child_weight=5)
```

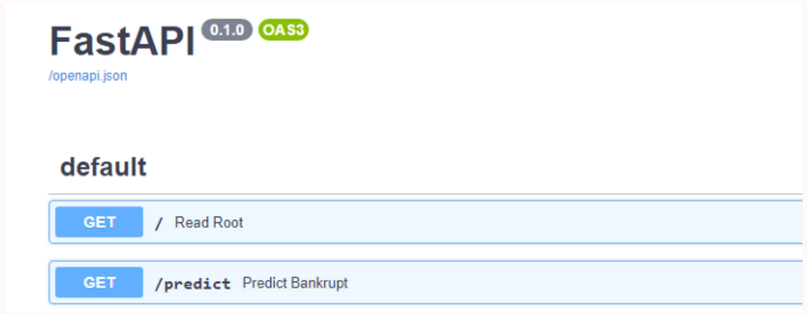
```
# SHAP Interpreter
explainer = shap.Explainer(model)
shap_values = explainer(X_copy)

```

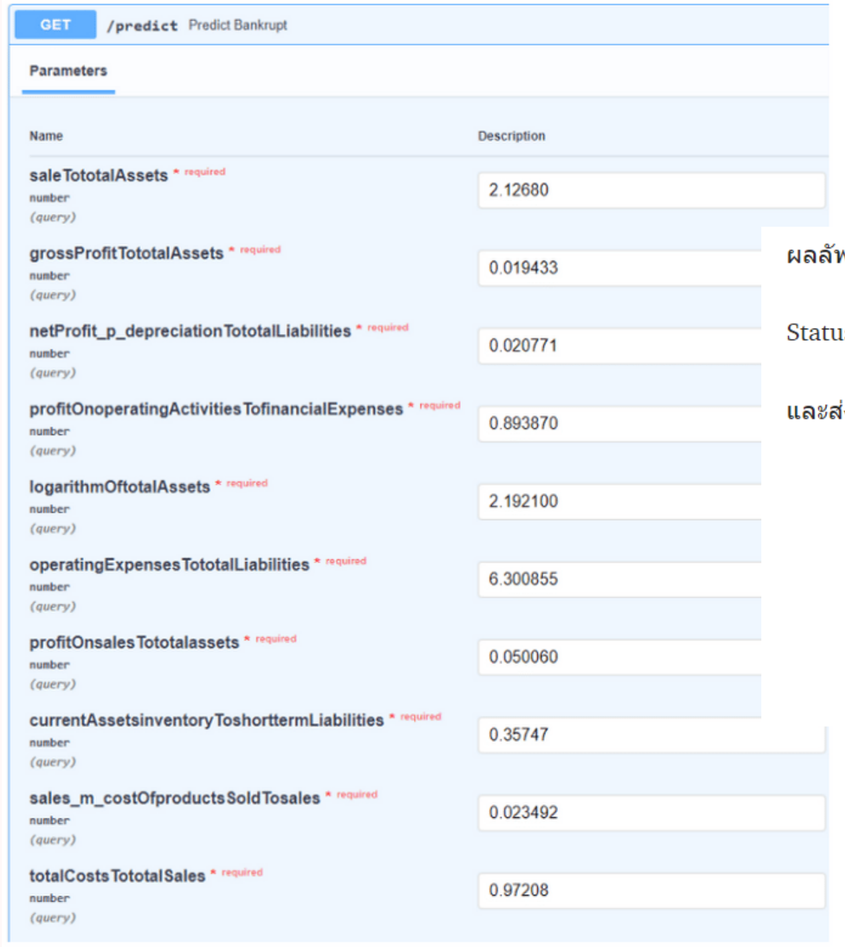
Attr27: profit on operating activities / financial expense
Attr21: sales (n) / sales (n-1)
Attr37: (current assets - inventories) / long-term liabilities



FEATURE SELECTION
VS
WITHOUT FEATURE SELECTION







ผลลัพธ์

Status Code : 200 เป็นรหัสตอบรับมาตรฐานสำหรับการร้องขอที่สำเร็จ

และส่ง Response มาว่าบริษัทนั้นล้มละลาย

สรุปผล

แนวทางการเพิ่มประสิทธิภาพ

- เราสามารถทำนายบริษัทนั้นจะล้มละลายหรือไม่ล้มละลาย ได้โดยโมเดล Random Forest

- หาข้อมูลเพิ่มเติมเพื่อลด Imbalanced data จะทำให้โมเดลมีประสิทธิภาพมากขึ้น

ขอบคุณ!