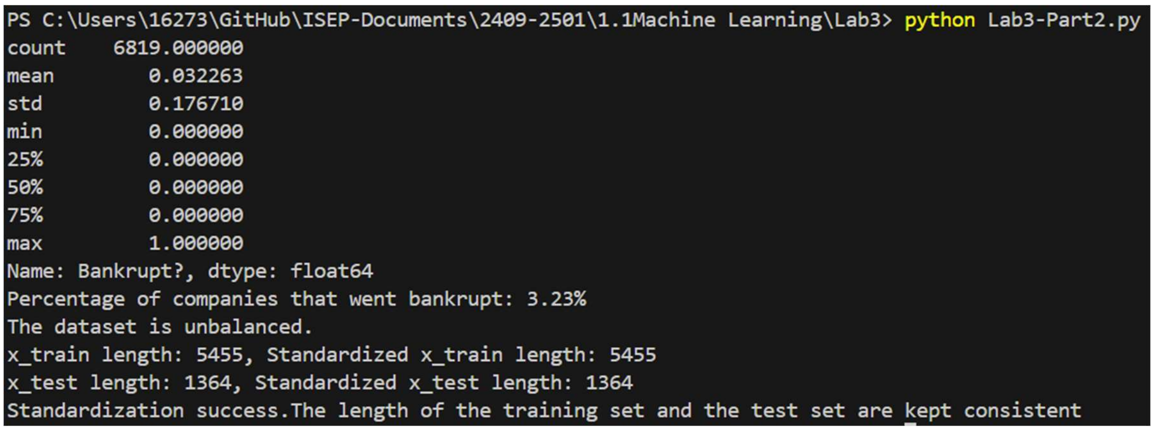
ISEP
Machine Learning
October 7th 2024

Lab 3 : Decision trees and Support Vector Machines (SVM)

GUO Xiaofan, FU Jintao

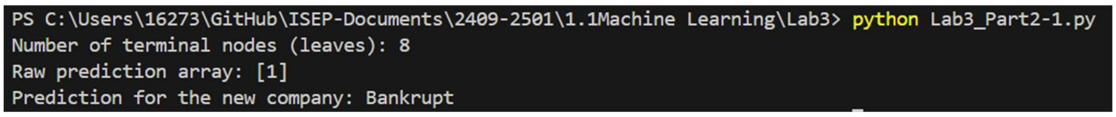
2. Part II: Practical applications

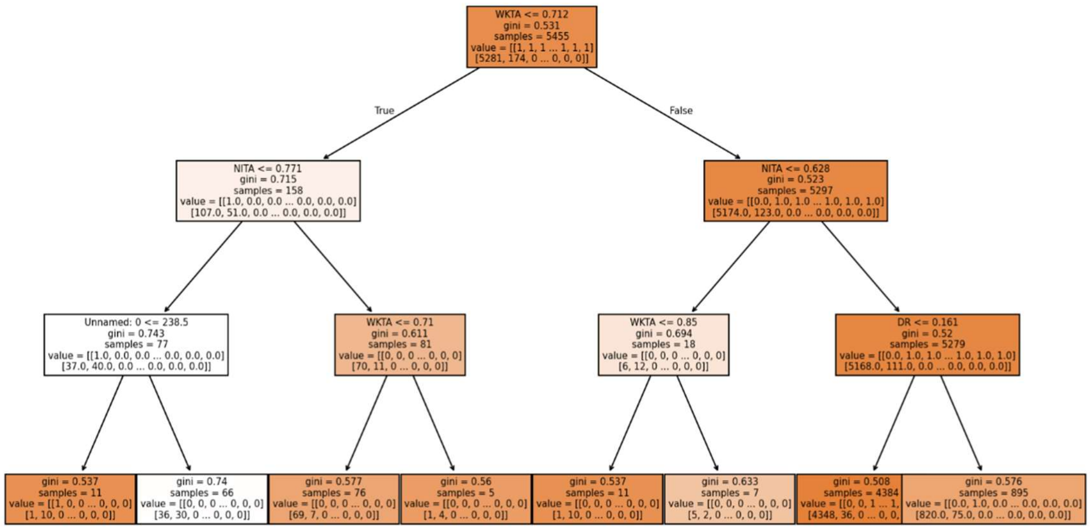

```
PS C:\Users\16273\GitHub\ISEP-Documents\2409-2501\1.1Machine Learning\Lab3> python Lab3-Part2.py
count    6819.000000
mean        0.032263
std         0.176710
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: Bankrupt?, dtype: float64
Percentage of companies that went bankrupt: 3.23%
The dataset is unbalanced.
x_train length: 5455, Standardized x_train length: 5455
x_test length: 1364, Standardized x_test length: 1364
Standardization success.The length of the training set and the test set are kept consistent
```

1. Percentage of companies that went bankrupt: **3.23%**.
2. The dataset is **unbalanced**.
3. The length of the training set and the test set are **kept consistent**: x_train length is 5455, x_test length is 1364. Standardization success.

2.1 Classification Trees

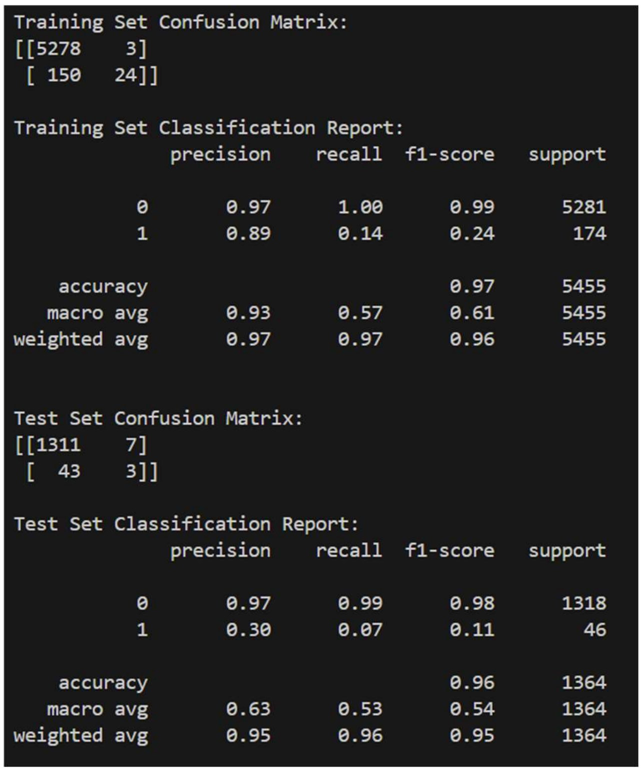
1.


```
PS C:\Users\16273\GitHub\ISEP-Documents\2409-2501\1.1Machine Learning\Lab3> python Lab3_Part2-1.py
Number of terminal nodes (leaves): 8
Raw prediction array: [1]
Prediction for the new company: Bankrupt
```



- The top-level nodes contain **segmentation features, Gini index, total number of samples, and category distribution information.**
- The Gini index is used to measure node purity, and the smaller the Gini index, the clearer the classification.
- $Gini = 1 - \sum(p(i)^2)$
- The tree has **8** terminal nodes.
- The company **will go bankrupt**.

2.



```
Training Set Confusion Matrix:
[[5278    3]
 [ 150   24]]

Training Set Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      5281
           1       0.89      0.14      0.24       174

    accuracy                           0.97      5455
   macro avg       0.93      0.57      0.61      5455
weighted avg       0.97      0.97      0.96      5455


Test Set Confusion Matrix:
[[1311    7]
 [  43    3]]

Test Set Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1318
           1       0.30      0.07      0.11        46

    accuracy                           0.96      1364
   macro avg       0.63      0.53      0.54      1364
weighted avg       0.95      0.96      0.95      1364
```

Training Set:

- **Precision:** The prediction precision for bankrupt companies is 0.89, and for non-bankrupt companies is 0.97.
- **Recall:** For bankrupt companies, the recall is only 0.14 (i.e., the model only correctly identified 14% of bankrupt companies), while the recall for non-bankrupt companies is almost 1.00.
- **F1 score:** The F1 score for bankrupt companies is 0.24, which is low, indicating that the model does not have a good overall classification effect on bankrupt companies.
- **Accuracy:** The overall accuracy is 97%, indicating that the model performs well overall on the training set, but its ability to identify bankrupt companies is weak.

Test Set:

- **Precision:** The precision for bankrupt companies is low at 0.30, indicating that the model incorrectly predicts many non-bankrupt companies as bankrupt.
- **Recall:** The recall for bankrupt companies is 0.07 (only 7% of bankrupt companies are correctly predicted), indicating that the model has very poor recognition of bankrupt companies on the test set.
- **F1 score:** The F1 score for bankrupt companies is only 0.11, which is very low, indicating that the model is poor at classifying bankrupt companies.
- **Accuracy:** The overall accuracy is 96%, but the accuracy is affected by the high recall of non-bankrupt companies (which are the majority).

Overfitting exists: The model performs well on the training set, but its performance in classifying bankrupt companies on the test set drops sharply, which indicates that the model has overfit to the training data. It has learned too much detail or noise in the training data and cannot generalize to new data.

Solutions to Reduce Overfitting:

- **Reduce Model Complexity:**

  Limit Tree Depth: Decrease the ‘max_depth’ parameter of the decision tree to make the model less complex. You can try smaller values like 2 or adjust other parameters like ‘min_samples_split’ or ‘min_samples_leaf’ to control the growth of the tree.

- **Use Pruning Techniques:**

  Cost Complexity Pruning: Use the ‘ccp_alpha’ parameter to prune the decision tree after training. This will help remove parts of the tree that don't contribute much to the final prediction, reducing complexity and overfitting.

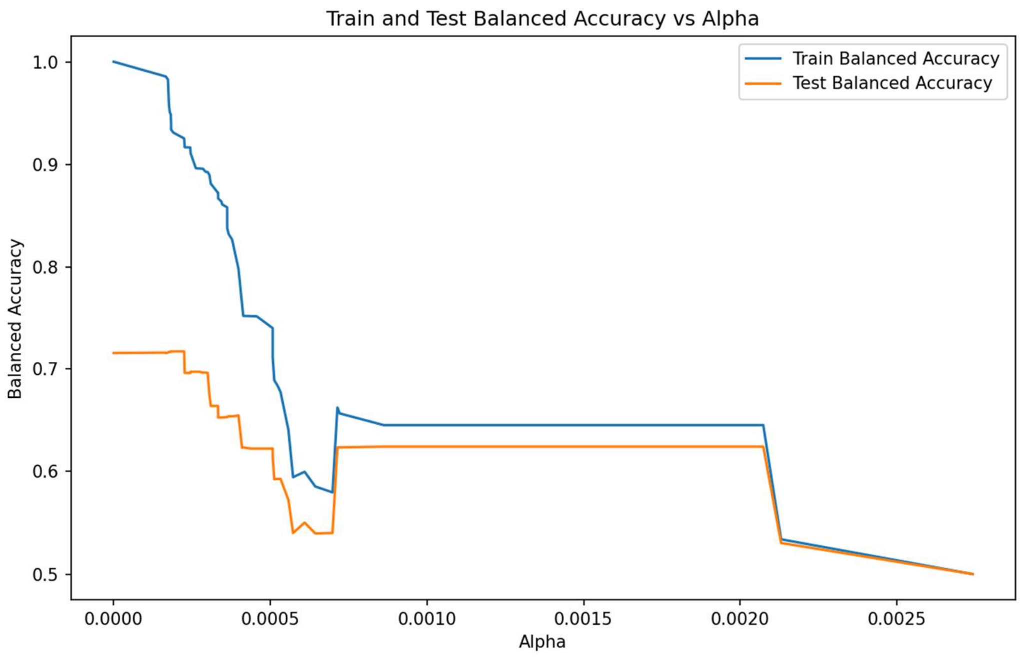
- **Cross-Validation:**

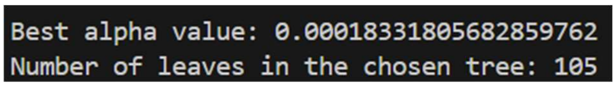
  Apply cross-validation to find the best model parameters. This ensures the model is not overfitting the training data and can generalize better to new data.

- **Increase Training Data:**

  Add more data, especially for bankrupt companies. More data can help the model learn better patterns and avoid overfitting to specific details in the training set.

c.

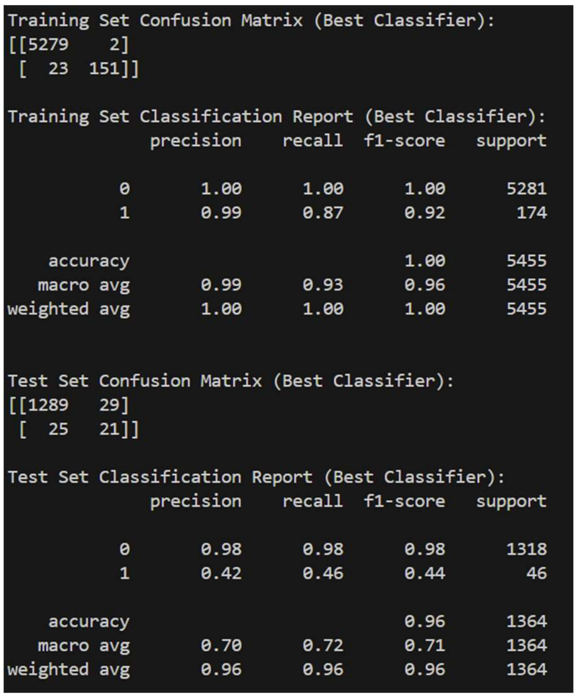




```
Best alpha value: 0.00018331805682859762
Number of leaves in the chosen tree: 105
```

- The best value of α: 0.00018331805682859762
- Number of leaves in the chosen tree: 105

d.



```
Training Set Confusion Matrix (Best Classifier):
[[5279    2]
 [  23  151]]

Training Set Classification Report (Best Classifier):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5281
           1       0.99      0.87      0.92       174

    accuracy                           1.00      5455
   macro avg       0.99      0.93      0.96      5455
weighted avg       1.00      1.00      1.00      5455


Test Set Confusion Matrix (Best Classifier):
[[1289   29]
 [  25   21]]

Test Set Classification Report (Best Classifier):
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1318
           1       0.42      0.46      0.44        46

    accuracy                           0.96      1364
   macro avg       0.70      0.72      0.71      1364
weighted avg       0.96      0.96      0.96      1364
```

Training Set:

- **Confusion Matrix:** The model correctly predicted almost all companies, with only 2 misclassified as not bankrupt and 23 as bankrupt.
- Precision, Recall, F1-score:
  - For non-bankrupt companies (class 0), the scores are perfect (1.00).
  - For bankrupt companies (class 1), Precision is 0.99, Recall is 0.87, and F1-score is 0.92. The model performs well overall on the training set.

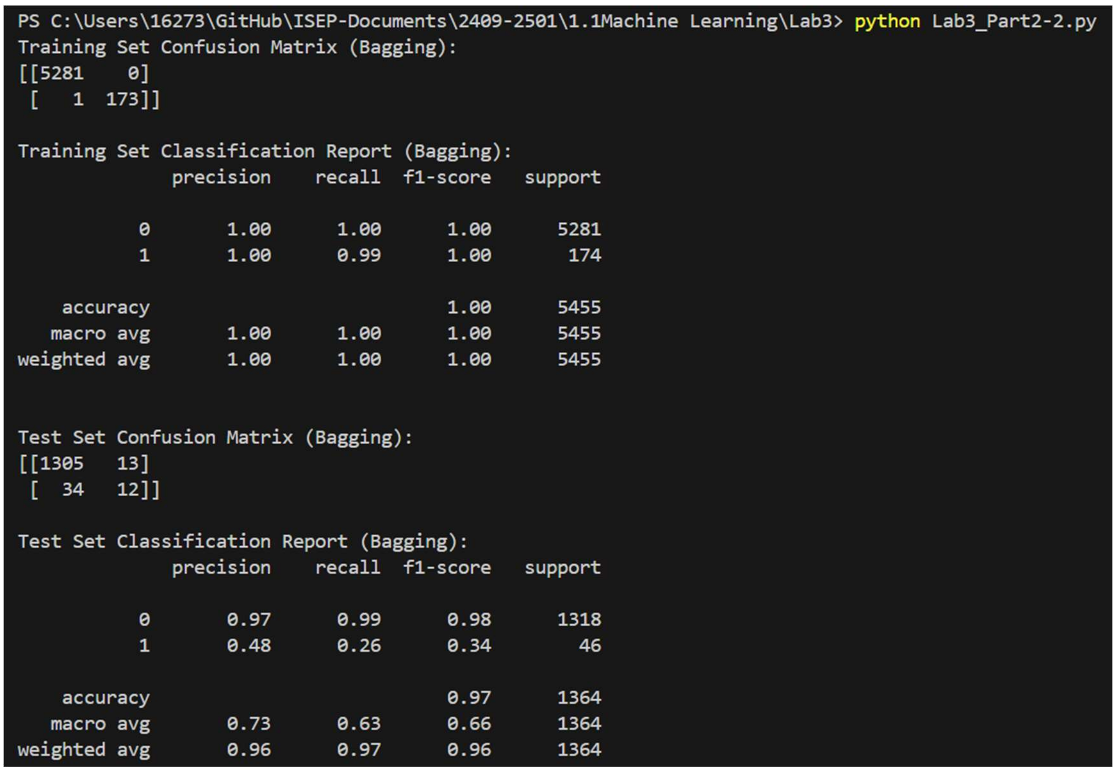
Test Set:

- **Confusion Matrix:** The model correctly predicted 1289 non-bankrupt companies but misclassified 29 as bankrupt. It correctly predicted 21 bankrupt companies but misclassified 25 as non-bankrupt.
- Precision, Recall, F1-score:
  - For non-bankrupt companies, Precision and Recall are both 0.98.
  - For bankrupt companies, Precision is 0.42, Recall is 0.46, and F1-score is 0.44. The model struggles more with predicting bankrupt companies on the test set.

The model performs well for non-bankrupt companies but has difficulty predicting bankrupt companies, especially on the test set.

2.2 Ensemble methods: Bagging, Random Forest and Boosting

1.



```
PS C:\Users\16273\GitHub\ISEP-Documents\2409-2501\1.1Machine Learning\Lab3> python Lab3_Part2-2.py
Training Set Confusion Matrix (Bagging):
[[5281    0]
 [   1  173]]

Training Set Classification Report (Bagging):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5281
           1       1.00      0.99      1.00       174

    accuracy                           1.00      5455
   macro avg       1.00      1.00      1.00      5455
weighted avg       1.00      1.00      1.00      5455


Test Set Confusion Matrix (Bagging):
[[1305   13]
 [  34   12]]

Test Set Classification Report (Bagging):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1318
           1       0.48      0.26      0.34        46

    accuracy                           0.97      1364
   macro avg       0.73      0.63      0.66      1364
weighted avg       0.96      0.97      0.96      1364
```

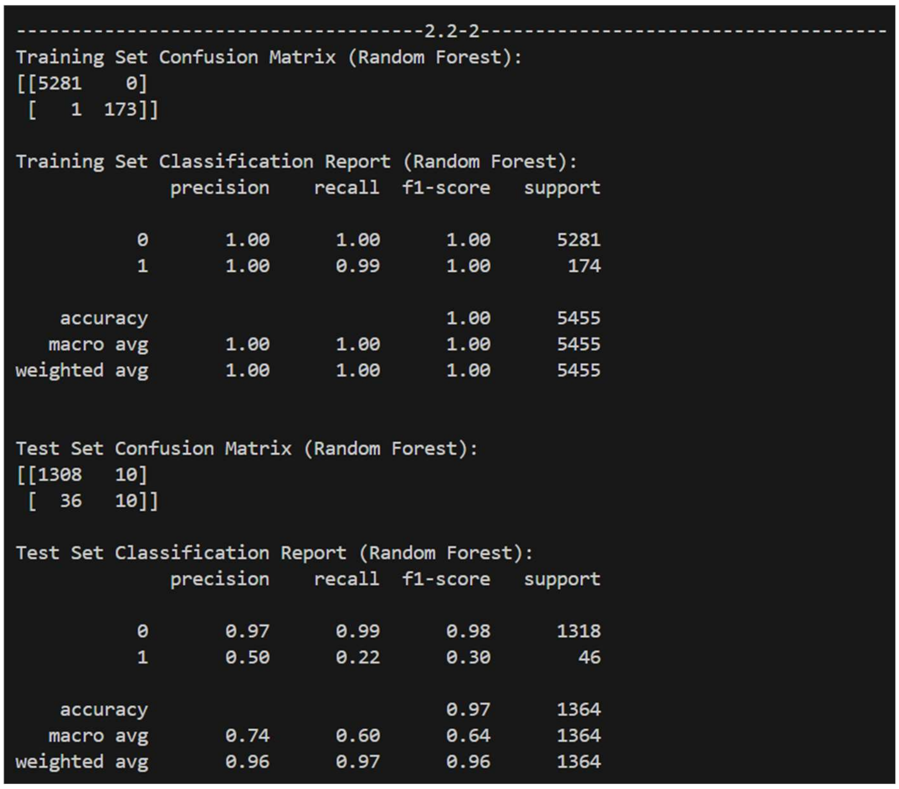
Training Set:

- The model performs almost perfectly on the training set, indicating potential overfitting as it has learned the training data too well.

Test Set:

- The model performs well for non-bankrupt companies on the test set but struggles to correctly classify bankrupt companies, with a low recall of only 26%.

2.



```
-----------------------------------2.2-2-----------------------------------
Training Set Confusion Matrix (Random Forest):
[[5281    0]
 [   1  173]]

Training Set Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5281
           1       1.00      0.99      1.00       174

    accuracy                           1.00      5455
   macro avg       1.00      1.00      1.00      5455
weighted avg       1.00      1.00      1.00      5455


Test Set Confusion Matrix (Random Forest):
[[1308   10]
 [  36   10]]

Test Set Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1318
           1       0.50      0.22      0.30        46

    accuracy                           0.97      1364
   macro avg       0.74      0.60      0.64      1364
weighted avg       0.96      0.97      0.96      1364
```

Training Set:

- The model performs perfectly on the training set with an accuracy of 1.00. Both precision and recall for bankrupt and non-bankrupt companies are very high, indicating overfitting to the training data.

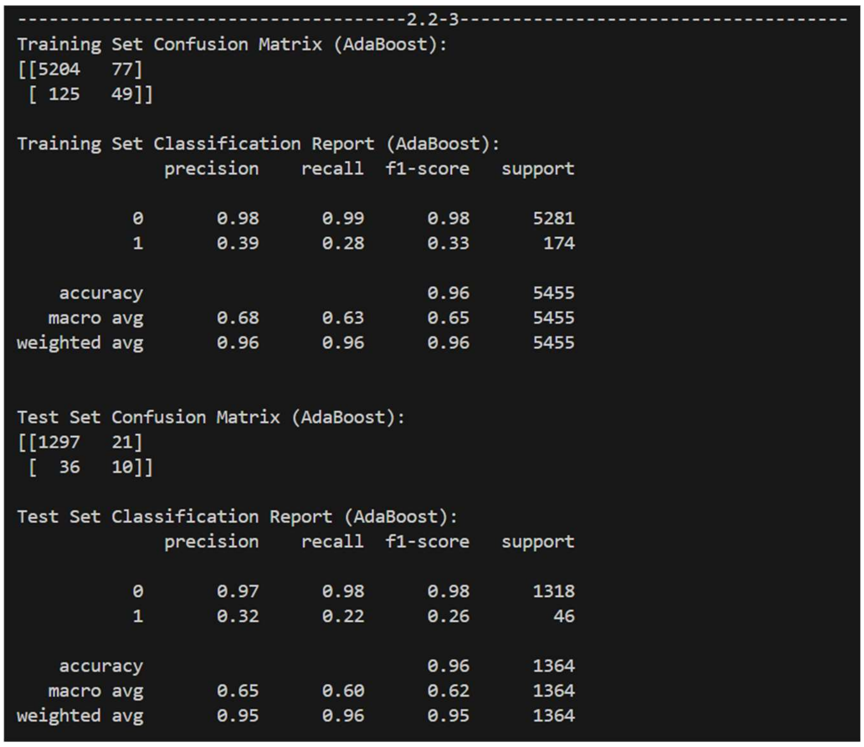
Test Set:

- For non-bankrupt companies (class 0), precision and recall are still very high (0.97 and 0.99, respectively).
- For bankrupt companies (class 1), precision is 0.50, slightly better than Bagging, but recall is only 0.22, meaning the model misses most bankrupt companies.
- Overall accuracy is 0.97, similar to Bagging, but the model still struggles with identifying bankrupt companies.

Comparison to Bagging:

- Improvement: Random forest shows a small improvement in precision for bankrupt companies (0.50 compared to Bagging's 0.48).
- Limitations: The recall for bankrupt companies remains low (0.22), so the model still struggles to identify them.

3.



```
-----------------------------------2.2-3-----------------------------------
Training Set Confusion Matrix (AdaBoost):
[[5204   77]
 [ 125   49]]

Training Set Classification Report (AdaBoost):
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      5281
           1       0.39      0.28      0.33       174

    accuracy                           0.96      5455
   macro avg       0.68      0.63      0.65      5455
weighted avg       0.96      0.96      0.96      5455


Test Set Confusion Matrix (AdaBoost):
[[1297   21]
 [  36   10]]

Test Set Classification Report (AdaBoost):
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      1318
           1       0.32      0.22      0.26        46

    accuracy                           0.96      1364
   macro avg       0.65      0.60      0.62      1364
weighted avg       0.95      0.96      0.95      1364
```

Training Set:

- The model performs well for non-bankrupt companies (class 0), with high precision (0.98) and recall (0.99).
- For bankrupt companies (class 1), precision is low at 0.39, and recall is 0.28, meaning the model misses many bankrupt companies.
- Overall accuracy is 0.96.

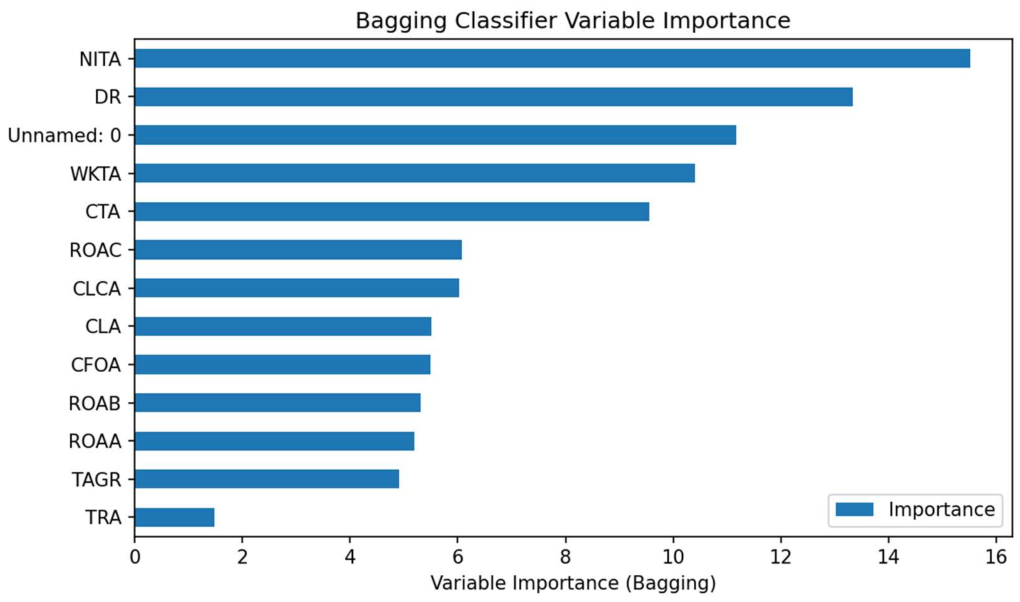
Test Set:

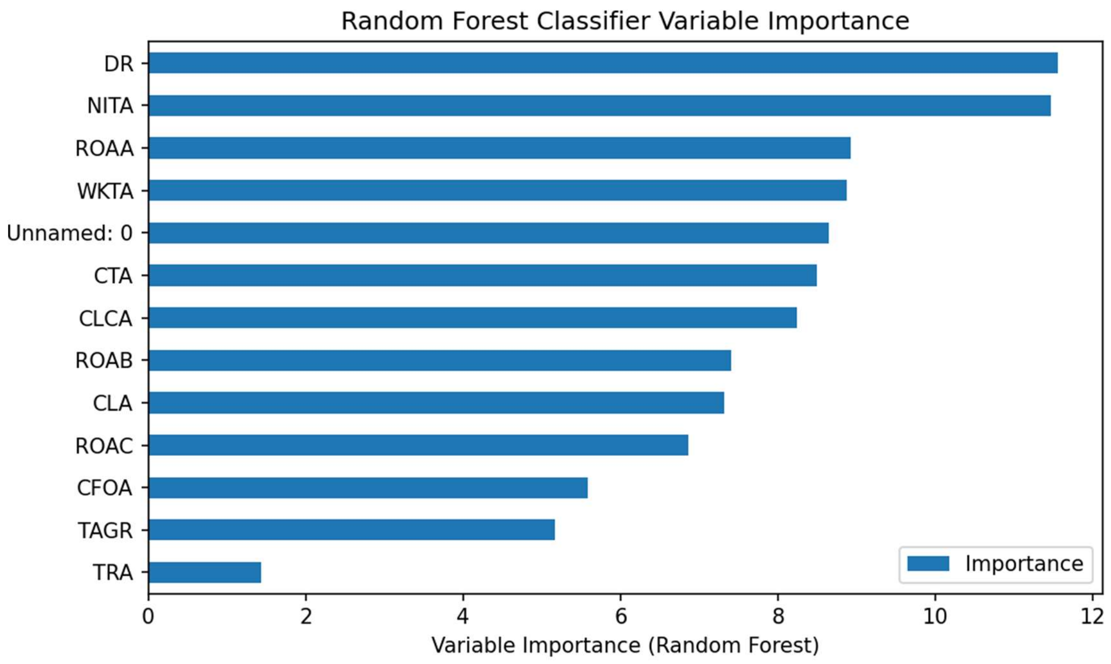
- The model continues to perform well for non-bankrupt companies, with precision of 0.97 and recall of 0.98.
- For bankrupt companies, precision is 0.32 and recall is 0.22, meaning many bankrupt companies are misclassified.
- Overall accuracy is 0.96, but the model struggles with identifying bankrupt companies.

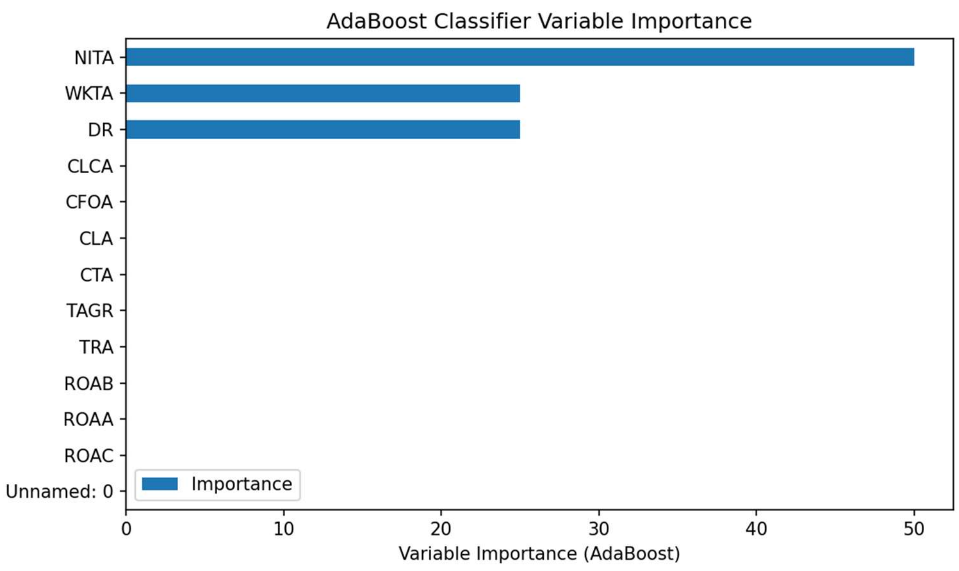
Conclusion:

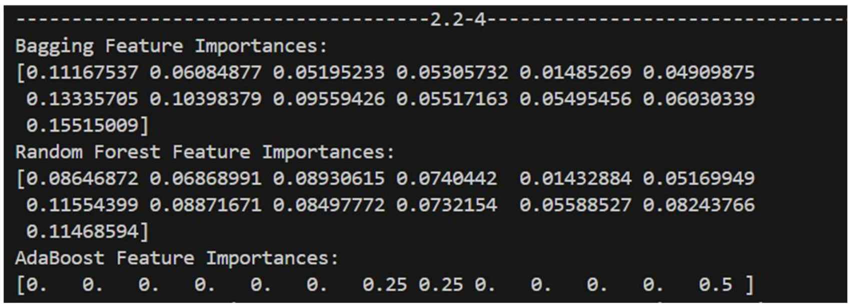
- AdaBoost works well for non-bankrupt companies but struggles with bankrupt companies, especially in recall. Improving the model's ability to identify bankrupt companies is needed.

4.


Bagging Classifier Variable Importance
NITA
DR
Unnamed: 0
WKTA
CTA
ROAC
CLCA
CLA
CFOA
ROAB
ROAA
TAGR
TRA
Importance
0
2
4
6
8
10
12
14
16
Variable Importance (Bagging)


Random Forest Classifier Variable Importance
DR
NITA
ROAA
WKTA
Unnamed: 0
CTA
CLCA
ROAB
CLA
ROAC
CFOA
TAGR
TRA
Importance
0
2
4
6
8
10
12
Variable Importance (Random Forest)





```
-----------------------------------2.2-4-------------------------------
Bagging Feature Importances:
[0.11167537 0.06084877 0.05195233 0.05305732 0.01485269 0.04909875
 0.13335705 0.10398379 0.09559426 0.05517163 0.05495456 0.06030339
 0.15515009]
Random Forest Feature Importances:
[0.08646872 0.06868991 0.08930615 0.0740442  0.01432884 0.05169949
 0.11554399 0.08871671 0.08497772 0.0732154  0.05588527 0.08243766
 0.11468594]
AdaBoost Feature Importances:
[0.   0.   0.   0.   0.   0.   0.25 0.25 0.   0.   0.   0.   0.5 ]
```

<u>**Bagging:**</u>

- The most important features are NITA, DR, and WKTA. NITA is the most significant (15.5%), followed by DR (13.3%). The importance is more balanced across the features.

<u>**Random Forest:**</u>

- Similar to Bagging, the most important features are DR and NITA, with ROAA also significant. It spreads the importance more evenly but focuses on these top features.

<u>**AdaBoost:**</u>

- Focuses almost entirely on NITA (50%), WKTA (25%), and DR (25%), ignoring all other features. It makes the model more interpretable but relies heavily on just a few key variables.

<u>**Conclusion:**</u>

- Across all classifiers, NITA, DR, and WKTA are consistently the most important predictors for bankruptcy.

2.3 Support Vector Machines (SVM)

1. Balanced Accuracy= 1/2 * (TP/(TP+FN)+ TN/(FP+TN))
TP is the number of true positives (correctly predicted positive cases).
TN is the number of true negatives (correctly predicted negative cases).
FP is the number of false positives (actual negatives incorrectly predicted as positives).
FN is the number of false negatives (actual positives incorrectly predicted as negatives).
The balanced accuracy is calculated as the average of the sensitivity (true positive rate) and specificity (true negative rate)
Balanced accuracy is preferable when dealing with imbalanced datasets, where one class significantly outnumbers the other

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# Define a range of C values for grid search
tuned_parameters = [{'C': [0.001, 0.01, 0.1, 1, 5, 10]}]

# Perform 10-fold cross-validation with the specified parameter grid
svm_bankrupt_CV = GridSearchCV(SVC(kernel='linear'), tuned_parameters, cv=10,
scoring='balanced_accuracy')
svm_bankrupt_CV.fit(x_train_std, np.ravel(y_train))

# Extract the best value for the cost parameter 'C'
best_C = svm_bankrupt_CV.best_params_['C']
print(f"The best value of C in terms of accuracy is: {best_C}")

# Train the SVM model with the best parameter
svm_bankrupt = SVC(kernel='linear', C=best_C)
svm_bankrupt.fit(x_train_std, np.ravel(y_train))

# Determine the total number of support vectors
total_support_vectors = sum(svm_bankrupt.n_support_)
print(f"The model has {total_support_vectors} support vectors.")
```

```
The best value of C in terms of accuracy is: 0.001
The model has 378 support vectors.
```

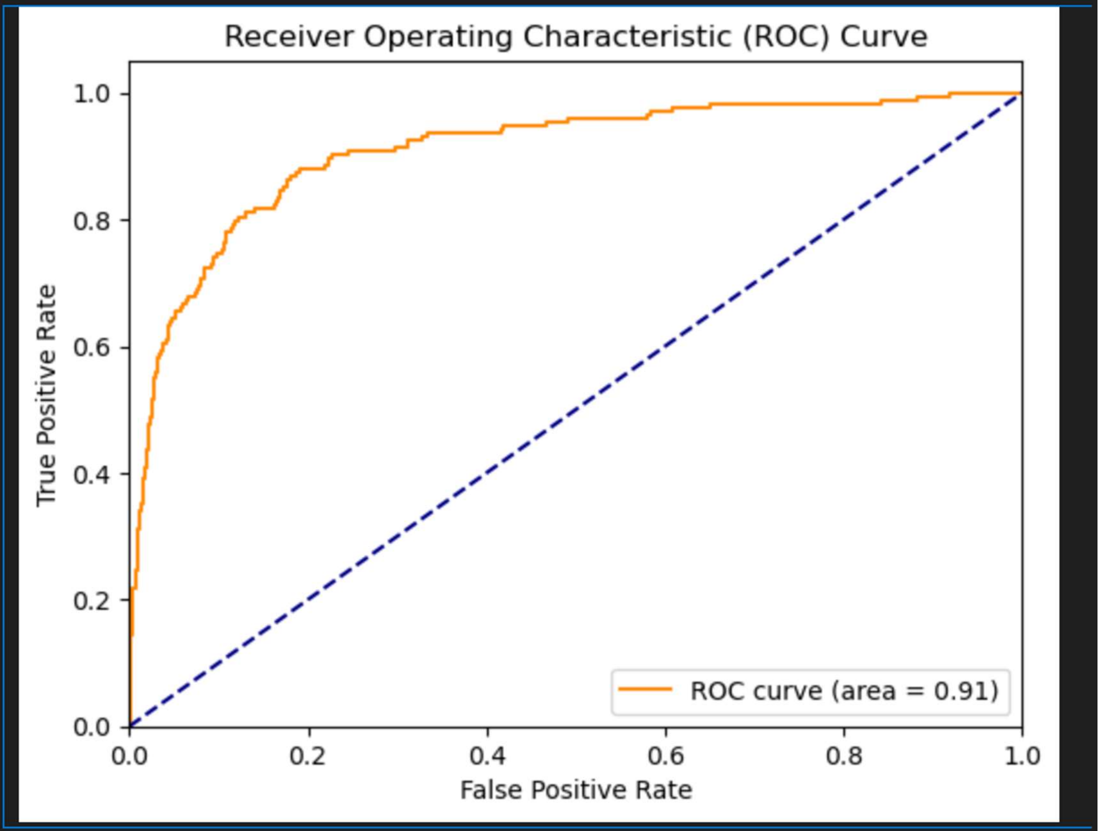
```
#3.
# Prediction using the best SVM model
y_pred = svm_bankrupt.predict(x_train_std)
```

```
print("Confusion Matrix:")
print(confusion_matrix(y_train, y_pred))

print("\nClassification Report:")
print(classification_report(y_train, y_pred))
```

```
Confusion Matrix: [[5281 0] [ 174 0]] Classification Report: precision recall f1-score
support 0 0.97 1.00 0.98 5281 1 0.00 0.00 0.00 174 accuracy 0.97 5455 macro avg 0.48
0.50 0.49 5455 weighted avg 0.94 0.97 0.95 5455
```

```
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision
and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision
and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result))
```

```
from sklearn.metrics import roc_curve, auc
# Getting the probabilities of our predictions
y_probs = svm_bankrupt.decision_function(x_train_std)
fpr, tpr, thresholds = roc_curve(y_train, y_probs)

roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



```
# Determine the optimal classification threshold
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Optimal threshold value:", optimal_threshold)

# Classification report for the chosen threshold
y_pred_optimal = np.where(y_probs >= optimal_threshold, 1, -1)

print("Classification Report for Optimal Threshold:")
print(classification_report(y_train, y_pred_optimal))
```

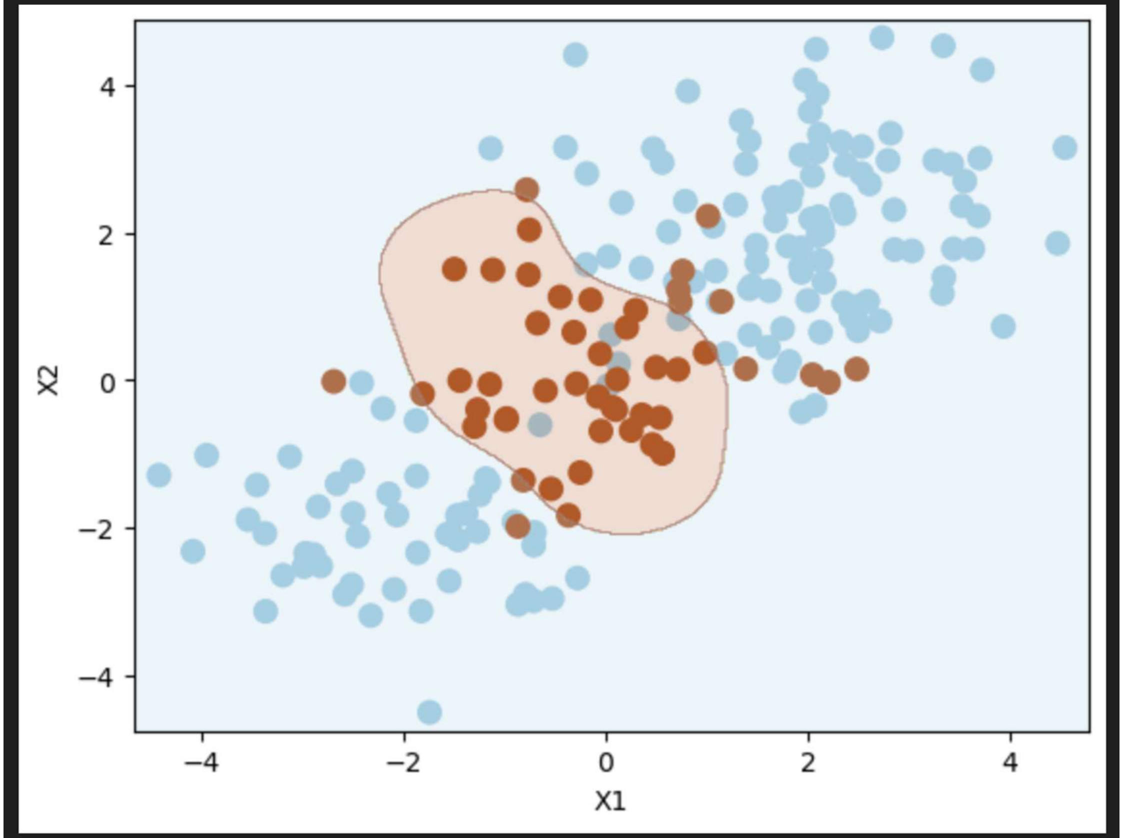
```
Optimal threshold value: -1.0001417233856156 Classification Report for Optimal
Threshold: precision recall f1-score support -1 0.00 0.00 0.00 0 0 0.00 0.00 0.00 5281 1
0.13 0.88 0.23 174 accuracy 0.03 5455 macro avg 0.04 0.29 0.08 5455 weighted avg
0.00 0.03 0.01 5455
```

```
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
labels with no predicted samples. Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and
```

```
F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision
and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and
F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision
and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result)) D:\software\Conda\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and
F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior. _warn_prf(average, modifier,
msg_start, len(result))
```

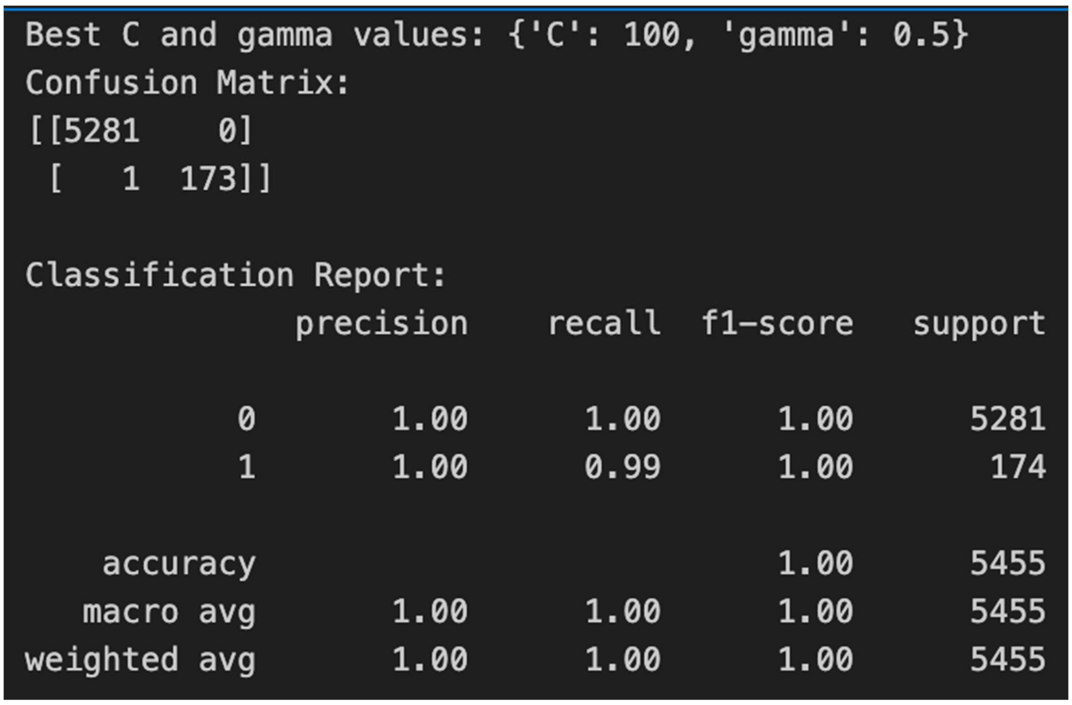
```
#4.
# Plot a SVM with non-linear kernels
np. random. seed (8)
X = np. random.randn (200 ,2)
X[: 100] = X[: 100] + 2
X[101: 150] = X[101: 150] - 2
Y = np.concatenate([np.repeat (-1, 150) , np.repeat ( 1, 50)])
plt.scatter(X[:, 0], X[:, 1], s=70, c=Y, cmap=plt.cm.Paired)
plt.xlabel ('X1')
plt.ylabel ('X2')

svm_radial = SVC(C=1.0, kernel='rbf', gamma=1)
svm_radial.fit(X, Y)

h = 0.02
pad = 0.25
xmin, xmax = X[:, 0].min() - pad, X[:, 0].max() + pad
ymin, ymax = X[:, 1].min() - pad, X[:, 1].max() + pad
xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))
```

```
Z = svm_radial.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.2)
plt.show()
```



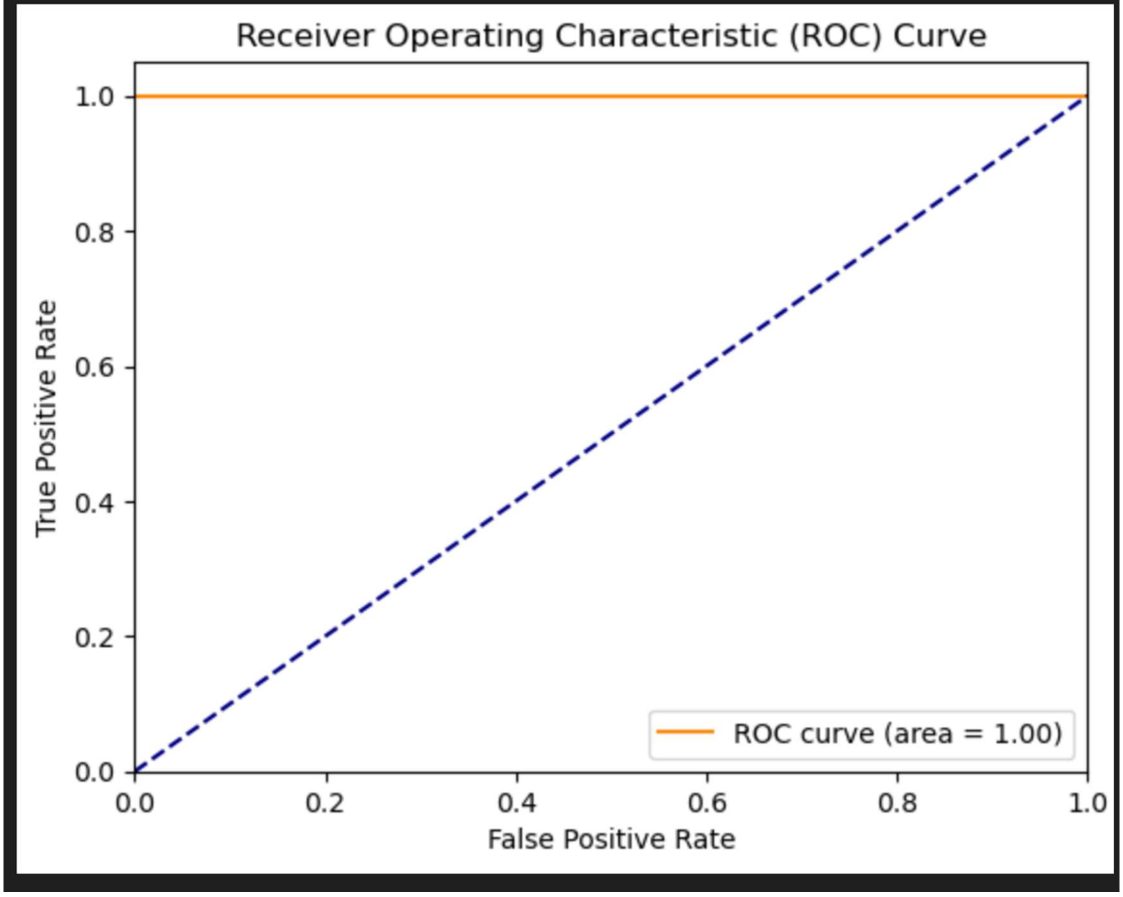
```
#4.
# Define a range of C and gamma values for grid search
tuned_parameters = [{'C': [0.01, 0.1, 1, 10, 100], 'gamma': [0.5, 1, 2, 3, 4]}]

# Create an SVM model with an RBF kernel
svm_rbf = SVC(kernel='rbf')

# Perform grid search for C and gamma with cross-validation
svm_rbf_CV = GridSearchCV(svm_rbf, tuned_parameters, cv=10,
scoring='balanced_accuracy')
svm_rbf_CV.fit(x_train_std, np.ravel(y_train))

# Get the best C and gamma values
best_params = svm_rbf_CV.best_params_
print(f"Best C and gamma values: {best_params}")
```

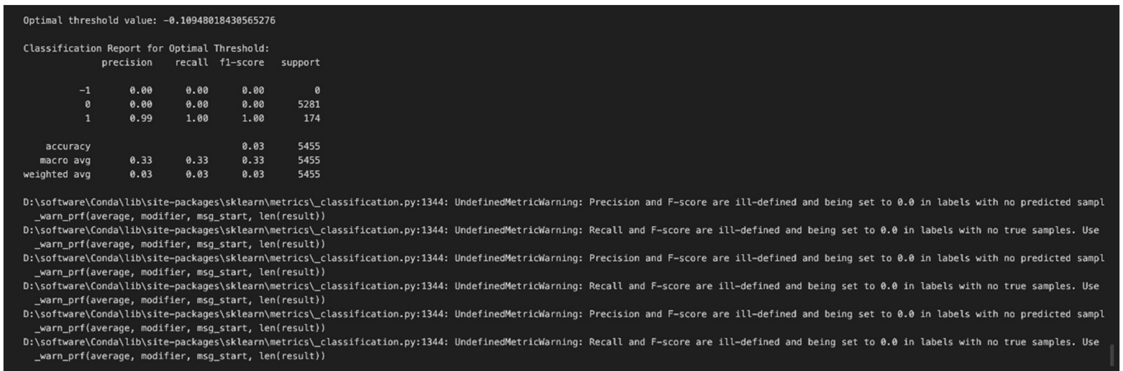
```
# Fit the SVM classifier with the chosen parameters
best_C = best_params['C']
best_gamma = best_params['gamma']

svm_radial_bankrupt = SVC(kernel='rbf', C=best_C, gamma=best_gamma)
svm_radial_bankrupt.fit(x_train_std, np.ravel(y_train))

# Make prediction
y_pred = svm_radial_bankrupt.predict(x_train_std)

# Calculate the confusion matrix and classification report
print("Confusion Matrix:")
print(confusion_matrix(y_train, y_pred))

print("\nClassification Report:")
print(classification_report(y_train, y_pred))
```



```
Best C and gamma values: {'C': 100, 'gamma': 0.5}
Confusion Matrix:
[[5281    0]
 [   1  173]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5281
           1       1.00      0.99      1.00       174

    accuracy                           1.00      5455
   macro avg       1.00      1.00      1.00      5455
weighted avg       1.00      1.00      1.00      5455
```

```
# Plot ROC Curve and Optimal Threshold
y_probs = svm_radial_bankrupt.decision_function(x_train_std)
fpr, tpr, thresholds = roc_curve(y_train, y_probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Optimal threshold value:", optimal_threshold)

y_pred_optimal = np.where(y_probs >= optimal_threshold, 1, -1)
print("\nClassification Report for Optimal Threshold:")
print(classification_report(y_train, y_pred_optimal))
```

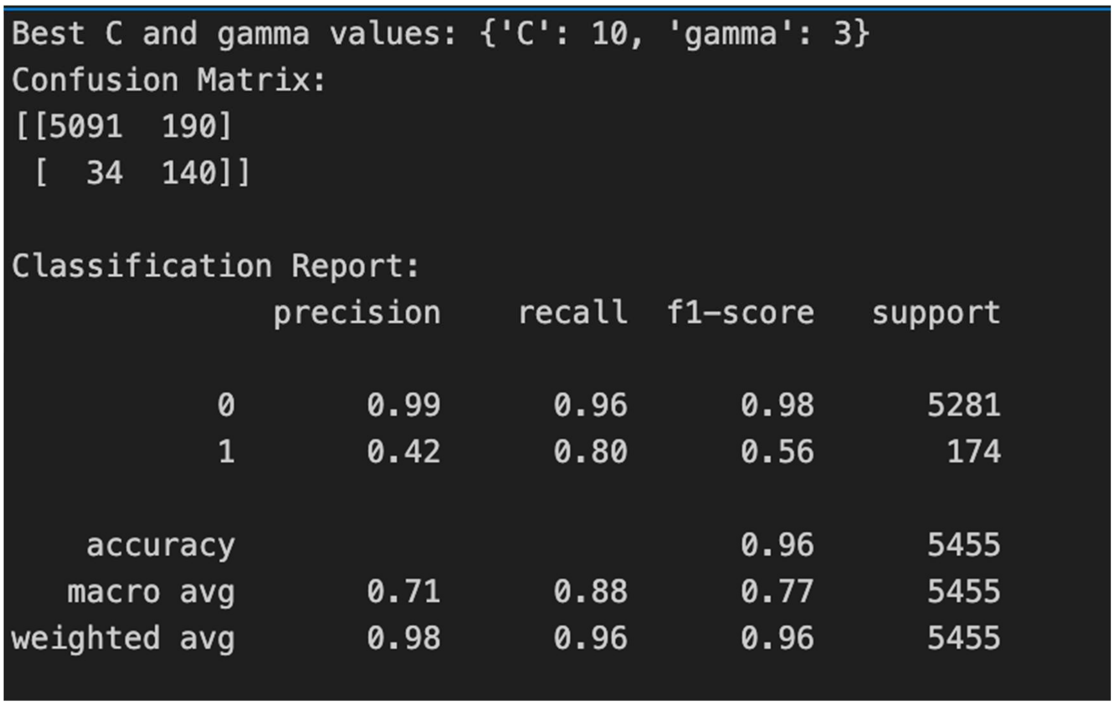




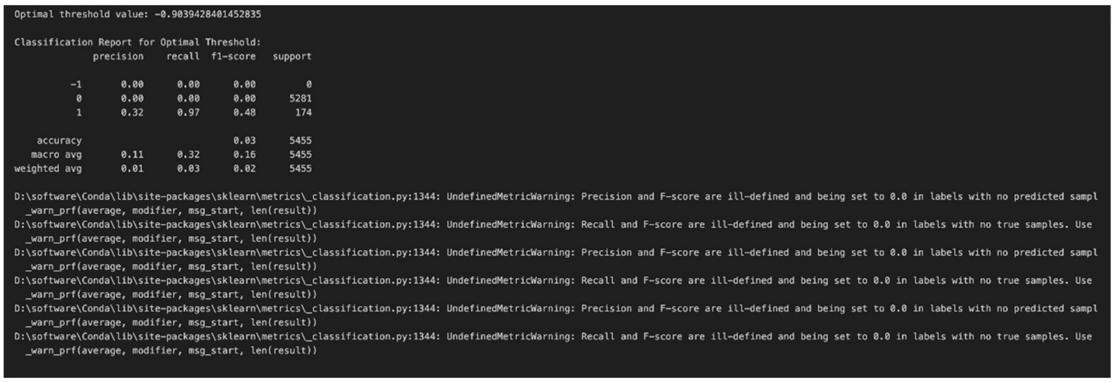
```
Optimal threshold value: -0.10948018430565276

Classification Report for Optimal Threshold:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00         0
           0       0.00      0.00      0.00      5281
           1       0.99      1.00      1.00       174

    accuracy                           0.03      5455
   macro avg       0.33      0.33      0.33      5455
weighted avg       0.03      0.03      0.03      5455

D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
  _warn_prf(average, modifier, msg_start, len(result))
```

```
# Without standerlization
# Define a range of C and gamma values for grid search
tuned_parameters = [{'C': [0.01, 0.1, 1, 10, 100], 'gamma': [0.5, 1, 2, 3, 4]}]

# Create an SVM model with an RBF kernel
svm_rbf = SVC(kernel='rbf')

# Perform grid search for C and gamma with cross-validation
svm_rbf_CV = GridSearchCV(svm_rbf, tuned_parameters, cv=10,
scoring='balanced_accuracy')
svm_rbf_CV.fit(x_train, np.ravel(y_train))

# Get the best C and gamma values
best_params = svm_rbf_CV.best_params_
print(f"Best C and gamma values: {best_params}")

# Fit the SVM classifier with the chosen parameters
best_C = best_params['C']
best_gamma = best_params['gamma']

svm_radial_bankrupt = SVC(kernel='rbf', C=best_C, gamma=best_gamma)
svm_radial_bankrupt.fit(x_train, np.ravel(y_train))

# Make prediction
y_pred = svm_radial_bankrupt.predict(x_train)

# Calculate the confusion matrix and classification report
print("Confusion Matrix:")
print(confusion_matrix(y_train, y_pred))

print("\nClassification Report:")
```

```python
print(classification_report(y_train, y_pred))

# Plot ROC Curve and Optimal Threshold
y_probs = svm_radial_bankrupt.decision_function(x_train)
fpr, tpr, thresholds = roc_curve(y_train, y_probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Optimal threshold value:", optimal_threshold)

y_pred_optimal = np.where(y_probs >= optimal_threshold, 1, -1)
print("\nClassification Report for Optimal Threshold:")
print(classification_report(y_train, y_pred_optimal))
```

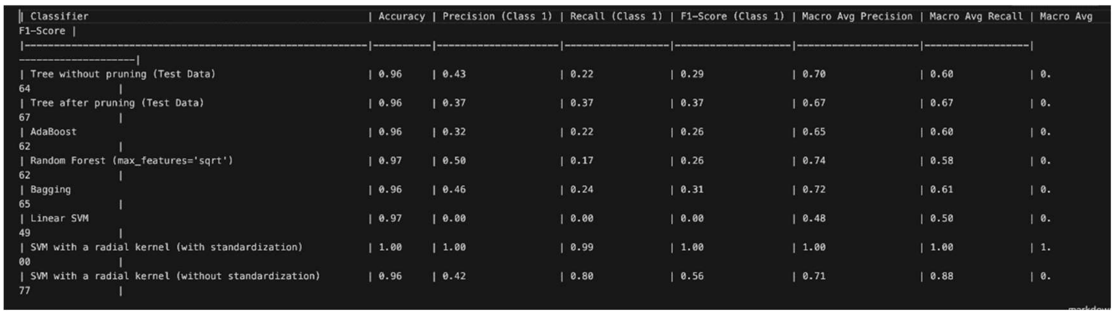


```
Best C and gamma values: {'C': 10, 'gamma': 3}
Confusion Matrix:
[[5091  190]
 [  34  140]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.96      0.98      5281
           1       0.42      0.80      0.56       174

    accuracy                           0.96      5455
   macro avg       0.71      0.88      0.77      5455
weighted avg       0.98      0.96      0.96      5455
```



```
Optimal threshold value: -0.9039428401452835

Classification Report for Optimal Threshold:
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00         0
           0       0.00      0.00      0.00      5281
           1       0.32      0.97      0.48       174

    accuracy                           0.03      5455
   macro avg       0.11      0.32      0.16      5455
weighted avg       0.01      0.03      0.02      5455

D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sampl
  _warn_prf(average, modifier, msg_start, len(result))
D:\software\Conda\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use
  _warn_prf(average, modifier, msg_start, len(result))
```

2.4. Conclusion



Classifier	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	Macro Avg Precision	Macro Avg Recall	Macro Avg F1-Score
Tree without pruning (Test Data)	0.96	0.43	0.22	0.29	0.70	0.60	0.64
Tree after pruning (Test Data)	0.96	0.37	0.37	0.37	0.67	0.67	0.67
AdaBoost	0.96	0.32	0.22	0.26	0.65	0.60	0.62
Random Forest (max_features='sqrt')	0.97	0.50	0.17	0.26	0.74	0.58	0.62
Bagging	0.96	0.46	0.24	0.31	0.72	0.61	0.65
Linear SVM	0.97	0.00	0.00	0.00	0.48	0.50	0.49
SVM with a radial kernel (with standardization)	1.00	1.00	0.99	1.00	1.00	1.00	1.00
SVM with a radial kernel (without standardization)	0.96	0.42	0.80	0.56	0.71	0.88	0.77

Ensemble Methods: Among AdaBoost, Random Forest, and Bagging, the Random Forest method with max_features='sqrt' has the highest precision for Class 1, while all of them have comparable high accuracy and F1-Score for Class 0. This indicates the strong capability of ensemble methods in handling the dominant class but might require tuning for the minority class.

Decision Trees: Pruning improved the recall for Class 1 from 0.22 to 0.37, which

```
can be essential if correctly identifying positive cases of Class 1 is a priority.

**SVM**: The radial SVM with standardization performed exceptionally well, achieving
near-perfect scores across all metrics. On the other hand, without standardization, the
precision drops, but the recall for Class 1 increases significantly. This suggests the
importance of feature scaling for SVM.
The linear SVM struggles with Class 1, indicating possible non-linearity in the data.

**Summary**: While ensemble methods and decision trees provide good results, SVM
with a radial kernel and feature standardization stands out as the top-performing model.

While some classifiers achieved a high overall accuracy, it's important to consider
metrics like F1-Score, precision, and recall, especially for imbalanced datasets.

However, the choice of the model should also consider the specific goals and
requirements of the project, such as whether precision or recall is more crucial for Class
1.
```