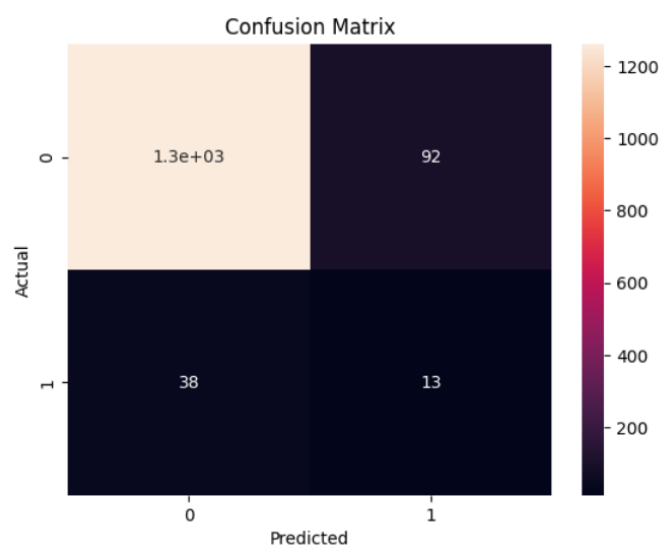
**Project Development Phase**
**Model Performance Test**

| | |
|---|---|
| Date | 16 November 2023 |
| Team ID | PNT2022TMIDxxxxxx |
| Project Name | Project - Anticipating Business Bankruptcy |
| Maximum Marks | 10 Marks |

**Model Performance Testing:**

| S.No. | Parameter | Values | Screenshot |
|---|---|---|---|
| 1. | Metrics | **Classification Model:**<br>**1) Random Forest classifier**<br>• Accuracy score<br>• Classification report<br>• Confusion matrix | [47]: `print('Training Accuracy for RandomForest = ', accuracy_score(y_train_resampled,y_pred1_train))`<br>`Training Accuracy for RandomForest =  0.9999074759437454`<br>[48]: `print('Testing Accuracy for RandomForest = ', accuracy_score(y_test,y_pred1_test))`<br>`Testing Accuracy for RandomForest =  0.9073414112615823`<br>[49]:<br>`print("====================Random Forest Classifier====================")`<br>`print(classification_report(y_test, y_pred1_test))`<br>`from sklearn.metrics import confusion_matrix`<br>`import matplotlib.pyplot as plt`<br>`cm=confusion_matrix(y_test, y_pred1_test)`<br>`# Create a heatmap using Seaborn`<br>`sbn.heatmap(cm, annot=True)`<br>`plt.xlabel('Predicted')`<br>`plt.ylabel('Actual')`<br>`plt.title('Confusion Matrix')`<br>`plt.show()`<br>`====================Random Forest Classifier====================`<br>`precision recall f1-score support`<br>`0.0 0.97 0.93 0.95 1352`<br>`1.0 0.12 0.25 0.17 51`<br>`accuracy 0.91 1403`<br>`macro avg 0.55 0.59 0.56 1403`<br>`weighted avg 0.94 0.91 0.92 1403`<br>  <br> |
| | | | |

**2) SupportVectorMachine (SVM)**

- Accuracy score
- Classification report
- Confusion matrix

```
[61]: print('Training Accuracy for SVM = ', accuracy_score(y_train_resampled,y_pred4_train))
```

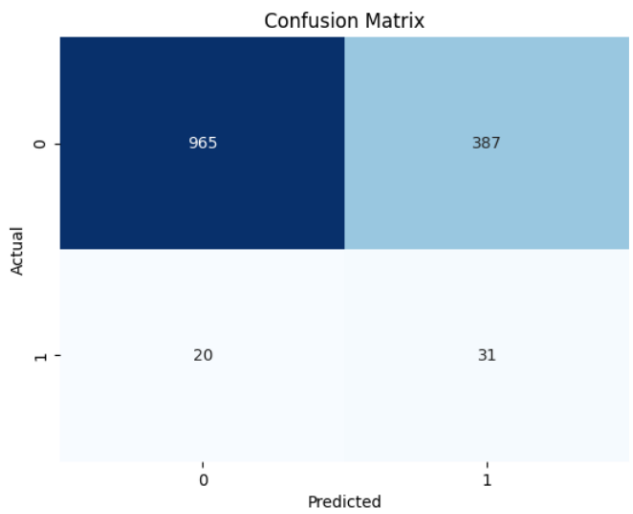
```
Training Accuracy for SVM =  0.8035714285714286
```

```
[62]: print('Testing Accuracy for SVM = ', accuracy_score(y_test,y_pred4_test))
```

```
Testing Accuracy for SVM =  0.7099073414112615
```

```
[63]: print("====================SVM===================")
print(classification_report(y_test, y_pred4_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm = confusion_matrix(y_test, y_pred4_test)

# Create a heatmap using Seaborn
sbn.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
====================SVM===================
              precision    recall  f1-score   support

         0.0       0.98      0.71      0.83      1352
         1.0       0.07      0.61      0.13        51

    accuracy                           0.71      1403
   macro avg       0.53      0.66      0.48      1403
weighted avg       0.95      0.71      0.80      1403
```




**3) K-Nearest Neighbors (KNN)**

- Accuracy score
- Classification report
- Confusion matrix

```
[51]: print('Training Accuracy for KNN = ', accuracy_score(y_train_resampled,y_pred2_train))
```

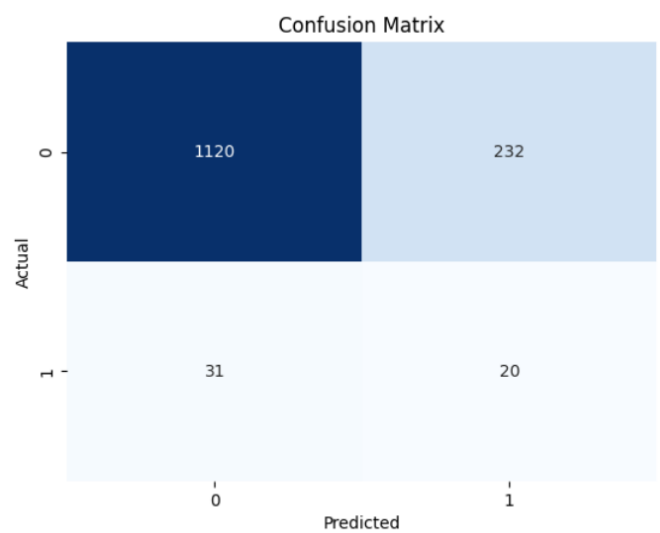
```
Training Accuracy for KNN =  0.9394892672094745
```

```
[52]: print('Testing Accuracy for KNN = ', accuracy_score(y_test,y_pred2_test))
```

```
Testing Accuracy for KNN =  0.812544547398432
```

```
[53]: print("====================KNN====================")
      print(classification_report(y_test, y_pred2_test))
      from sklearn.metrics import confusion_matrix
      import matplotlib.pyplot as plt


      cm = confusion_matrix(y_test, y_pred2_test)

      # Create a heatmap using Seaborn
      sbn.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
      plt.title('Confusion Matrix')
      plt.show()
```

```
====================KNN====================
              precision    recall  f1-score   support

         0.0       0.97      0.83      0.89      1352
         1.0       0.08      0.39      0.13        51

    accuracy                           0.81      1403
   macro avg       0.53      0.61      0.51      1403
weighted avg       0.94      0.81      0.87      1403
```



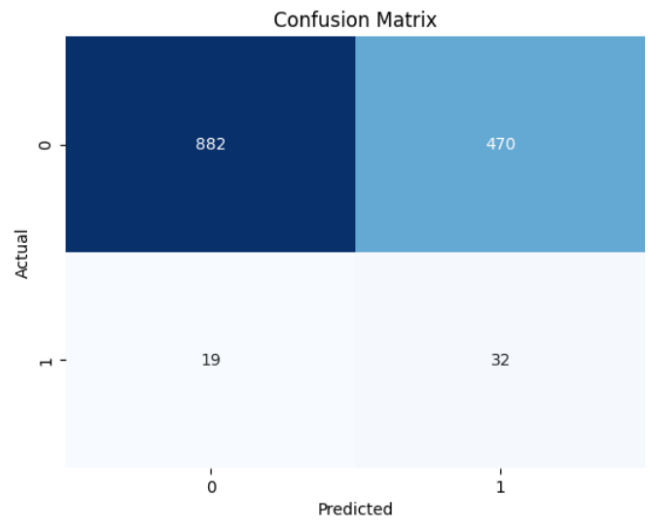

**4) Logistic Regression**

- Accuracy score
- Classification report
- Confusion matrix

```
[56]: print('Training Accuracy for Logistic Reg = ', accuracy_score(y_train_resampled,y_pred3_train))
      Training Accuracy for Logistic Reg =  0.6871761658031088

[57]: print('Testing Accuracy for Logistic Reg = ', accuracy_score(y_test,y_pred3_test))
      Testing Accuracy for Logistic Reg =  0.6514611546685674
```

```
[58]: print("====================Logistic Regression====================")
print(classification_report(y_test, y_pred3_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm = confusion_matrix(y_test, y_pred3_test)

# Create a heatmap using Seaborn
sbn.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
====================Logistic Regression====================
              precision    recall  f1-score   support

         0.0       0.98      0.65      0.78      1352
         1.0       0.06      0.63      0.12        51

    accuracy                           0.65      1403
   macro avg       0.52      0.64      0.45      1403
weighted avg       0.95      0.65      0.76      1403
```




**5) Decision tree classifier**

- Accuracy score
- Classification report
- Confusion matrix

```
[66]: print('Training Accuracy for DecisionTree = ', accuracy_score(y_train_resampled,y_pred5_train))
```

```
Training Accuracy for DecisionTree =  0.9999074759437454
```

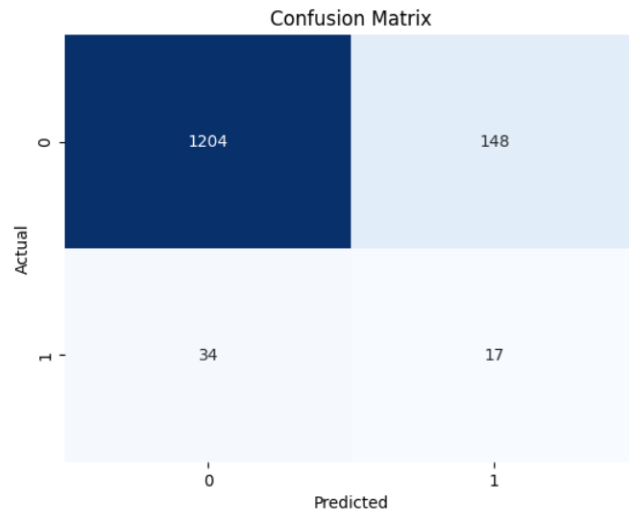
```
[67]: print('Testing Accuracy for DecisionTree = ', accuracy_score(y_test,y_pred5_test))
```

```
Testing Accuracy for DecisionTree =  0.8702779757662152
```

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>

```
[68]: print("====================Decision Trees====================")
print(classification_report(y_test, y_pred5_test))
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


cm = confusion_matrix(y_test, y_pred5_test)

# Create a heatmap using Seaborn
sbn.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
====================Decision Trees====================
              precision    recall  f1-score   support

         0.0       0.97      0.89      0.93      1352
         1.0       0.10      0.33      0.16        51

    accuracy                           0.87      1403
   macro avg       0.54      0.61      0.54      1403
weighted avg       0.94      0.87      0.90      1403
```



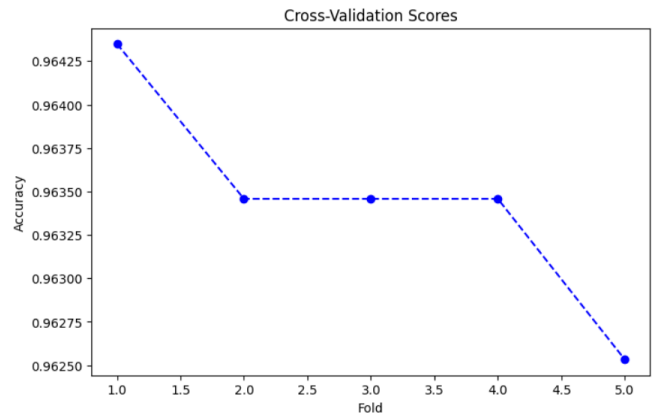

</td>
</tr>
<tr>
<td>2.</td>
<td>Tune the Model</td>
<td>

- Hyperparameter Tuning for random forest classifier
- Validation Method

</td>
<td>

```
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create GridSearchCV
grid_search = GridSearchCV(rfc, param_grid, cv=5, scoring='accuracy', n_jobs=-1)

# Fit the model
grid_search.fit(X_train, y_train)

# Print the best parameters
print("Best Parameters:", grid_search.best_params_)

# Cross-validation scores
cv_scores = cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=5)

# Print cross-validation scores
print("Cross-validation Scores:", cv_scores)

# Plotting cross-validation scores
plt.figure(figsize=(8, 5))
plt.plot(range(1, 6), cv_scores, marker='o', linestyle='--', color='b')
plt.title('Cross-Validation Scores')
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.show()
```

</td>
</tr>
</table>

| | | | |
|---|---|---|---|
| | | | Best Parameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50}<br>Cross-validation Scores: [0.96434938 0.96345811 0.96345811 0.96345811 0.96253345]<br>  <br> |