Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMIDxxxxxx
Project Name	Project – Smart Lender
Maximum Marks	10 Marks

Model Performance Testing:

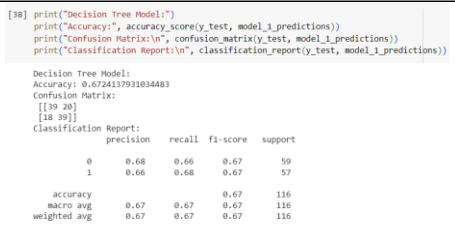
Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	See below	See below

Values

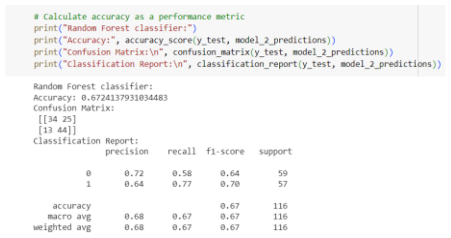
```
Decision Tree Model:
Accuracy: 0.6724137931034483
Confusion Matrix:
 [[39 20]
 [18 39]]
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.66      0.67        59
           1       0.66      0.68      0.67        57

    accuracy                           0.67       116
   macro avg       0.67      0.67      0.67       116
weighted avg       0.67      0.67      0.67       116


Random Forest classifier:
Accuracy: 0.6724137931034483
Confusion Matrix:
 [[34 25]
 [13 44]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.58      0.64        59
           1       0.64      0.77      0.70        57

    accuracy                           0.67       116
   macro avg       0.68      0.67      0.67       116
weighted avg       0.68      0.67      0.67       116


KNN Classifier:
Accuracy: 0.6637931034482759
Confusion Matrix:
 [[31 28]
 [11 46]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.53      0.61        59
           1       0.62      0.81      0.70        57

    accuracy                           0.66       116
   macro avg       0.68      0.67      0.66       116
weighted avg       0.68      0.66      0.66       116


XG Boost:
Accuracy: 0.7155172413793104
Confusion Matrix:
 [[37 22]
 [11 46]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.63      0.69        59
           1       0.68      0.81      0.74        57

    accuracy                           0.72       116
   macro avg       0.72      0.72      0.71       116
weighted avg       0.72      0.72      0.71       116
```

Screenshot



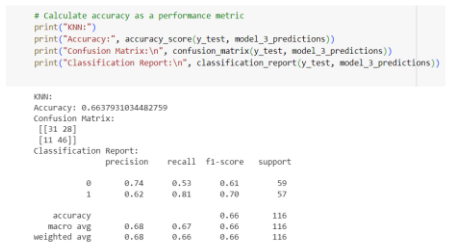
```
[38] print("Decision Tree Model:")
     print("Accuracy:", accuracy_score(y_test, model_1_predictions))
     print("Confusion Matrix:\n", confusion_matrix(y_test, model_1_predictions))
     print("Classification Report:\n", classification_report(y_test, model_1_predictions))
```

```
Decision Tree Model:
Accuracy: 0.6724137931034483
Confusion Matrix:
 [[39 20]
 [18 39]]
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.66      0.67        59
           1       0.66      0.68      0.67        57

    accuracy                           0.67       116
   macro avg       0.67      0.67      0.67       116
weighted avg       0.67      0.67      0.67       116
```



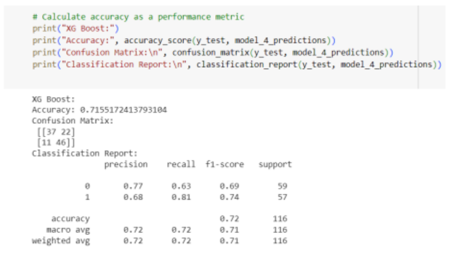
```
# Calculate accuracy as a performance metric
print("Random Forest classifier:")
print("Accuracy:", accuracy_score(y_test, model_2_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, model_2_predictions))
print("Classification Report:\n", classification_report(y_test, model_2_predictions))
```

```
Random Forest classifier:
Accuracy: 0.6724137931034483
Confusion Matrix:
 [[34 25]
 [13 44]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.58      0.64        59
           1       0.64      0.77      0.70        57

    accuracy                           0.67       116
   macro avg       0.68      0.67      0.67       116
weighted avg       0.68      0.67      0.67       116
```



```
# Calculate accuracy as a performance metric
print("KNN:")
print("Accuracy:", accuracy_score(y_test, model_3_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, model_3_predictions))
print("Classification Report:\n", classification_report(y_test, model_3_predictions))
```

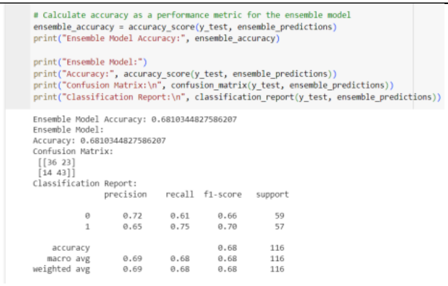
```
KNN:
Accuracy: 0.6637931034482759
Confusion Matrix:
 [[31 28]
 [11 46]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.53      0.61        59
           1       0.62      0.81      0.70        57

    accuracy                           0.66       116
   macro avg       0.68      0.67      0.66       116
weighted avg       0.68      0.66      0.66       116
```



```
# Calculate accuracy as a performance metric
print("XG Boost:")
print("Accuracy:", accuracy_score(y_test, model_4_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, model_4_predictions))
print("Classification Report:\n", classification_report(y_test, model_4_predictions))
```

```
XG Boost:
Accuracy: 0.7155172413793104
Confusion Matrix:
 [[37 22]
 [11 46]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.63      0.69        59
           1       0.68      0.81      0.74        57

    accuracy                           0.72       116
   macro avg       0.72      0.72      0.71       116
weighted avg       0.72      0.72      0.71       116
```

```
Ensemble Model:
Accuracy: 0.6810344827586207
Confusion Matrix:
 [[36 23]
 [14 43]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.61      0.66        59
           1       0.65      0.75      0.70        57

    accuracy                           0.68       116
   macro avg       0.69      0.68      0.68       116
weighted avg       0.69      0.68      0.68       116
```



```
# Calculate accuracy as a performance metric for the ensemble model
ensemble_accuracy = accuracy_score(y_test, ensemble_predictions)
print("Ensemble Model Accuracy:", ensemble_accuracy)

print("Ensemble Model:")
print("Accuracy:", accuracy_score(y_test, ensemble_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, ensemble_predictions))
print("Classification Report:\n", classification_report(y_test, ensemble_predictions))
```

```
Ensemble Model Accuracy: 0.6810344827586207
Ensemble Model:
Accuracy: 0.6810344827586207
Confusion Matrix:
 [[36 23]
 [14 43]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.61      0.66        59
           1       0.65      0.75      0.70        57

    accuracy                           0.68       116
   macro avg       0.69      0.68      0.68       116
weighted avg       0.69      0.68      0.68       116
```

2. Tune the Model

Decision Tree Model:

Hyperparameter Tuning –

```
Best Parameters: {'criterion': 'entropy', 'max_depth':
10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Accuracy: 0.6637931034482759
Confusion Matrix:
 [[40 19]
 [20 37]]
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.68      0.67        59
           1       0.66      0.65      0.65        57

    accuracy                           0.66       116
   macro avg       0.66      0.66      0.66       116
weighted avg       0.66      0.66      0.66       116
```

Validation Method –

```
Decision Tree Model:
Best Parameters: {'criterion': 'entropy', 'max_depth':
10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Cross-Validation Scores: [0.66666667 0.59259259
0.75925926 0.69811321 0.66037736]
Mean CV Accuracy: 0.6754018169112508
```

Random Forest classifier:

Hyperparameter tuning:

```
Best Parameters: {'max_depth': 10, 'min_samples_leaf':
2, 'min_samples_split': 10, 'n_estimators': 50}
Accuracy: 0.6982758620689655
Confusion Matrix:
 [[34 25]
 [10 47]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.58      0.66        59
           1       0.65      0.82      0.73        57

    accuracy                           0.70       116
   macro avg       0.71      0.70      0.69       116
weighted avg       0.71      0.70      0.69       116
```

Validation:

```
Random Forest Model:
Best Parameters: {'max_depth': 10, 'min_samples_leaf':
2, 'min_samples_split': 10, 'n_estimators': 50}
Cross-Validation Scores: [0.74074074 0.62962963
0.77777778 0.81132075 0.66037736]
Mean CV Accuracy: 0.7239692522711391
```

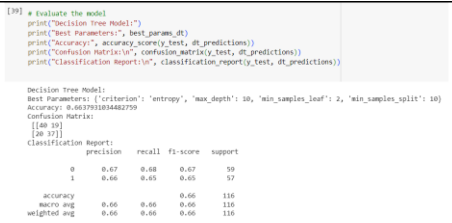
KNN:

Hyperparameter tuning -

```
Best Parameters: {'n_neighbors': 5, 'p': 1, 'weights':
'distance'}
Accuracy: 0.6379310344827587
Confusion Matrix:
 [[30 29]
 [13 44]]
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.51      0.59        59
           1       0.60      0.77      0.68        57

    accuracy                           0.64       116
   macro avg       0.65      0.64      0.63       116
weighted avg       0.65      0.64      0.63       116
```

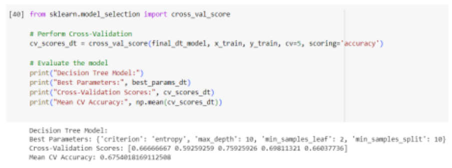
Validation:

```
Best Parameters: {'n_neighbors': 5, 'p': 1, 'weights':
'distance'}
```



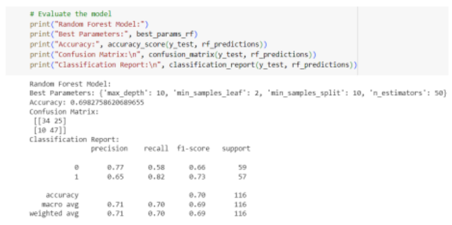
```
[39] # Evaluate the model
     print("Decision Tree Model:")
     print("Best Parameters:", best_params_dt)
     print("Accuracy:", accuracy_score(y_test, dt_predictions))
     print("Confusion Matrix:\n", confusion_matrix(y_test, dt_predictions))
     print("Classification Report:\n", classification_report(y_test, dt_predictions))
```

```
Decision Tree Model:
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Accuracy: 0.6637931034482759
Confusion Matrix:
 [[40 19]
 [20 37]]
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.68      0.67        59
           1       0.66      0.65      0.65        57

    accuracy                           0.66       116
   macro avg       0.66      0.66      0.66       116
weighted avg       0.66      0.66      0.66       116
```



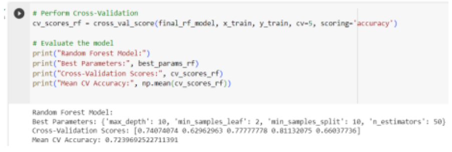
```
[40] from sklearn.model_selection import cross_val_score

     # Perform Cross-Validation
     cv_scores_dt = cross_val_score(final_dt_model, x_train, y_train, cv=5, scoring='accuracy')

     # Evaluate the model
     print("Decision Tree Model:")
     print("Best Parameters:", best_params_dt)
     print("Cross-Validation Scores:", cv_scores_dt)
     print("Mean CV Accuracy:", np.mean(cv_scores_dt))
```

```
Decision Tree Model:
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10}
Cross-Validation Scores: [0.66666667 0.59259259 0.75925926 0.69811321 0.66037736]
Mean CV Accuracy: 0.6754018169112508
```



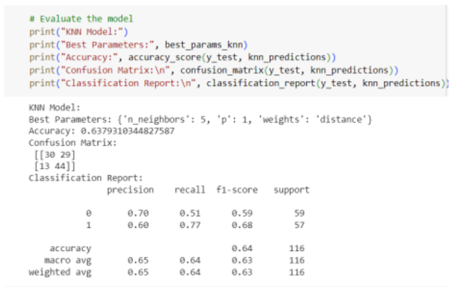
```
# Evaluate the model
print("Random Forest Model:")
print("Best Parameters:", best_params_rf)
print("Accuracy:", accuracy_score(y_test, rf_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_predictions))
print("Classification Report:\n", classification_report(y_test, rf_predictions))
```

```
Random Forest Model:
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 50}
Accuracy: 0.6982758620689655
Confusion Matrix:
 [[34 25]
 [10 47]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.58      0.66        59
           1       0.65      0.82      0.73        57

    accuracy                           0.70       116
   macro avg       0.71      0.70      0.69       116
weighted avg       0.71      0.70      0.69       116
```



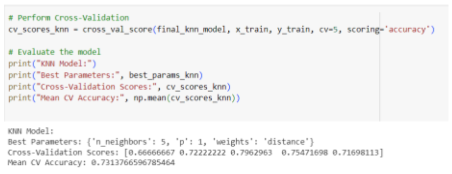
```
# Perform Cross-Validation
cv_scores_rf = cross_val_score(final_rf_model, x_train, y_train, cv=5, scoring='accuracy')

# Evaluate the model
print("Random Forest Model:")
print("Best Parameters:", best_params_rf)
print("Cross-Validation Scores:", cv_scores_rf)
print("Mean CV Accuracy:", np.mean(cv_scores_rf))
```

```
Random Forest Model:
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 50}
Cross-Validation Scores: [0.74074074 0.62962963 0.77777778 0.81132075 0.66037736]
Mean CV Accuracy: 0.7239692522711391
```



```
# Evaluate the model
print("KNN Model:")
print("Best Parameters:", best_params_knn)
print("Accuracy:", accuracy_score(y_test, knn_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, knn_predictions))
print("Classification Report:\n", classification_report(y_test, knn_predictions))
```

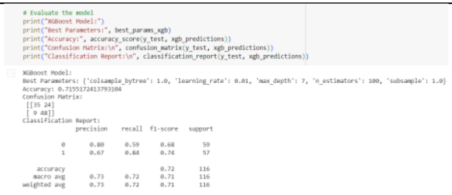
```
KNN Model:
Best Parameters: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
Accuracy: 0.6379310344827587
Confusion Matrix:
 [[30 29]
 [13 44]]
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.51      0.59        59
           1       0.60      0.77      0.68        57

    accuracy                           0.64       116
   macro avg       0.65      0.64      0.63       116
weighted avg       0.65      0.64      0.63       116
```



```
# Perform Cross-Validation
cv_scores_knn = cross_val_score(final_knn_model, x_train, y_train, cv=5, scoring='accuracy')

# Evaluate the model
print("KNN Model:")
print("Best Parameters:", best_params_knn)
print("Cross-Validation Scores:", cv_scores_knn)
print("Mean CV Accuracy:", np.mean(cv_scores_knn))
```

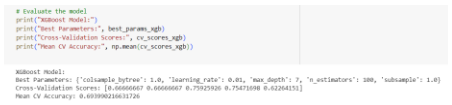
```
KNN Model:
Best Parameters: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
Cross-Validation Scores: [0.66666667 0.72222222 0.7962963  0.75471698 0.71698113]
Mean CV Accuracy: 0.7313766596785464
```

```
Cross-Validation Scores: [0.66666667 0.72222222
0.7962963  0.75471698 0.71698113]
Mean CV Accuracy: 0.7313766596785464


XG Boost:
Hyperparameter tuning:
XGBoost Model:
Best Parameters: {'colsample_bytree': 1.0,
'learning_rate': 0.01, 'max_depth': 7, 'n_estimators':
100, 'subsample': 1.0}
Accuracy: 0.7155172413793104
Confusion Matrix:
 [[35 24]
 [ 9 48]]
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.59      0.68        59
           1       0.67      0.84      0.74        57

    accuracy                           0.72       116
   macro avg       0.73      0.72      0.71       116
weighted avg       0.73      0.72      0.71       116

Validation:
XGBoost Model:
Best Parameters: {'colsample_bytree': 1.0,
'learning_rate': 0.01, 'max_depth': 7, 'n_estimators':
100, 'subsample': 1.0}
Cross-Validation Scores: [0.66666667 0.66666667
0.75925926 0.75471698 0.62264151]
Mean CV Accuracy: 0.693990216631726
```



```
# Evaluate the model
print("XGBoost Model:")
print("Best Parameters:", best_params_xgb)
print("Accuracy:", accuracy_score(y_test, xgb_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, xgb_predictions))
print("Classification Report:\n", classification_report(y_test, xgb_predictions))
```

```
XGBoost Model:
Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 100, 'subsample': 1.0}
Accuracy: 0.7155172413793104
Confusion Matrix:
 [[35 24]
 [ 9 48]]
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.59      0.68        59
           1       0.67      0.84      0.74        57

    accuracy                           0.72       116
   macro avg       0.73      0.72      0.71       116
weighted avg       0.73      0.72      0.71       116
```



```
# Evaluate the model
print("XGBoost Model:")
print("Best Parameters:", best_params_xgb)
print("Cross-Validation Scores:", cv_scores_xgb)
print("Mean CV Accuracy:", np.mean(cv_scores_xgb))
```

```
XGBoost Model:
Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 100, 'subsample': 1.0}
Cross-Validation Scores: [0.66666667 0.66666667 0.75925926 0.75471698 0.62264151]
Mean CV Accuracy: 0.693990216631726
```