# Model Optimization and Tuning Report

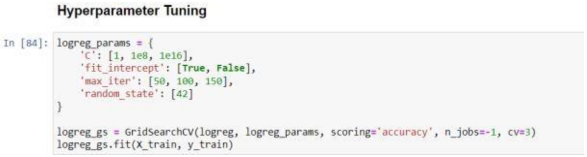
| | |
|---|---|
| Date | 15 April 2024 |
| Team ID | Team-738164 |
| Project Title | Rainfall Prediction Using Machine Learning |
| Maximum Marks | 10 Marks |

## Model Optimization and Tuning Phase:

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

## Hyperparameter Tuning Documentation (6 Marks):

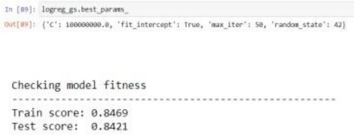
<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr>
<td>Logistic<br>Regression</td>
<td>



Hyperparameter Tuning

```
In [84]: logreg_params = {
             'C': [1, 1e8, 1e16],
             'fit_intercept': [True, False],
             'max_iter': [50, 100, 150],
             'random_state': [42]
         }

         logreg_gs = GridSearchCV(logreg, logreg_params, scoring='accuracy', n_jobs=-1, cv=3)
         logreg_gs.fit(X_train, y_train)
```

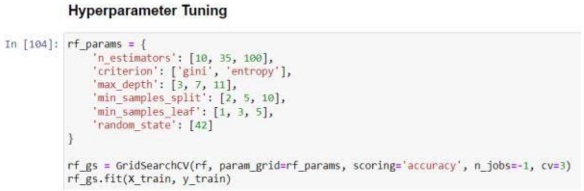
</td>
<td>



```
In [89]: logreg_gs.best_params_
Out[89]: {'C': 100000000.0, 'fit_intercept': True, 'max_iter': 50, 'random_state': 42}
```

```
Checking model fitness
-----------------------------------------------------------
Train score: 0.8469
Test score:  0.8421
```

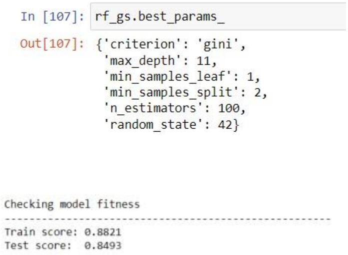
</td>
</tr>
<tr>
<td>Random Forest</td>
<td>



Hyperparameter Tuning

```
In [104]: rf_params = {
              'n_estimators': [10, 35, 100],
              'criterion': ['gini', 'entropy'],
              'max_depth': [3, 7, 11],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 3, 5],
              'random_state': [42]
          }

          rf_gs = GridSearchCV(rf, param_grid=rf_params, scoring='accuracy', n_jobs=-1, cv=3)
          rf_gs.fit(X_train, y_train)
```

</td>
<td>

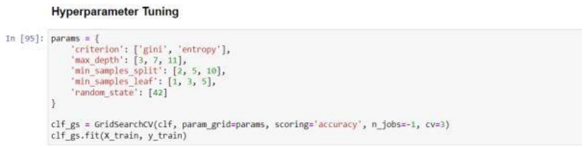


```
In [107]: rf_gs.best_params_
Out[107]: {'criterion': 'gini',
            'max_depth': 11,
            'min_samples_leaf': 1,
            'min_samples_split': 2,
            'n_estimators': 100,
            'random_state': 42}
```

```
Checking model fitness
-----------------------------------------------------------
Train score: 0.8821
Test score:  0.8493
```

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

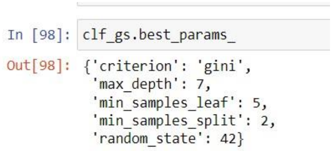
<table>
<tr>
<td>Decision Tree</td>
<td>



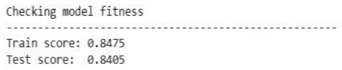
Hyperparameter Tuning

```
In [95]: params = {
             'criterion': ['gini', 'entropy'],
             'max_depth': [3, 7, 11],
             'min_samples_split': [2, 5, 10],
             'min_samples_leaf': [1, 3, 5],
             'random_state': [42]
         }

         clf_gs = GridSearchCV(clf, param_grid=params, scoring='accuracy', n_jobs=-1, cv=3)
         clf_gs.fit(X_train, y_train)
```

</td>
<td>

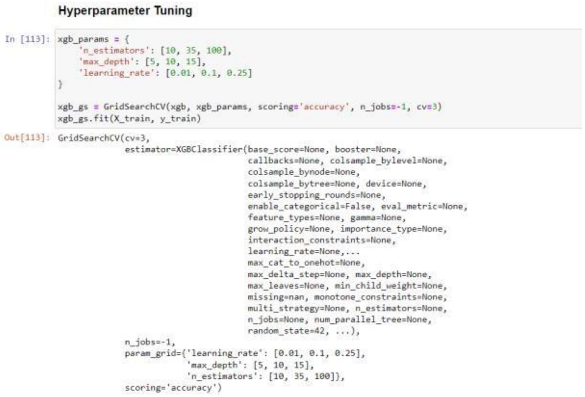


```
In [98]: clf_gs.best_params_

Out[98]: {'criterion': 'gini',
          'max_depth': 7,
          'min_samples_leaf': 5,
          'min_samples_split': 2,
          'random_state': 42}
```



```
Checking model fitness
------------------------------------------------------------
Train score: 0.8475
Test score:  0.8405
```

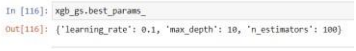
</td>
</tr>
<tr>
<td>XG-Boost</td>
<td>



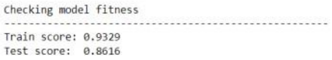
Hyperparameter Tuning

```
In [113]: xgb_params = {
              'n_estimators': [10, 35, 100],
              'max_depth': [5, 10, 15],
              'learning_rate': [0.01, 0.1, 0.25]
          }

          xgb_gs = GridSearchCV(xgb, xgb_params, scoring='accuracy', n_jobs=-1, cv=3)
          xgb_gs.fit(X_train, y_train)

Out[113]: GridSearchCV(cv=3,
                       estimator=XGBClassifier(base_score=None, booster=None,
                                               callbacks=None, colsample_bylevel=None,
                                               colsample_bynode=None,
                                               colsample_bytree=None, device=None,
                                               early_stopping_rounds=None,
                                               enable_categorical=False, eval_metric=None,
                                               feature_types=None, gamma=None,
                                               grow_policy=None, importance_type=None,
                                               interaction_constraints=None,
                                               learning_rate=None,...
                                               max_cat_to_onehot=None,
                                               max_delta_step=None, max_depth=None,
                                               max_leaves=None, min_child_weight=None,
                                               missing=nan, monotone_constraints=None,
                                               multi_strategy=None, n_estimators=None,
                                               n_jobs=None, num_parallel_tree=None,
                                               random_state=42, ...),
                       n_jobs=-1,
                       param_grid={'learning_rate': [0.01, 0.1, 0.25],
                                   'max_depth': [5, 10, 15],
                                   'n_estimators': [10, 35, 100]},
                       scoring='accuracy')
```

</td>
<td>



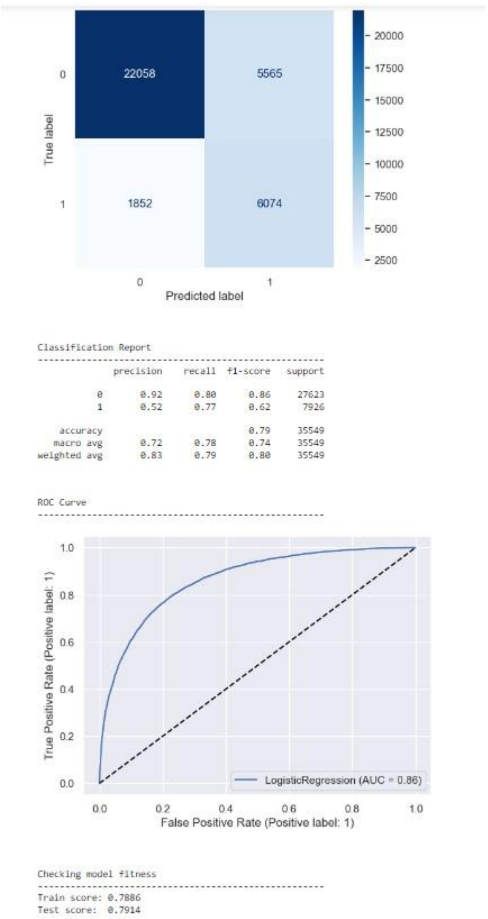
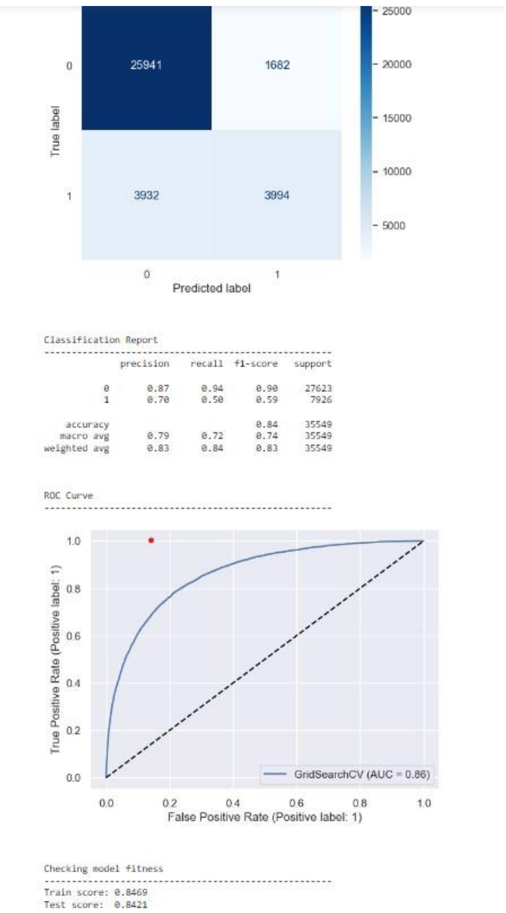
```
In [116]: xgb_gs.best_params_

Out[116]: {'learning_rate': 0.1, 'max_depth': 10, 'n_estimators': 100}
```

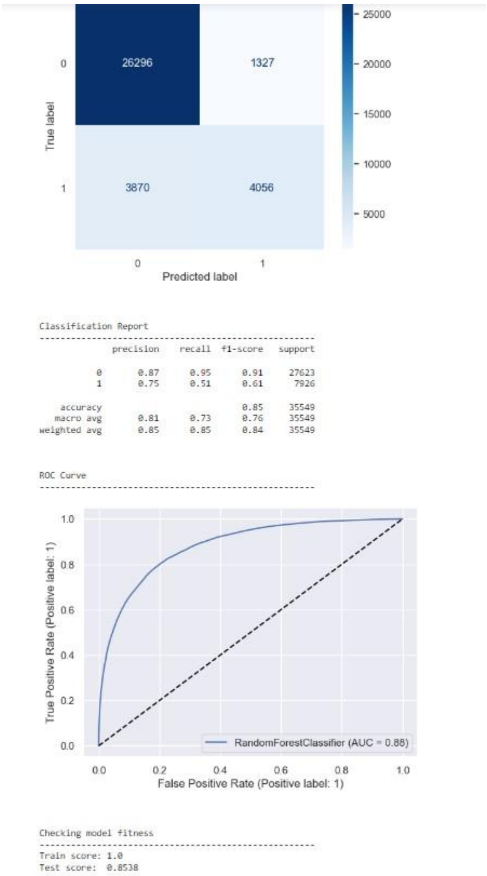
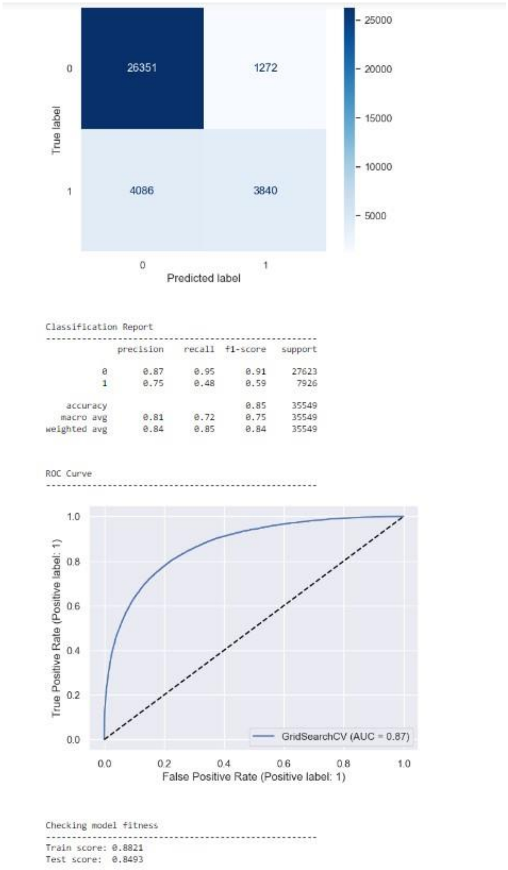


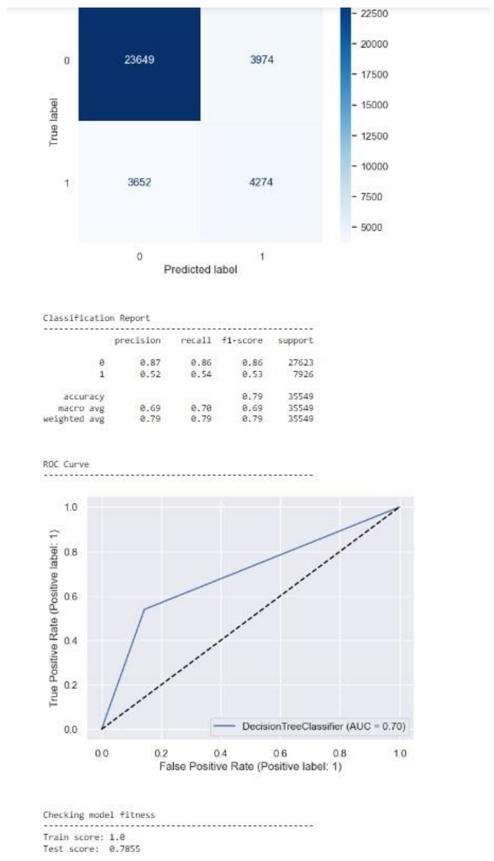
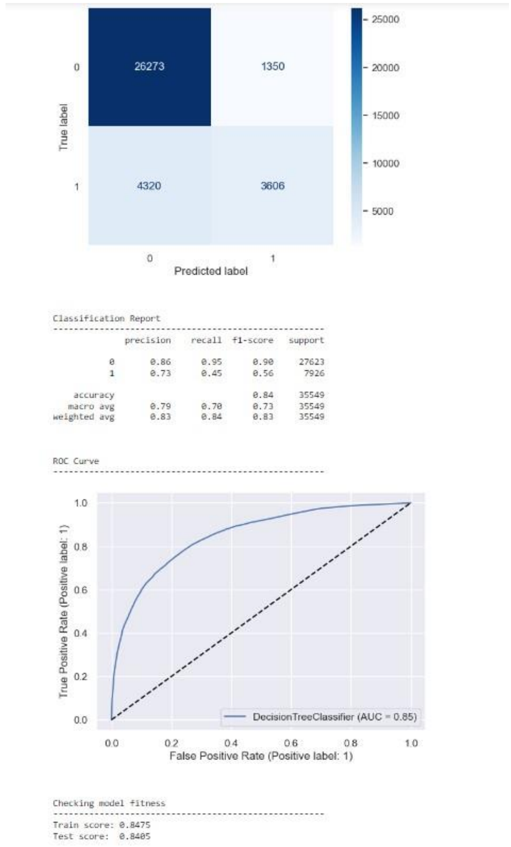
```
Checking model fitness
------------------------------------------------------------
Train score: 0.9329
Test score:  0.8616
```

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

## Performance Metrics Comparison Report (2 Marks):

| Model | Baseline Metric | Optimized Metric |
|---|---|---|
| Logistic<br>Regression | <br> True label<br>0<br>1<br>22058<br>5565<br>1852<br>6074<br>Predicted label<br>20000<br>17500<br>15000<br>12500<br>10000<br>7500<br>5000<br>2500<br>True Positive Rate (Positive label: 1)<br>False Positive Rate (Positive label: 1)<br>LogisticRegression (AUC = 0.86)<br>0.0<br>0.2<br>0.4<br>0.6<br>0.8<br>1.0<br>Classification Report<br>precision recall f1-score support<br>0 0.92 0.80 0.86 27623<br>1 0.52 0.77 0.62 7926<br>accuracy 0.79 35549<br>macro avg 0.72 0.78 0.74 35549<br>weighted avg 0.83 0.79 0.80 35549<br>ROC Curve<br>Checking model fitness<br>Train score: 0.7886<br>Test score: 0.7914<br>Accuracy: 79% AUC: 0.86 | <br> True label<br>0<br>1<br>25941<br>1682<br>3932<br>3994<br>Predicted label<br>25000<br>20000<br>15000<br>10000<br>5000<br>True Positive Rate (Positive label: 1)<br>False Positive Rate (Positive label: 1)<br>GridSearchCV (AUC = 0.86)<br>0.0<br>0.2<br>0.4<br>0.6<br>0.8<br>1.0<br>Classification Report<br>precision recall f1-score support<br>0 0.87 0.94 0.90 27623<br>1 0.70 0.50 0.59 7926<br>accuracy 0.84 35549<br>macro avg 0.79 0.72 0.74 35549<br>weighted avg 0.83 0.84 0.83 35549<br>ROC Curve<br>Checking model fitness<br>Train score: 0.8469<br>Test score: 0.8421<br>Accuracy: 84% AUC: 0.86 |

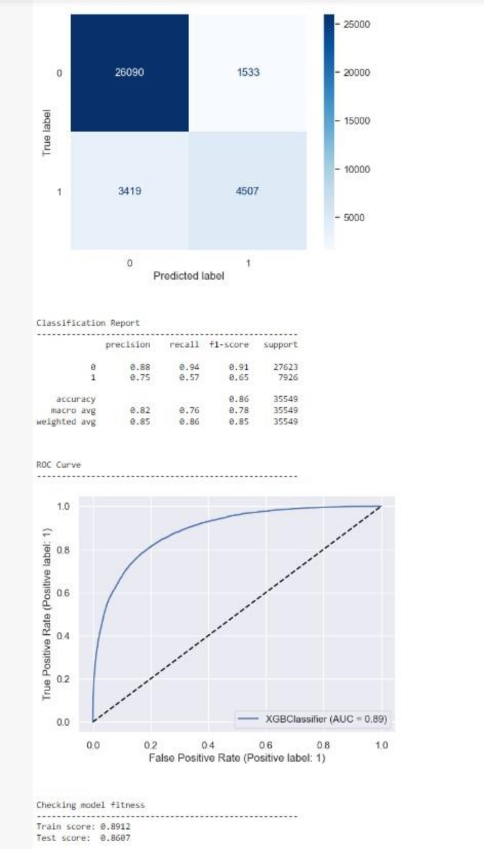
| | | |
|---|---|---|
| Random<br>Forest |  0<br>1<br>True label<br>26296<br>1327<br>3870<br>4056<br>0<br>1<br>Predicted label<br>25000<br>20000<br>15000<br>10000<br>5000<br>Classification Report<br>--------------------------------------------------------------<br>precision recall f1-score support<br>0 0.87 0.95 0.91 27623<br>1 0.75 0.51 0.61 7926<br>accuracy 0.85 35549<br>macro avg 0.81 0.73 0.76 35549<br>weighted avg 0.85 0.85 0.84 35549<br>ROC Curve<br>--------------------------------------------------------------<br> 1.0<br>0.8<br>0.6<br>0.4<br>0.2<br>0.0<br>True Positive Rate (Positive label: 1)<br>RandomForestClassifier (AUC = 0.88)<br>0.0<br>0.2<br>0.4<br>0.6<br>0.8<br>1.0<br>False Positive Rate (Positive label: 1)<br>Checking model fitness<br>--------------------------------------------------------------<br>Train score: 1.0<br>Test score:  0.8538<br>Accuracy: 85%     AUC: 0.88 |  0<br>1<br>True label<br>26351<br>1272<br>4086<br>3840<br>0<br>1<br>Predicted label<br>25000<br>20000<br>15000<br>10000<br>5000<br>Classification Report<br>--------------------------------------------------------------<br>precision recall f1-score support<br>0 0.87 0.95 0.91 27623<br>1 0.75 0.48 0.59 7926<br>accuracy 0.85 35549<br>macro avg 0.81 0.72 0.75 35549<br>weighted avg 0.84 0.85 0.84 35549<br>ROC Curve<br>--------------------------------------------------------------<br> 1.0<br>0.8<br>0.6<br>0.4<br>0.2<br>0.0<br>True Positive Rate (Positive label: 1)<br>GridSearchCV (AUC = 0.87)<br>0.0<br>0.2<br>0.4<br>0.6<br>0.8<br>1.0<br>False Positive Rate (Positive label: 1)<br>Checking model fitness<br>--------------------------------------------------------------<br>Train score: 0.8821<br>Test score:  0.8493<br>Accuracy: 85%     AUC: 0.87 |

|  |  |  |
|---|---|---|
| Decision Tree | <br> 23649<br>3974<br>3652<br>4274<br>True label<br>Predicted label<br>0<br>1<br>22500<br>20000<br>17500<br>15000<br>12500<br>10000<br>7500<br>5000<br>True Positive Rate (Positive label: 1)<br>False Positive Rate (Positive label: 1)<br>0.0<br>0.2<br>0.4<br>0.6<br>0.8<br>1.0<br>DecisionTreeClassifier (AUC = 0.70)<br>Classification Report<br>--------------------------------------------------------<br>precision recall f1-score support<br>0 0.87 0.86 0.86 27623<br>1 0.52 0.54 0.53 7926<br>accuracy 0.79 35549<br>macro avg 0.69 0.70 0.69 35549<br>weighted avg 0.79 0.79 0.79 35549<br>ROC Curve<br>--------------------------------------------------------<br>Checking model fitness<br>--------------------------------------------------------<br>Train score: 1.0<br>Test score:  0.7855<br><br>Accuracy: 79%     AUC: 0.70 | <br> 26273<br>1350<br>4320<br>3606<br>True label<br>Predicted label<br>0<br>1<br>25000<br>20000<br>15000<br>10000<br>5000<br>True Positive Rate (Positive label: 1)<br>False Positive Rate (Positive label: 1)<br>0.0<br>0.2<br>0.4<br>0.6<br>0.8<br>1.0<br>DecisionTreeClassifier (AUC = 0.85)<br>Classification Report<br>--------------------------------------------------------<br>precision recall f1-score support<br>0 0.86 0.95 0.90 27623<br>1 0.73 0.45 0.56 7926<br>accuracy 0.84 35549<br>macro avg 0.79 0.70 0.73 35549<br>weighted avg 0.83 0.84 0.83 35549<br>ROC Curve<br>--------------------------------------------------------<br>Checking model fitness<br>--------------------------------------------------------<br>Train score: 0.8475<br>Test score:  0.8405<br><br>Accuracy: 84%     AUC: 0.85 |

<table>
<tr>
<td>

XG-Boost

</td>
<td>



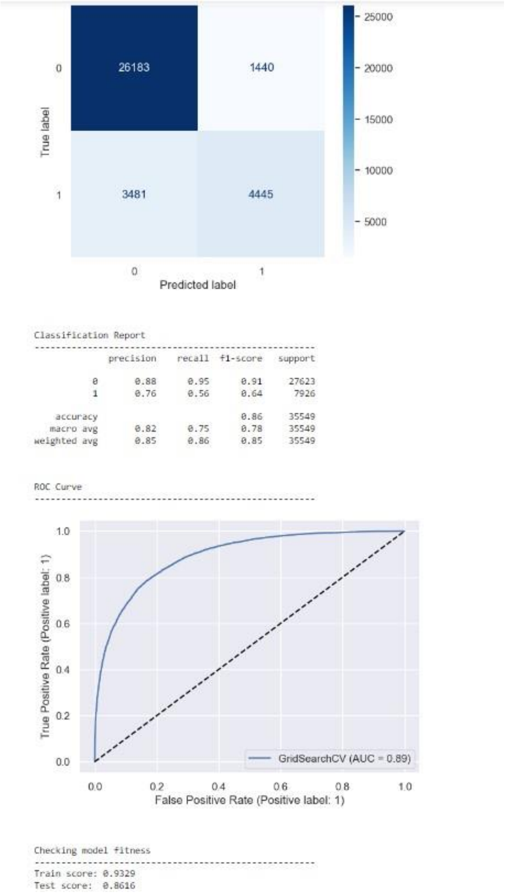
```
Classification Report
--------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     27623
           1       0.75      0.57      0.65      7926

    accuracy                           0.86     35549
   macro avg       0.82      0.76      0.78     35549
weighted avg       0.85      0.86      0.85     35549
```

```
ROC Curve
--------------------------------------------------------
```

```
Checking model fitness
--------------------------------------------------------
Train score: 0.8912
Test score:  0.8607
```

Accuracy: 86%     AUC: 0.89

</td>
<td>



```
Classification Report
--------------------------------------------------------
              precision    recall  f1-score   support

           0       0.88      0.95      0.91     27623
           1       0.76      0.56      0.64      7926

    accuracy                           0.86     35549
   macro avg       0.82      0.75      0.78     35549
weighted avg       0.85      0.86      0.85     35549
```

```
ROC Curve
--------------------------------------------------------
```

```
Checking model fitness
--------------------------------------------------------
Train score: 0.9329
Test score:  0.8616
```

Accuracy: 86%     AUC: 0.89

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

### Final Model Selection Justification (2 Marks):

| Final Model | Reasoning |
|---|---|
| XG-Boost | The best performing model is the hyperparameter-tuned XG-Boost model with an accuracy of approximately 86%. The scores for both the training and testing data were similar, reducing concerns of the model being overfit. |