Model Optimization and Tuning Phase Template

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):

Model	Tuned Hyperparameters	Optimal Values

Logistic Regression	**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)`
		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`

Random Forest	**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)`
		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`

<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>XGBoost</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')
```

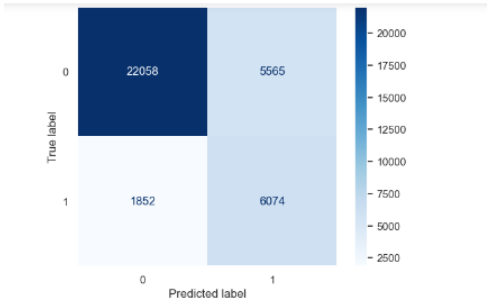
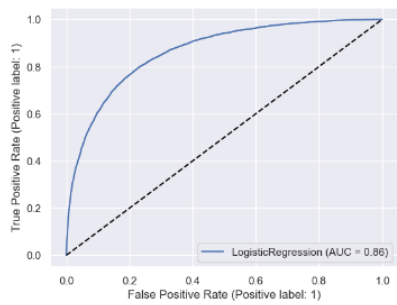
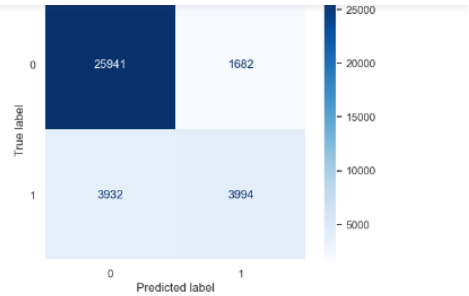
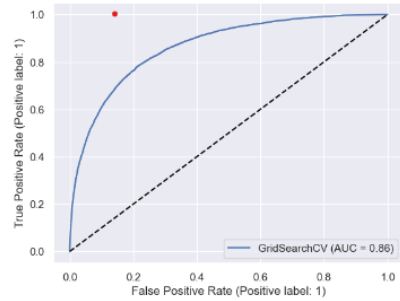
</td>
<td>

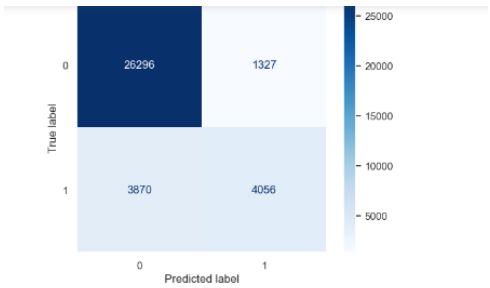
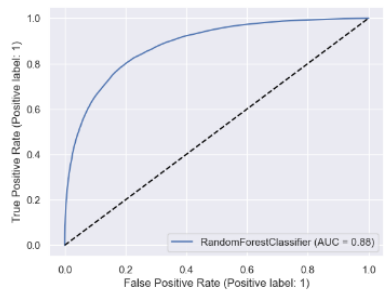
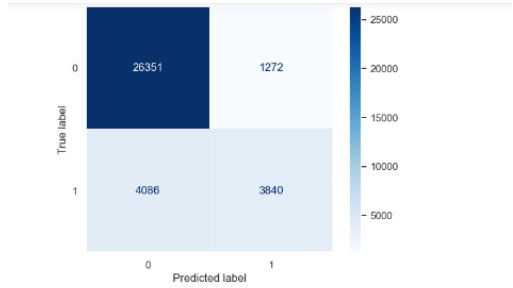
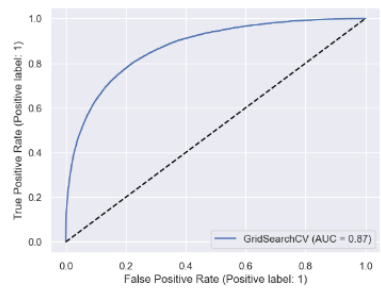
```
In [116]: xgb_gs.best_params_

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

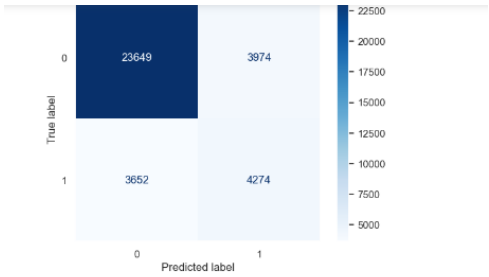
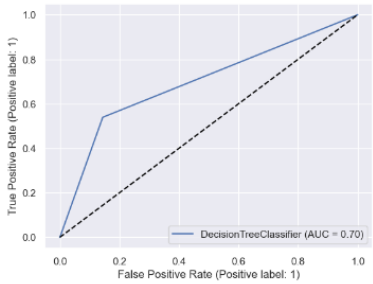
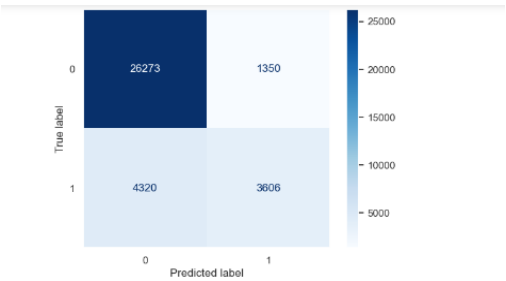
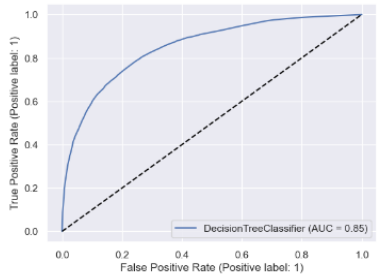
```
Checking model fitness
-----------------------------------------------------
Train score: 0.9329
Test score:  0.8616
```

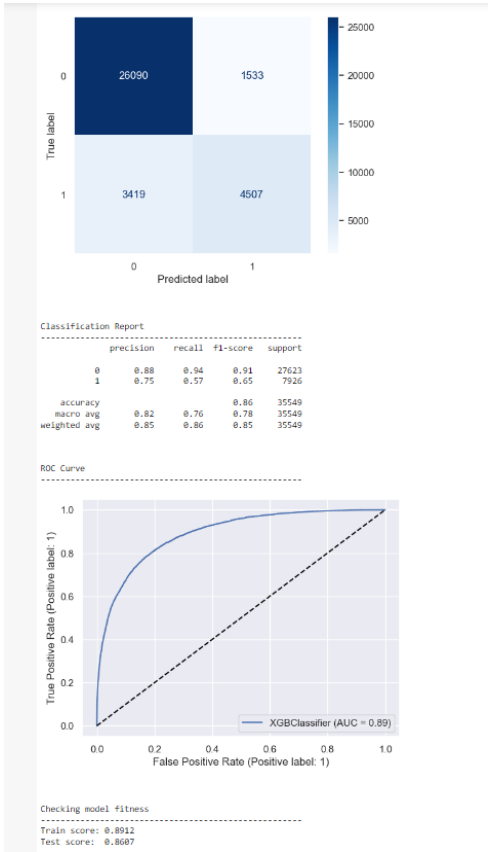
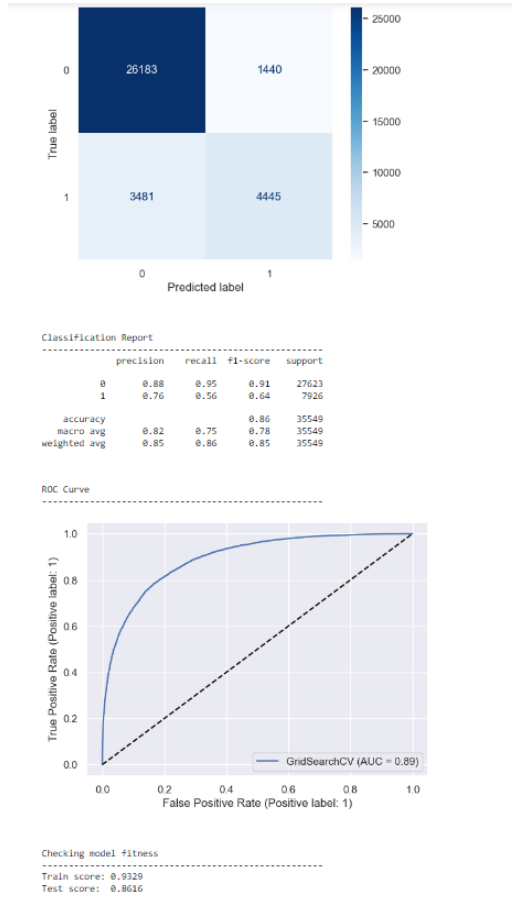
</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Logistic Regression	 Classification Report --- precision recall f1-score support 0 0.92 0.80 0.86 27623 1 0.52 0.77 0.62 7926 accuracy 0.79 35549 macro avg 0.72 0.78 0.74 35549 weighted avg 0.83 0.79 0.80 35549 ROC Curve ---  Checking model fitness --- Train score: 0.7886 Test score: 0.7914 Accuracy: 79% AUC: 0.86	 Classification Report --- precision recall f1-score support 0 0.87 0.94 0.90 27623 1 0.70 0.50 0.59 7926 accuracy 0.84 35549 macro avg 0.79 0.72 0.74 35549 weighted avg 0.83 0.84 0.83 35549 ROC Curve ---  Checking model fitness --- Train score: 0.8469 Test score: 0.8421 Accuracy: 84% AUC: 0.86

Random Forest	 Classification Report ---------- precision recall f1-score support 0 0.87 0.95 0.91 27623 1 0.75 0.51 0.61 7926 accuracy 0.85 35549 macro avg 0.81 0.73 0.76 35549 weighted avg 0.85 0.85 0.84 35549 ROC Curve ----------  Checking model fitness ---------- Train score: 1.0 Test score: 0.8538 Accuracy: 85% AUC: 0.88	 Classification Report ---------- precision recall f1-score support 0 0.87 0.95 0.91 27623 1 0.75 0.48 0.59 7926 accuracy 0.85 35549 macro avg 0.81 0.72 0.75 35549 weighted avg 0.84 0.85 0.84 35549 ROC Curve ----------  Checking model fitness ---------- Train score: 0.8821 Test score: 0.8493 Accuracy: 85% AUC: 0.87

Decision Tree	 Classification Report precision recall f1-score support 0 0.87 0.86 0.86 27623 1 0.52 0.54 0.53 7926 accuracy 0.79 35549 macro avg 0.69 0.70 0.69 35549 weighted avg 0.79 0.79 0.79 35549 ROC Curve  Checking model fitness Train score: 1.0 Test score: 0.7855 Accuracy: 79% AUC: 0.70	 Classification Report precision recall f1-score support 0 0.86 0.95 0.90 27623 1 0.73 0.45 0.56 7926 accuracy 0.84 35549 macro avg 0.79 0.70 0.73 35549 weighted avg 0.83 0.84 0.83 35549 ROC Curve  Checking model fitness Train score: 0.8475 Test score: 0.8485 Accuracy: 84% AUC: 0.85

XGBoost	 Accuracy: 86% AUC: 0.89	 Accuracy: 86% AUC: 0.89

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
XGBoost	The best performing model is the hyperparameter-tuned XGBoost 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.