

Model Optimization and Tuning Phase Template

Date	24 April 2024
Team ID	Team-738169
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
Random Forest	<pre># Create the random grid random_grid={'n_estimators':n_estimators, 'max_features':max_features, 'max_depth':max_depth, 'min_samples_split':min_samples_split, 'min_samples_leaf':min_samples_leaf, 'criterion':['entropy','gini']} print(random_grid) {'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600, 2800, 3000], 'max_features': ['sqrt', 'log2', 'max_features'], 'max_depth': [None, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100], 'min_samples_split': [2, 5, 10, 14], 'min_samples_leaf': [1, 2, 4, 8, 16]} rf_randomCV.best_params_ {'n_estimators': 600, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 120, 'criterion': 'entropy'}</pre>	<pre>from sklearn.metrics import accuracy_score y_pred = best_random_grid.predict(X_test) print(confusion_matrix(y_test,y_pred)) print("Accuracy score {}".format(accuracy_score(y_test,y_pred))) print("Classification report {}".format(classification_report(y_test,y_pred))) [[1686 172] [221 321]] Accuracy score 0.83625</pre>

XGBoost	<pre># Create the random grid random_grid = {'n_estimators': n_estimators, 'learning_rate': learning_rate, 'max_depth': max_depth, 'subsample': subsample, 'min_child_weight': min_child_weight} print(random_grid) {'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800], 'h': [5, 10, 15, 20, 25, 30], 'subsample': [0.7, 0.6, 0.8]} xg_random.best_params_ {'subsample': 0.6, 'n_estimators': 400, 'min_child_weight': 5, 'max_depth': 25, 'learning_rate': '0.05'}</pre>	<pre>from sklearn.metrics import accuracy_score y_predict = xg_random.predict(X_test) print(confusion_matrix(y_test,y_predict)) print('Accuracy score {}'.format(accuracy_score(y_test,y_predict))) print('Classification report {}'.format(classification_report(y_test,y_predict))) [[1716 154] [218 312]] Accuracy score 0.845</pre>
---------	---	--

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric																														
Logistic Regression	<pre>y_pred2 = logreg.predict(X_test) print(confusion_matrix(y_test,y_pred2)) print(accuracy_score(y_test,y_pred2)) print(classification_report(y_test,y_pred2))</pre> <pre>[[17456 5261] [1508 4867]] 0.7673243503368624</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.92</td><td>0.77</td><td>0.84</td><td>22717</td></tr><tr><td>1</td><td>0.48</td><td>0.76</td><td>0.59</td><td>6375</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.77</td><td>29092</td></tr><tr><td>macro avg</td><td>0.70</td><td>0.77</td><td>0.71</td><td>29092</td></tr><tr><td>weighted avg</td><td>0.82</td><td>0.77</td><td>0.78</td><td>29092</td></tr></tbody></table>		precision	recall	f1-score	support	0	0.92	0.77	0.84	22717	1	0.48	0.76	0.59	6375	accuracy			0.77	29092	macro avg	0.70	0.77	0.71	29092	weighted avg	0.82	0.77	0.78	29092
	precision	recall	f1-score	support																											
0	0.92	0.77	0.84	22717																											
1	0.48	0.76	0.59	6375																											
accuracy			0.77	29092																											
macro avg	0.70	0.77	0.71	29092																											
weighted avg	0.82	0.77	0.78	29092																											

Decision Tree	<pre>y_pred = model_dt.predict(X_test) print(confusion_matrix(y_test,y_pred)) print(accuracy_score(y_test,y_pred)) print(classification_report(y_test,y_pred))</pre> <pre>[[1616 240] [236 308]] 0.8016666666666666</pre> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.87</td><td>0.87</td><td>0.87</td><td>1856</td></tr><tr><td>1</td><td>0.56</td><td>0.57</td><td>0.56</td><td>544</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.80</td><td>2400</td></tr><tr><td>macro avg</td><td>0.72</td><td>0.72</td><td>0.72</td><td>2400</td></tr><tr><td>weighted avg</td><td>0.80</td><td>0.80</td><td>0.80</td><td>2400</td></tr></table>		precision	recall	f1-score	support	0	0.87	0.87	0.87	1856	1	0.56	0.57	0.56	544	accuracy			0.80	2400	macro avg	0.72	0.72	0.72	2400	weighted avg	0.80	0.80	0.80	2400
	precision	recall	f1-score	support																											
0	0.87	0.87	0.87	1856																											
1	0.56	0.57	0.56	544																											
accuracy			0.80	2400																											
macro avg	0.72	0.72	0.72	2400																											
weighted avg	0.80	0.80	0.80	2400																											
Random Forest	<pre>y_pred1 = rf.predict(X_test) print(confusion_matrix(y_test,y_pred1)) print(accuracy_score(y_test,y_pred1)) print(classification_report(y_test,y_pred1))</pre> <pre>[[1683 175] [225 317]] 0.8333333333333334</pre> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.88</td><td>0.91</td><td>0.89</td><td>1858</td></tr><tr><td>1</td><td>0.64</td><td>0.58</td><td>0.61</td><td>542</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.83</td><td>2400</td></tr><tr><td>macro avg</td><td>0.76</td><td>0.75</td><td>0.75</td><td>2400</td></tr><tr><td>weighted avg</td><td>0.83</td><td>0.83</td><td>0.83</td><td>2400</td></tr></table>		precision	recall	f1-score	support	0	0.88	0.91	0.89	1858	1	0.64	0.58	0.61	542	accuracy			0.83	2400	macro avg	0.76	0.75	0.75	2400	weighted avg	0.83	0.83	0.83	2400
	precision	recall	f1-score	support																											
0	0.88	0.91	0.89	1858																											
1	0.64	0.58	0.61	542																											
accuracy			0.83	2400																											
macro avg	0.76	0.75	0.75	2400																											
weighted avg	0.83	0.83	0.83	2400																											
KNN	<pre>y_pred4 = knn.predict(X_test) print(confusion_matrix(y_test,y_pred4)) print(accuracy_score(y_test,y_pred4)) print(classification_report(y_test,y_pred4))</pre> <pre>[[17410 5307] [1811 4564]] 0.7553279252028049</pre> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.91</td><td>0.77</td><td>0.83</td><td>22717</td></tr><tr><td>1</td><td>0.46</td><td>0.72</td><td>0.56</td><td>6375</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.76</td><td>29092</td></tr><tr><td>macro avg</td><td>0.68</td><td>0.74</td><td>0.70</td><td>29092</td></tr><tr><td>weighted avg</td><td>0.81</td><td>0.76</td><td>0.77</td><td>29092</td></tr></table>		precision	recall	f1-score	support	0	0.91	0.77	0.83	22717	1	0.46	0.72	0.56	6375	accuracy			0.76	29092	macro avg	0.68	0.74	0.70	29092	weighted avg	0.81	0.76	0.77	29092
	precision	recall	f1-score	support																											
0	0.91	0.77	0.83	22717																											
1	0.46	0.72	0.56	6375																											
accuracy			0.76	29092																											
macro avg	0.68	0.74	0.70	29092																											
weighted avg	0.81	0.76	0.77	29092																											

SVM

```
: y_pred5 = svc.predict(X_test)
print(confusion_matrix(y_test,y_pred5))
print(accuracy_score(y_test,y_pred5))
print(classification_report(y_test,y_pred5))
```

```
[[1436  463]
 [ 137  364]]
0.75
```

	precision	recall	f1-score	support
0	0.91	0.76	0.83	1899
1	0.44	0.73	0.55	501
accuracy			0.75	2400
macro avg	0.68	0.74	0.69	2400
weighted avg	0.81	0.75	0.77	2400

XGBoost

```
y_pred = xgb.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[1709  161]
 [ 218  312]]
0.8420833333333333
```

	precision	recall	f1-score	support
0	0.89	0.91	0.90	1870
1	0.66	0.59	0.62	530
accuracy			0.84	2400
macro avg	0.77	0.75	0.76	2400
weighted avg	0.84	0.84	0.84	2400

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
XGBoost	Thorough experimentation and hyperparameter tuning, the XGBoost model consistently outperformed other models in terms of accuracy metrics. XGBoost's inherent capability to handle missing values and its regularization techniques contribute to its superior performance and generalization ability, justifying its selection as the final model.