



Model Optimization and Tuning Phase Template

Date	04 October 2024
Team ID	LTVIP2024TMID24922
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
	<pre>rf=RandomForestClassifier() rf.fit(X_train_res,y_train_res)</pre>	
Random Forest	<pre># Number of trees in random forest n_estimators=[int(x) for x in np.linspace(start=200,stop=2000,num=10)] # Number of features to consider at every split max_features=['auto','sqrt', 'log2'] # Maximum number of levels in tree max_depth=[int(x) for x in np.linspace(10,1000,10)]</pre>	from skleam.metrics import accuracy_score
	<pre># Minimum number of samples required to split a node min_samples_split=[2,5,10,14] # Minimum number of samples required at each leaf node min_samples_leaf=[1,2,4,6,8] # Create the random grid random grid={'n_estimators':n_estimators,</pre>	<pre>print('Classification report {}'.format(classification_report(y_test,y_pred)))</pre>
	<pre>'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)</pre>	

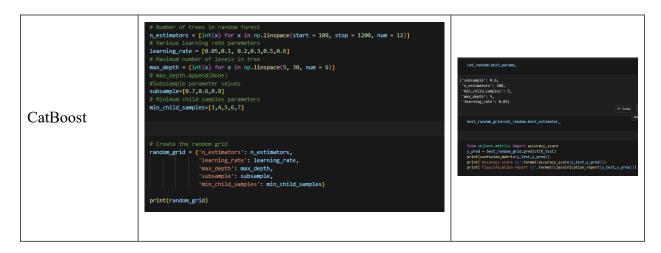




Decision Tree	<pre># Setup the parameters and distributions to sample from: param_dist param_dist = {"max_depth": [3, None],</pre>	from sklearn.metrics import accorney score y_mred_tree = tree_cv.redict(2_test) print((Compain_matrix(y_test),yred_tree)) print((Compain_matrix(y_test),yred_tree)) print((Classification_report ()'.format(classification_report(y_test,y_pred_tree))) print((Classification_report ()'.format(classification_report(y_test,y_pred_tree)))
K-Neighbors Classifier	<pre>knn = KNeighborsClassifier(n_neighbors=3) knn.fit(X_train_res, y_train_res)</pre>	<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>
Logestic Regression	<pre>logreg = LogisticRegression() logreg.fit(X_train_res, y_train_res)</pre>	<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>
XGBoost	<pre># Number of trees in random forest n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)] # Various learning rate parameters learning_rate = ['0.05','0.1', '0.2','0.3','0.5','0.6'] # Naximum number of levels in tree max_depth = [int(x) for x in np.linspace(5, 30, num = 6)] # max_depth = [int(x) for x in np.linspace(5, 30, num = 6)] # Subsample parameter values subsample=[0.7,0.6,0.8] # Minimum child weight parameters min_child_weight=[3,4,5,6,7] # Create the random grid random_grid = ('n_estimators': n_estimators,</pre>	from asteron.metrics. Import accoracy_score year accoracy
SVC	<pre>svc = SVC() svc.fit(X_train_res, y_train_res)</pre>	<pre>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))</pre>







Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric			
	<pre>print('Classification report {}'.format(classification_report(y_test,y_pred)))</pre>			
Random Forest	Classification report precision recall f1-score support 0 0.89 0.89 0.89 1897 1 0.58 0.58 503 accuracy 0.74 0.73 0.73 2400 weighted avg 0.82 0.82 0.82 2400 print(confusion_matrix(y_test,y_pred)) [[1690 207] [213 290]]			
Decision Tree	print('Classification report {} '.format(classification_report(y_test,y_pred_tree))) Classification report			

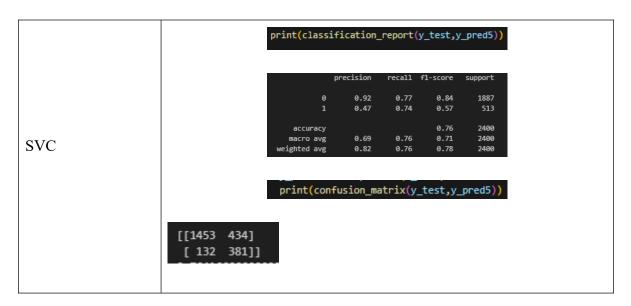




	<pre>print(classification_report(y_test,y_pred4))</pre>				
		precision	recall	f1-score	support
	0 1	0.91 0.46	0.77 0.72	0.83 0.56	22717 6375
K-Neighbors	accuracy macro avg	0.68		0.76 0.70	29092 29092
Classifier	weighted avg	0.81	0.76	0.77	29092
	print(co	nfusion_ma	ntrix(y_	test,y	pred4))
			9 530 8 456		
	print(clas	sification	_report(<u>)</u>	_test,y_	pred2))
		precision		f1-score	support
	0 1		0.77 0.76	0.84 0.59	22717 6375
Logistic Regression	accuracy macro avg weighted avg	0.70	0.77 0.77	0.77 0.71 0.78	29092 29092 29092
	print(co	nfusion_ma	atrix(y	_test,y_	pred2))
		[[17439 [1507			
	<pre>print('Classificati</pre>	on report {}'.form	t(classificati	on report/v tes	t.v predict)))
	W 2014 (22052) 12002	report [] From	.,		-37_b.cozcc///
	Classification rep	0.87 0.93 0.68 0.52	0.90 1	recall f1-scor 1874 526	e support
XGBoost	accuracy macro avg	0.78 0.73 0.83 0.84	0.84 2 0.75 2	2400 2400 2400	
11300001	print(conf	usion_mat	rix(y_te	est,y_pr	edict))
			5 129] 9 276]		







Final Model Selection Justification (2 Marks):

Final Model	Reasoning			
	The Random Forest model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its			
Random Forest	selection as the final model			