Project Development Phase
Model Performance Test

Date	18 November 2022
Team ID	PNT2022TMID23758
Project Name	Project – Efficient Water Quality Analysis and Prediction Using Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

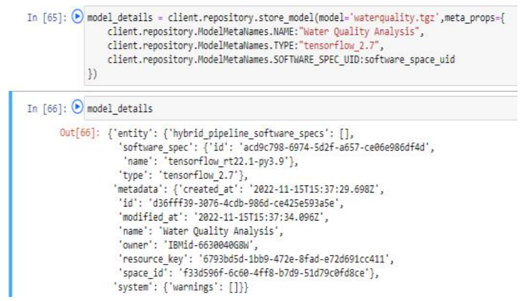
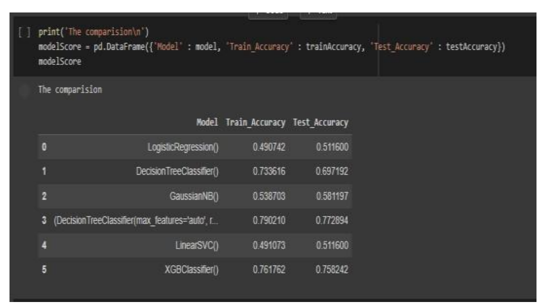
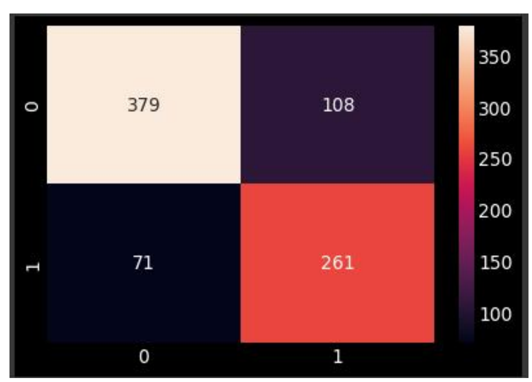
Project team shall fill the following information in model performance testing template.

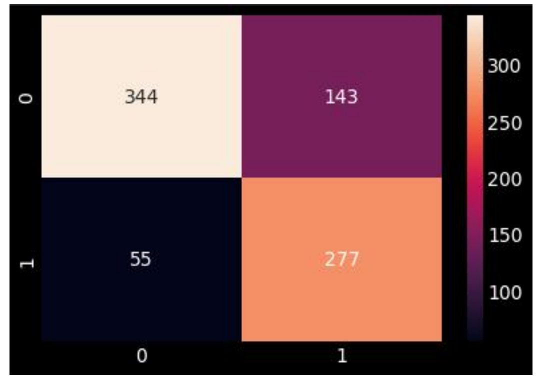
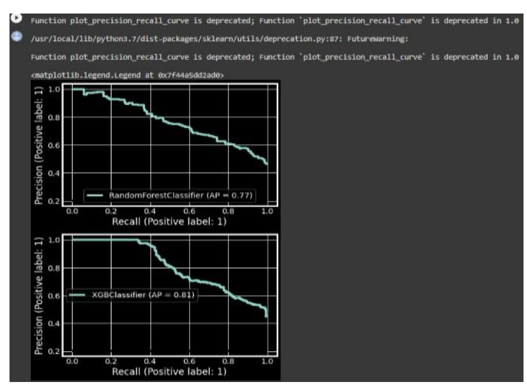
S.No.	Parameter	Values	Screenshot

1.	Regression Model	from sklearn.ensemble import RandomForestRegressor
		regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
		regressor.fit(x_train, y_train)
		y_pred = regressor.predict(x_test)
		
		from sklearn import metrics
		print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
		print('MSE:',metrics.mean_squared_error(y_test,y_pred))
		print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
		
		MAE: 1.013774436090232
		MSE: 6.2406858345864675
		RMSE: 2.498136472370248
		#accuracy of the model
		metrics.r2_score(y_test, y_pred)
		0.9659820315121997
			
			from sklearn import metrics
			print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
			print('MSE:',metrics.mean_squared_error(y_test,y_pred))
			print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
			MAE: 1.013774436090232
			MSE: 6.2406858345864675
			RMSE: 2.498136472370248
			#accuracy of the model
			metrics.r2_score(y_test, y_pred)
			0.9659820315121997

2.	Hyperparameter tuning	SPACE = [
		    skopt.space.Real(0.01, 0.5, name='learning_rate', prior='log-uniform'),
		    skopt.space.Integer(1, 30, name='max_depth'),
		    skopt.space.Integer(2, 100, name='num_leaves'),
		    skopt.space.Real(0.1, 1.0, name='feature_fraction', prior='uniform'),
		    skopt.space.Real(0.1, 1.0, name='subsample', prior='uniform')]
		@skopt.utils.use_named_args(SPACE)
		def objective(**params):
		    return -1.0 * train_evaluate(params)
		results = skopt.forest_minimize(objective, SPACE, n_calls=30, n_random_starts=10)
		best_auc = -1.0 * results.fun
		best_params = results.x
		print('best result: ', best_auc)
		print('best parameters: ', best_params)
			best result: 0.6509559162948146
			best parameters: [0.014509467657194726, 21, 26, 0.9723402117733363, 0.6065207062490089]

3.	Validation Method	def train_evaluate(search_params):
		   path = "water_potability.csv"
		   data = pd.read_csv(path)
		   X = data.drop(['Sulfate','Potability'], axis=1)
		   y = data['Potability']
		   X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=1234)
		   train_data = lgb.Dataset(X_train, label=y_train)
		   valid_data = lgb.Dataset(X_valid, label=y_valid, reference=train_data)
		
		   params = {'objective': 'binary',
		             'metric': 'auc',
		             **search_params}
		   model = lgb.train(params, train_data,
		                     num_boost_round=300,
		                     early_stopping_rounds=30,
		                     valid_sets=[valid_data],
		                     valid_names=['valid'])
		
		   score = model.best_score['valid']['auc']
		   return score
		if __name__== '__main__':
		    score = train_evaluate(SEARCH_PARAMS)
		    print('validation AUC:', score)
			validation AUC: 0.6509559162948146

4.	Model Summary	{'entity': {'hybrid_pipeline_software_specs': [], 'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d', 'name': 'tensorflow_rt22.1-py3.9'}, 'type': 'tensorflow_2.7'}, 'metadata': {'created_at': '2022-11-15T15:37:29.698Z', 'id': 'd36fff39-3076-4cdb-986d-ce425e593a5e', 'modified_at': '2022-11-15T15:37:34.096Z', 'name': 'Water Quality Analysis', 'owner': 'IBMid-6630040G8W', 'resource_key': '6793bd5d-1bb9-472e-8fad-e72d691cc411', 'space_id': 'f33d596f-6c60-4ff8-b7d9-51d79c0fd8ce'}, 'system': {'warnings': []}}	
5.	Accuracy	Model Train_Accuracy Test_Accuracy 0 LogisticRegression() 0.490742 0.511600 1 DecisionTreeClassifier() 0.733616 0.697192 2 GaussianNB() 0.538703 0.581197 3 (DecisionTreeClassifier(max_features='auto', r... 0.790210 0.772894 4 LinearSVC() 0.491073 0.511600 5 XGBClassifier() 0.761762 0.758242	
6.	Confusion Matrix (Random Forest Classifier)	print('Random Forest Classifier\n') Rfc = RandomForestClassifier() Rfc.fit(X_train, y_train) y_Rfc = Rfc.predict(X_test) print(metrics.classification_report(y_test, y_Rfc)) print(modelAccuracy.append(metrics.accuracy_score(y_test, y_Rfc))) sns.heatmap(confusion_matrix(y_test, y_Rfc), annot=True, fmt='d') plt.show()	

	Confusion Matrix (XGB Classifier)	print('XGB Classifier\n') xgb = XGBClassifier() xgb.fit(X_train, y_train) y_xgb = xgb.predict(X_test) print(metrics.classification_report(y_test, y_xgb)) print(modelAccuracy.append(metrics.accuracy_score(y_test, y_xgb))) sns.heatmap(confusion_matrix(y_test, y_xgb), annot=True, fmt='d') plt.show()	
7.	Precision Recall F1 Score (Random Forest Classifier)	print('Random Forest Classifier\n') Rfc = RandomForestClassifier() Rfc.fit(X_train, y_train) y_Rfc = Rfc.predict(X_test) print(metrics.classification_report(y_test, y_Rfc)) print(modelAccuracy.append(metrics.accuracy_score(y_test, y_Rfc)))	Random Forest Classifier precision recall f1-score support 0 0.84 0.78 0.81 487 1 0.71 0.79 0.74 332 accuracy 0.78 819 macro avg 0.77 0.78 0.78 819 weighted avg 0.79 0.78 0.78 819
	Precision Recall F1 Score (XGB Classifier)	print('XGB Classifier\n') xgb = XGBClassifier() xgb.fit(X_train, y_train) y_xgb = xgb.predict(X_test) print(metrics.classification_report(y_test, y_xgb)) print(modelAccuracy.append(metrics.accuracy_score(y_test, y_xgb)))	XGB Classifier precision recall f1-score support 0 0.86 0.71 0.78 487 1 0.66 0.83 0.74 332 accuracy 0.76 819 macro avg 0.76 0.77 0.76 819 weighted avg 0.78 0.76 0.76 819
8.	Precision- Recall or PR curve	from scikitplot.metrics import plot_roc_curve from sklearn.metrics import plot_precision_recall_curve plot_precision_recall_curve(Rfc,X_test,y_test) plt.plot([0,1], [0.2035,0.2035], c='k') plt.legend(loc='best') plot_precision_recall_curve(xgb,X_test,y_test) plt.plot([0,1], [0.2035,0.2035], c='k') plt.legend(loc='best')	

9.	PR vs ROC curve	plot_roc_curve(Rfc,X_test,y_test) plot_roc_curve(xgb,X_test,y_test)