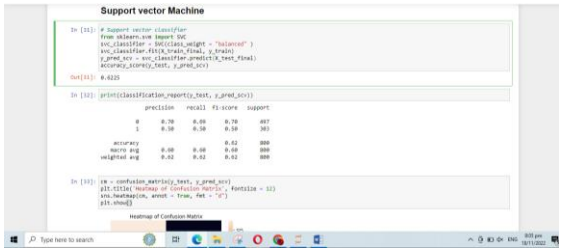
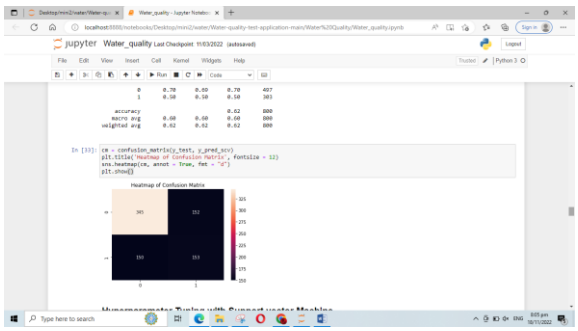
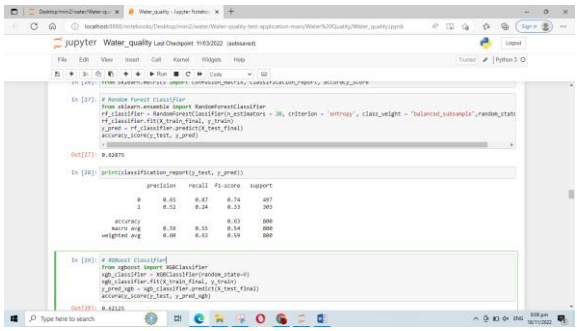
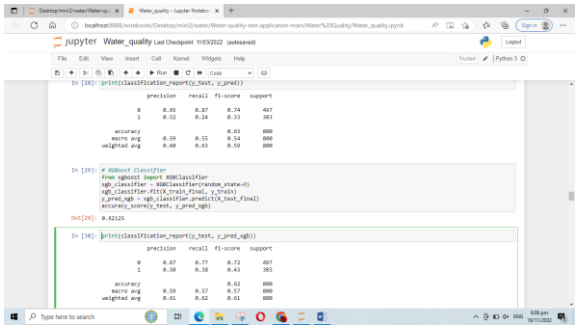
Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID13654
Project Name	Efficient water quality analysis and prediction using machine learning
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** # Support vector classifier from sklearn.svm import SVC svc_classifier = SVC(class_weight = "balanced") svc_classifier.fit(X_train_final, y_train) y_pred_scv = svc_classifier.predict(X_test_final) **ACCURACY:** accuracy_score(y_test, y_pred_scv) #0.62255 **CLASSIFICATION REPORT :** print(classification_report(y_test, y_pred_scv)) **CONFUSION MATRIX:** cm = confusion_matrix(y_test, y_pred_scv) plt.title('Heatmap of Confusion Matrix', fontsize = 12) sns.heatmap(cm, annot = True, fmt = "d") plt.show()	 

		# Random Forest Classifier from sklearn.ensemble import RandomForestClassifier rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', class_weight = "balanced_subsample",random_state = 51) rf_classifier.fit(X_train_final, y_train) y_pred = rf_classifier.predict(X_test_final) **ACCURACY_SCORE:** accuracy_score(y_test, y_pred) **CLASSIFICATION REPORT:** print(classification_report(y_test, y_pred)) **# XGBoost Classifier** from xgboost import XGBClassifier xgb_classifier = XGBClassifier(random_state=0) xgb_classifier.fit(X_train_final, y_train) y_pred_xgb = xgb_classifier.predict(X_test_final) **ACCURACY_SCORE:** accuracy_score(y_test, y_pred_xgb) **CLASSIFICATION REPORT:** print(classification_report(y_test, y_pred_xgb))	 
2.	**Tune the Model**	**Hyperparameter Tuning** param_grid = {'C': [0.1, 1, 10, 100, 200 , 400 , 600 , 800], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbf']}	

		from sklearn.model_selection import GridSearchCV
		grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)
		
		# fitting the model for grid search
		grid.fit(X_train_final, y_train)
		# print best parameter after tuning
		print(grid.best_params_)
		
		# print how our model looks after hyper-parameter tuning
		print(grid.best_estimator_)