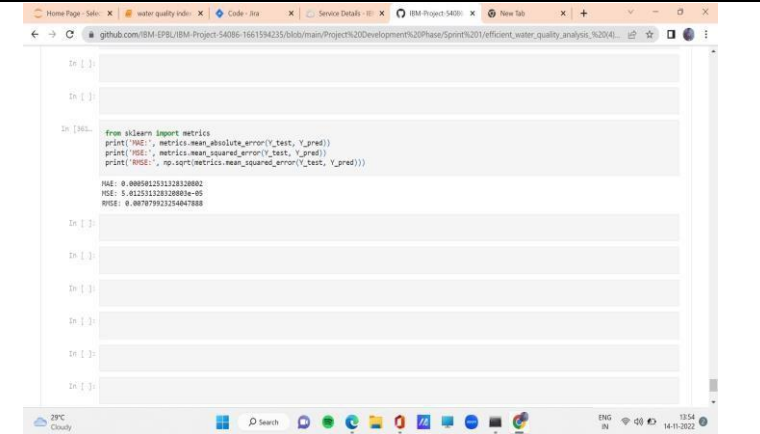
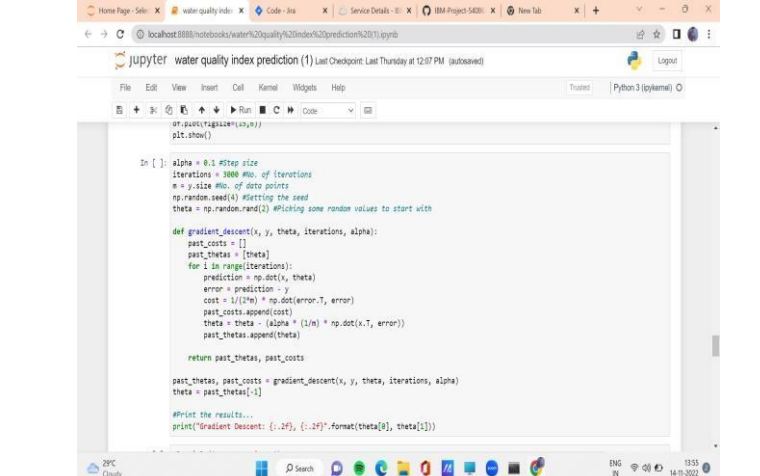


**Project Development Phase**  
**Model Performance Test**

Date	November 2022
Team ID	PNT2022TMID287
Project Name	Efficient water quality analysis & prediction using ML
Maximum Marks	Marks

L

**Model Performance Testing:**

S.No.	Parameter	Values	Screenshot
1.	Metrics	<p><b>Regression Model:</b> MAE - , MSE - , RMSE - , R2 score -</p> <p><b>Classification Model:</b> Confusion Matrix - , Accuray Score&amp; Classification Report -</p>	 <pre> In [ ]:  In [ ]:  In [36]: 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: 0.000981231232830802 MSE: 5.812513283208004e-05 RMSE: 0.007879932325404788 </pre>
2.	Tune the Model	Hyperparameter Tuning - Validation Method- Gradient descent	 <pre> In [ ]: alpha = 0.1 #Step size iterations = 3000 #No. of iterations n = y.size #No. of data points np.random.seed(4) #Setting the seed theta = np.random.rand(2) #Picking some random values to start with  def gradient_descent(x, y, theta, iterations, alpha):     past_costs = []     past_thetas = [theta]     for i in range(iterations):         prediction = np.dot(x, theta)         error = prediction - y         cost = 1/(2*n) * np.dot(error.T, error)         past_costs.append(cost)         theta = theta - (alpha * (1/n) * np.dot(x.T, error))         past_thetas.append(theta)     return past_thetas, past_costs  past_thetas, past_costs = gradient_descent(x, y, theta, iterations, alpha) theta = past_thetas[-1]  #Print the results... print("Gradient Descent: [-.245], [-.245]-&gt;format(theta[0], theta[1]) </pre>