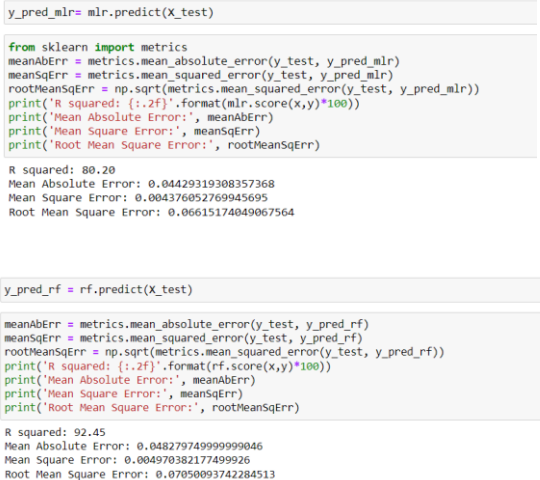
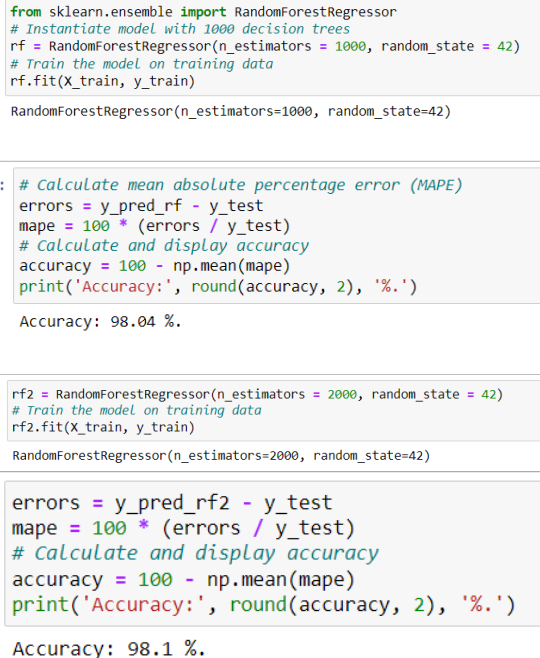


Project Development Phase Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID52898
Project Name	Project – University Admit Eligibility Predictor
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.	Metrics	<p>Regression Model:</p> <p>Linear Regression MAE - 80.20, MSE - 0.044293193083, RMSE - 0.0043760527699, R2 score - 0.066151740490</p> <p>Random Forest Regression: MAE - 92.45, MSE - 0.04827974999999, RMSE - 0.04827974999999, R2 score - 0.07050093742</p>	 <pre> y_pred_mlr = mlr.predict(X_test) from sklearn import metrics meanAbsErr = metrics.mean_absolute_error(y_test, y_pred_mlr) meanSqErr = metrics.mean_squared_error(y_test, y_pred_mlr) rootMeanSqErr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_mlr)) print('R squared: {:.2f}'.format(mlr.score(x,y)*100)) print('Mean Absolute Error:', meanAbsErr) print('Mean Square Error:', meanSqErr) print('Root Mean Square Error:', rootMeanSqErr) R squared: 80.20 Mean Absolute Error: 0.04429319308357368 Mean Square Error: 0.004376052769945695 Root Mean Square Error: 0.06615174049067564 y_pred_rf = rf.predict(X_test) meanAbsErr = metrics.mean_absolute_error(y_test, y_pred_rf) meanSqErr = metrics.mean_squared_error(y_test, y_pred_rf) rootMeanSqErr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf)) print('R squared: {:.2f}'.format(rf.score(x,y)*100)) print('Mean Absolute Error:', meanAbsErr) print('Mean Square Error:', meanSqErr) print('Root Mean Square Error:', rootMeanSqErr) R squared: 92.45 Mean Absolute Error: 0.048279749999999046 Mean Square Error: 0.004970382177499926 Root Mean Square Error: 0.07050093742284513 </pre>
2.	Tune the Model	<p>Hyperparameter Tuning – Applicable for Random forest Regression : Number of estimators</p> <p>Validation Method – We can see that relatively linear regression out performs random forest in metrics like MAE, MSE and RMSE and decision tree is slightly overfit for a linear problem like chance prediction</p>	 <pre> from sklearn.ensemble import RandomForestRegressor # Instantiate model with 1000 decision trees rf = RandomForestRegressor(n_estimators = 1000, random_state = 42) # Train the model on training data rf.fit(X_train, y_train) RandomForestRegressor(n_estimators=1000, random_state=42) : # Calculate mean absolute percentage error (MAPE) errors = y_pred_rf - y_test mape = 100 * (errors / y_test) # Calculate and display accuracy accuracy = 100 - np.mean(mape) print('Accuracy:', round(accuracy, 2), '%.') Accuracy: 98.04 %. : rf2 = RandomForestRegressor(n_estimators = 2000, random_state = 42) # Train the model on training data rf2.fit(X_train, y_train) : RandomForestRegressor(n_estimators=2000, random_state=42) errors = y_pred_rf2 - y_test mape = 100 * (errors / y_test) # Calculate and display accuracy accuracy = 100 - np.mean(mape) print('Accuracy:', round(accuracy, 2), '%.') Accuracy: 98.1 %. </pre>

