

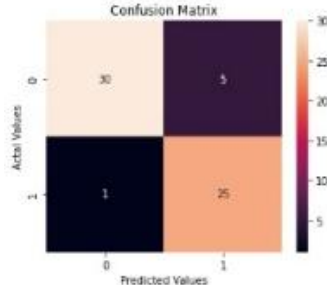
## Project Development Phase

### Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID05680
Project Name	Project - Car Resale value Prediction
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><b>Regression Model:</b> MAE -0.0983, MSE - 0.3136, RMSE - 0.3136, R2 score – 0.5978</p> <p><b>Classification Model:</b> Confusion Matrix, Accuray Score-0.9016 &amp; Classification Report</p>	<p>RMSE: 0.31362502409359</p> <p>MSE: 0.31362502409359</p> <p>MAE: 0.09836065573770492</p> <p>R2 SCORE: 0.5978021978021978</p> <p>Confusion Matrix</p>  <p>CLASSIFICATION REPORT</p> <pre>[60] from sklearn.metrics import classification_report print(classification_report(original_classes, pred_classes))</pre> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0.0</td><td>0.97</td><td>0.86</td><td>0.91</td><td>35</td></tr><tr><td>1.0</td><td>0.83</td><td>0.96</td><td>0.89</td><td>26</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.90</td><td>61</td></tr><tr><td>macro avg</td><td>0.90</td><td>0.91</td><td>0.90</td><td>61</td></tr><tr><td>weighted avg</td><td>0.91</td><td>0.90</td><td>0.90</td><td>61</td></tr></tbody></table>		precision	recall	f1-score	support	0.0	0.97	0.86	0.91	35	1.0	0.83	0.96	0.89	26	accuracy			0.90	61	macro avg	0.90	0.91	0.90	61	weighted avg	0.91	0.90	0.90	61
	precision	recall	f1-score	support																													
0.0	0.97	0.86	0.91	35																													
1.0	0.83	0.96	0.89	26																													
accuracy			0.90	61																													
macro avg	0.90	0.91	0.90	61																													
weighted avg	0.91	0.90	0.90	61																													
2.	Tune the Model	Hyperparameter Tuning , Validation Method	<p>Hyperparameter tuning for Random Forest using GridSearchCV and its data</p> <pre>[40] # define search space (grid of hyperparameters) # search across 100 different combinations of hyperparameters # make a 10-fold cross-validation, 5-fold internal cross-validation # 10-fold = 10, 5-fold = 5, 10-fold = 10, 5-fold = 5  [41] rf = RandomForestClassifier() [42] param_grid = {'n_estimators': [50, 100, 200, 400, 800, 1600],                  'max_depth': [None, 5, 10, 20, 40, 80, 160],                  'min_samples_split': [2, 5, 10, 20, 40, 80, 160],                  'min_samples_leaf': [1, 2, 5, 10, 20, 40, 80, 160],                  'bootstrap': [True, False],                  'oob_score': [True, False],                  'verbose': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]} [43] rf_cv = GridSearchCV(rf, param_grid, cv=5, scoring='r2') [44] rf_cv.fit(X_train, y_train) [45] rf_cv.best_estimator_ [46] rf_cv.best_score_ [47] rf_cv.best_params_ [48] rf_cv.score(X_test, y_test) [49] rf_cv.score(X_train, y_train) [50] rf_cv.score(X_val, y_val) [51] rf_cv.score(X_test, y_test) [52] rf_cv.score(X_train, y_train) [53] rf_cv.score(X_val, y_val) [54] rf_cv.score(X_test, y_test) [55] rf_cv.score(X_train, y_train) [56] rf_cv.score(X_val, y_val) [57] rf_cv.score(X_test, y_test) [58] rf_cv.score(X_train, y_train) [59] rf_cv.score(X_val, y_val) [60] rf_cv.score(X_test, y_test) [61] rf_cv.score(X_train, y_train) [62] rf_cv.score(X_val, y_val) [63] rf_cv.score(X_test, y_test) [64] rf_cv.score(X_train, y_train) [65] rf_cv.score(X_val, y_val) [66] rf_cv.score(X_test, y_test) [67] rf_cv.score(X_train, y_train) [68] rf_cv.score(X_val, y_val) [69] rf_cv.score(X_test, y_test) [70] rf_cv.score(X_train, y_train) [71] rf_cv.score(X_val, y_val) [72] rf_cv.score(X_test, y_test) [73] rf_cv.score(X_train, y_train) [74] rf_cv.score(X_val, y_val) [75] rf_cv.score(X_test, y_test) [76] rf_cv.score(X_train, y_train) [77] rf_cv.score(X_val, y_val) [78] rf_cv.score(X_test, y_test) [79] rf_cv.score(X_train, y_train) [80] rf_cv.score(X_val, y_val) [81] rf_cv.score(X_test, y_test) [82] rf_cv.score(X_train, y_train) [83] rf_cv.score(X_val, y_val) [84] rf_cv.score(X_test, y_test) [85] rf_cv.score(X_train, y_train) [86] rf_cv.score(X_val, y_val) [87] rf_cv.score(X_test, y_test) [88] rf_cv.score(X_train, y_train) [89] rf_cv.score(X_val, y_val) [90] rf_cv.score(X_test, y_test) [91] rf_cv.score(X_train, y_train) [92] rf_cv.score(X_val, y_val) [93] rf_cv.score(X_test, y_test) [94] rf_cv.score(X_train, y_train) [95] rf_cv.score(X_val, y_val) [96] rf_cv.score(X_test, y_test) [97] rf_cv.score(X_train, y_train) [98] rf_cv.score(X_val, y_val) [99] rf_cv.score(X_test, y_test)</pre>																														

