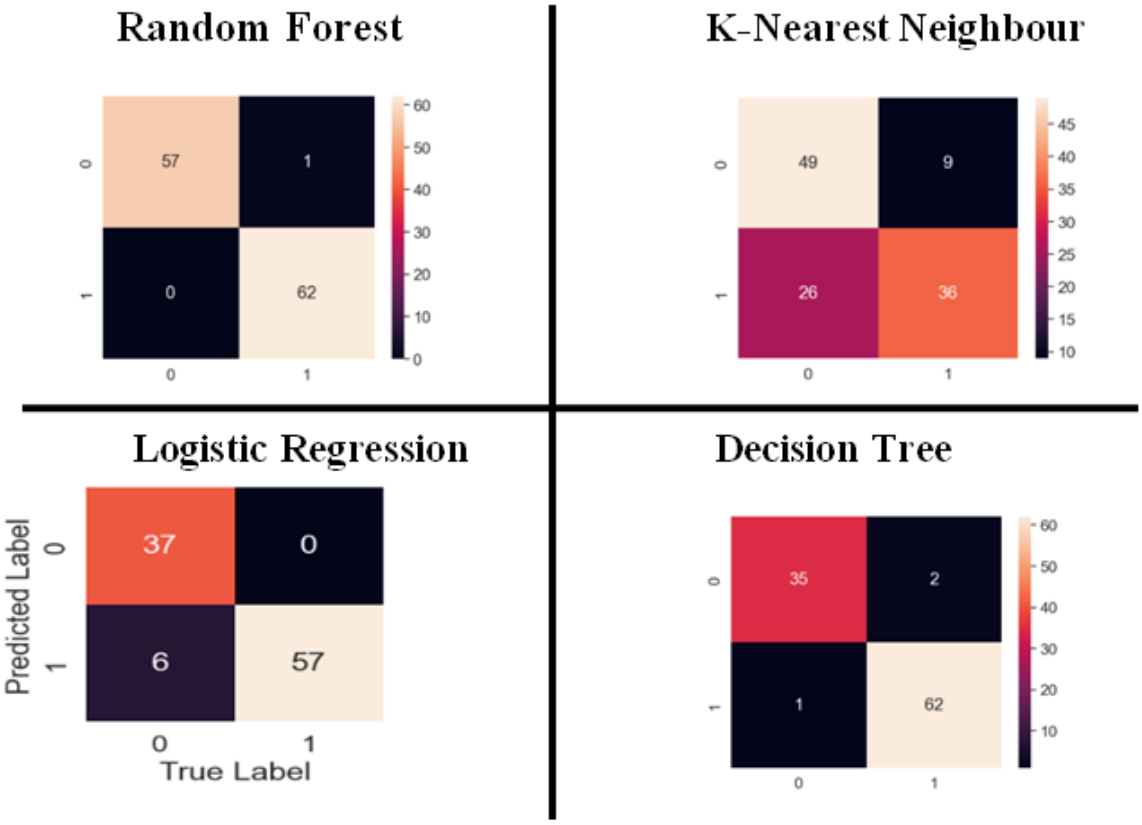
Project Development Phase
Model Performance Test

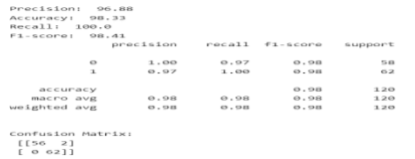
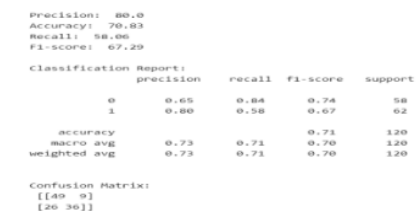
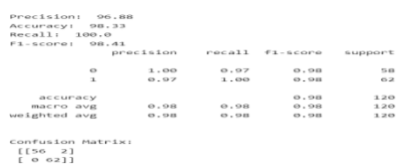
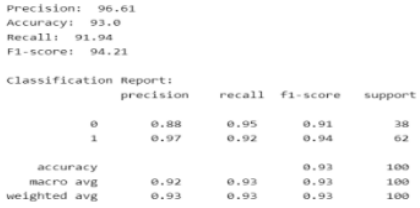
Date	10 November 2022
Team ID	PNT2022TMID22283
Project Name	Early Detection of Chronic Kidney disease using Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

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

Confusion matrix:



S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report -	**Random forest:**  Precision: 96.88 Accuracy: 98.33 Recall: 100.0 F1-score: 98.41 precision recall f1-score support 0 1.00 0.97 0.98 58 1 0.97 1.00 0.98 62 accuracy 0.98 120 macro avg 0.98 0.98 0.98 120 weighted avg 0.98 0.98 0.98 120 Confusion Matrix: [[56 2] [0 62]] **K-Nearest Neighbour:**  Precision: 80.0 Accuracy: 70.83 Recall: 58.06 F1-score: 67.29 Classification Report: precision recall f1-score support 0 0.65 0.84 0.74 58 1 0.80 0.58 0.67 62 accuracy 0.71 120 macro avg 0.73 0.71 0.70 120 weighted avg 0.73 0.71 0.70 120 Confusion Matrix: [[49 9] [26 36]] **Logistic Regression:**  Precision: 96.88 Accuracy: 98.33 Recall: 100.0 F1-score: 98.41 precision recall f1-score support 0 1.00 0.97 0.98 58 1 0.97 1.00 0.98 62 accuracy 0.98 120 macro avg 0.98 0.98 0.98 120 weighted avg 0.98 0.98 0.98 120 Confusion Matrix: [[56 2] [0 62]] **Decision Tree**  Precision: 96.61 Accuracy: 93.0 Recall: 91.94 F1-score: 94.21 Classification Report: precision recall f1-score support 0 0.88 0.95 0.91 38 1 0.97 0.92 0.94 62 accuracy 0.93 100 macro avg 0.92 0.93 0.93 100 weighted avg 0.93 0.93 0.93 100 Confusion Matrix: [[36 2] [5 57]]

2.	Tune the Model	Hyperparameter Tuning - Validation Method -	In [0]:
			`from sklearn.model_selection import GridSearchCV`
			`import numpy as np`
			
			`max_features_range = np.arange(1,6,1)`
			`n_estimators_range = np.arange(10,210,10)`
			`param_grid = dict(max_features=max_features_range, n_estimators=n_estimators_range)`
			
			`rf = RandomForestClassifier()`
			
			`grid = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5)`
			
			In [13]:
			`grid.fit(X_train, Y_train)`
			
			Out[13]:
			`GridSearchCV(cv=5, error_score=nan,`
			`estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,`
			`class_weight=None,`
			`criterion='gini', max_depth=None,`
			`max_features='auto',`
			`max_leaf_nodes=None,`
			`max_samples=None,`
			`min_impurity_decrease=0.0,`
			`min_impurity_split=None,`
			`min_samples_leaf=1,`
			`min_samples_split=2,`
			`min_weight_fraction_leaf=0.0,`
			`n_estimators=100, n_jobs=None,`
			`oob_score=False,`
			`random_state=None, verbose=0,`
			`warm_start=False),`
			`iid='deprecated', n_jobs=None,`
			`param_grid={'max_features': array([1, 2, 3, 4, 5]),`
			`'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,`
			`140, 150, 160, 170, 180, 190, 200])},`
			`pre_dispatch='2*n_jobs', refit=True, return_train_score=False,`
			`scoring=None, verbose=0)`
			
			In [14]:
			`print("The best parameters are %s with a score of %0.2f"`
			`% (grid.best_params_, grid.best_score_))`
			
			`The best parameters are {'max_features': 1, 'n_estimators': 120} with a score of 0.98`