Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID22069
Project Name	Project – 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.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuracy Score- & Classification Report -	See Below
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	See Below

1. Metrics

Model: Random Forest Classifier

Random Forest Classifier

In [115]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train , y_train)
prediction = model.predict(x_test)
from sklearn.metrics import confusion_matrix
print('RandomForest\n')
print('confusion_matrix')
print(confusion_matrix(prediction,y_test))
print('\n')
print('accuracy_score')
print(accuracy_score(prediction,y_test))
print('\n')
```

```
RandomForest

confusion_matrix
[[51  1]
 [ 3 25]]


accuracy_score
0.95
```

2. Tune the Model

Hyperparameter Tuning:

- The number of features is important and should be tuned in random forest classification.
- Initially all parameters in the dataset are taken as independent values to arrive at the dependent decision of Chronic Kidney Disease or No Chronic Kidney Disease.
- But the result was not accurate so used only 8 more correlated values as independent values to arrive at the dependent decision of Chronic Kidney Disease or not.

Validation Method:

It involves **partitioning the training data set into subsets, where one subset is held out to test the performance of the model**. This data set is called the validation data set.

Cross validation is to use different models and identify the best:

Logistic Regression Model performance values:

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(solver ='lbfgs',max_iter=500)
print('LogisticRegression\n')
model.fit(x_train.values,y_train.values.ravel())
prediction = model.predict(x_test)
from sklearn.metrics import confusion_matrix
print('confusion_matrix')
print(confusion_matrix(prediction,y_test))
print('\n')
print('accuracy_score')
print(accuracy_score(prediction,y_test))
print('\n')
```

```
LogisticRegression

confusion_matrix
[[48  0]
 [ 6 26]]


accuracy_score
0.925
```

Hence we tested with Logistic regression and Random Forest Classification wherein the accuracy of Random Forest classification is 95% compared with Logistic Regression.

Metric	Logistic Regression	Random Forest Classification
Accuracy	0.925	0.95

Other metrics	Logistic Regression
	`from sklearn.linear_model import LogisticRegression`
	`model=LogisticRegression(solver ='lbfgs',max_iter=500)`
	`print('LogisticRegression\n')`
	`model.fit(x_train.values,y_train.values.ravel())`
	`prediction = model.predict(x_test)`
	`from sklearn.metrics import confusion_matrix`
	`print('confusion_matrix')`
	`print(confusion_matrix(prediction,y_test))`
	`print('\n')`
	`print('accuracy_score')`
	`print(accuracy_score(prediction,y_test))`
	`print('\n')`
	LogisticRegression
	confusion_matrix
	[[48 0]
	[6 26]]
	accuracy_score
	0.925
		Random Forest Classifier
		In [115]:
		`from sklearn.ensemble import RandomForestClassifier`
		`model = RandomForestClassifier()`
		`model.fit(x_train , y_train)`
		`prediction = model.predict(x_test)`
		`from sklearn.metrics import confusion_matrix`
		`print('RandomForest\n')`
		`print('confusion_matrix')`
		`print(confusion_matrix(prediction,y_test))`
		`print('\n')`
		`print('accuracy_score')`
		`print(accuracy_score(prediction,y_test))`
		`print('\n')`
		RandomForest
		confusion_matrix
		[[51 1]
		[3 25]]
		accuracy_score
		0.95

The above table shows that Random Forest Classification gives better results over Logistic Regression.