Project Development Phase
Model Performance Test

Date	18 November 2022
Team ID	PNT2022TMID28091
Project Name	Efficient Water Quality Analysis & Prediction using Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

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

Model Summary:

jupyter water potability Last Checkpoint: 15 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [79]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from xgboost import XGBClassifier


from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
```

In [80]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
```

In [81]:
```
lr = LogisticRegression(random_state=42)

knn = KNeighborsClassifier()

dt = DecisionTreeClassifier()

rf = RandomForestClassifier()

ada = AdaBoostClassifier()

xgb =XGBClassifier(eval_metric = 'logloss', use_label_encoder=False)


para_knn = {'n_neighbors':np.arange(1, 50)}  #parameters of knn
grid_knn = GridSearchCV(knn, param_grid=para_knn, cv=5) #search knn for 5 fold cross validation

#parameters for decision tree
para_dt = {'criterion':['gini','entropy'],'max_depth':np.arange(1, 50), 'min_samples_leaf':[1,2,4,5,10,20,30,40,80,100]}
grid_dt = GridSearchCV(dt, param_grid=para_dt, cv=5) #grid search decision tree for 5 fold cv
#"gini" for the Gini impurity and "entropy" for the information gain.
#min_samples_leaf: The minimum number of samples required to be at a leaf node, have the effect of smoothing the model

#parameters for random forest
#n_estimators: The number of trees in the forest.
params_rf = {'n_estimators':[100,200, 350, 500], 'min_samples_leaf':[2, 10, 30]}
grid_rf = GridSearchCV(rf, param_grid=params_rf, cv=5)

#parameters fpr AdaBoost
params_ada = {'n_estimators': [50,100,250,400,500,600], 'learning_rate': [0.2,0.5,0.8,1]}
grid_ada =  GridSearchCV(ada, param_grid=params_ada, cv=5)
```

Accuracy

Training Accuracy and Validation Accuracy: -

In [86]:
```
from sklearn.metrics import classification_report

y_pred_rf= xgb.predict(X_test)
print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.93      0.90      0.91       518
           1       0.66      0.75      0.70       138

    accuracy                           0.87       656
   macro avg       0.80      0.82      0.81       656
weighted avg       0.87      0.87      0.87       656
```

In [87]:
```
from sklearn.metrics import classification_report, precision_score, recall_score, confusion_matrix
print(precision_score(y_test, y_pred_rf))
print(recall_score(y_test, y_pred_rf))
```

```
0.6645161290322581
0.7463768115942029
```

In [88]:
```
print(confusion_matrix(y_test, y_pred_rf))
```

```
[[466  52]
 [ 35 103]]
```

In [89]:
```
import pickle
filename = 'xgboost.sav'
pickle.dump(xgb, open(filename, 'wb'))

# some time later...

# load the model from disk
```