Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID21489
Project Name	Essential Water Quality Analysis and Prediction 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	**Classification Model:**	

```
[31] #Splitting the data into dependent and independent variables
     X= df[['year', 'DO', 'PH', 'CO','BOD','NI','Tot_col']]
     df['wqi']=df['wqi'].astype('int')
     Y= df[['wqi']]
```

```
[32] X.shape

     (1900, 7)
```

```
[33] Y.shape

     (1900, 1)
```

```
[34] from sklearn.model_selection import train_test_split
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.neural_network import MLPClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn import linear_model
     from sklearn import metrics
     import math
     from sklearn.metrics import mean_squared_error

     X_train, X_test, Y_train, Y_test = train_test_split( X, Y,test_size=0.2, random_state=10)

     #from sklearn.preprocessing import StandardScaler
     #sc_X = StandardScaler()
     #X_train = sc_X.fit_transform(X_train)
     #X_test = sc_X.transform(X_test)

     #{Decision Tree Model}
     clf = DecisionTreeClassifier()
     clf = clf.fit(X_train,Y_train)
     clf_pred=clf.predict(X_test)
     clf_accuracy=metrics.accuracy_score(Y_test,clf_pred)
     print("1) Using Decision Tree Prediction, Accuracy is " + str(clf_accuracy))
```

```
#(K Neighbors Classifier)
knn = KNeighborsClassifier(n_neighbors=7)
knn=knn.fit(X_train,Y_train.values.ravel())
knn_pred=knn.predict(X_test)
knn_accuracy=metrics.accuracy_score(Y_test,knn_pred)
print ("2) Using K Neighbors Classifier Prediction, Accuracy is " + str(knn_accuracy))

#(using MLPClassifier)
mlpc = MLPClassifier()
mlpc.fit(X_train,Y_train.values.ravel())
mlpc_pred=mlpc.predict(X_test)
mlpc_accuracy=metrics.accuracy_score(Y_test,mlpc_pred)
print ("3) Using MLPC Classifier Prediction, Accuracy is " + str(mlpc_accuracy))

#(using MLPClassifier)
rfor = RandomForestClassifier()
rfor.fit(X_train,Y_train.values.ravel())
rfor_pred=rfor.predict(X_test)
rfor_accuracy=metrics.accuracy_score(Y_test,rfor_pred)
print ("4) Using RandomForest Classifier Prediction, Accuracy is " + str(rfor_accuracy))

#(using Linear Regression)
linreg=linear_model.LinearRegression()
linreg.fit(X_train,Y_train)
linreg_pred=rfor.predict(X_test)
linreg_accuracy=metrics.accuracy_score(Y_test,linreg_pred)
rmse = math.sqrt(mean_squared_error(Y_test,linreg_pred))
print ("5) Using Linear Regression Prediction, Accuracy is " + str(linreg_accuracy))
```

```
1) Using Decision Tree Prediction, Accuracy is 0.8131578947368421
2) Using K Neighbors Classifier Prediction, Accuracy is 0.3157894736842105
3) Using MLPC Classifier Prediction, Accuracy is 0.14473684210526316
4) Using RandomForest Classifier Prediction, Accuracy is 0.8184210526315789
5) Using Linear Regression Prediction, Accuracy is 0.8184210526315789
```

Confusion Matrix

```
[35] metrics.confusion_matrix(Y_test, rfor_pred)

array([[ 0,  0,  1, ...,  0,  0,  0],
       [ 0,  0,  0, ...,  0,  0,  0],
       [ 0,  0,  3, ...,  0,  0,  0],
       ...,
       [ 0,  0,  0, ..., 12,  0,  0],
       [ 0,  0,  0, ...,  0, 11,  0],
       [ 0,  0,  0, ...,  1,  1,  1]])
```

Accuracy Score

Accuracy of algorithms

- Decision Tree - 81.57%
- KNN - 31.57%
- MLPC classifier - 12.36%
- Random Forest - 82.10%
- Linear Regression - 82.10%

		Classification Report	

```
[36] print(metrics.classification_report(Y_test, rfor_pred))
```

	precision	recall	f1-score	support
50	0.75	1.00	0.86	6
51	0.00	0.00	0.00	2
52	0.00	0.00	0.00	1
53	0.00	0.00	0.00	1
54	0.00	0.00	0.00	1
55	0.86	0.92	0.89	13
56	0.00	0.00	0.00	2
58	0.00	0.00	0.00	1
59	0.00	0.00	0.00	1
60	1.00	0.67	0.80	6
61	0.76	0.81	0.79	16
62	0.00	0.00	0.00	1
64	0.00	0.00	0.00	1
65	1.00	0.67	0.80	3
66	0.73	0.62	0.67	13
67	0.85	0.85	0.85	13
68	0.00	0.00	0.00	2
69	1.00	1.00	1.00	1
70	0.86	1.00	0.92	6
71	1.00	0.62	0.77	8
72	0.79	0.88	0.83	17
73	0.73	0.73	0.73	11
74	0.00	0.00	0.00	1
75	0.00	0.00	0.00	2
76	0.71	0.71	0.71	17
77	0.71	0.62	0.67	8
78	0.79	0.73	0.76	15
79	0.76	0.89	0.82	18

```
[36]
```

76	0.71	0.71	0.71	17
77	0.71	0.62	0.67	8
78	0.79	0.73	0.76	15
79	0.76	0.89	0.82	18
81	0.40	1.00	0.57	2
82	0.93	0.96	0.94	69
83	0.81	0.94	0.87	18
84	0.83	0.83	0.83	6
85	0.83	0.83	0.83	6
87	0.57	1.00	0.73	4
88	0.95	0.98	0.96	42
89	0.75	1.00	0.86	6
90	1.00	1.00	1.00	2
93	1.00	1.00	1.00	12
94	0.92	1.00	0.96	11
99	1.00	0.67	0.80	3
accuracy			0.83	380
macro avg	0.50	0.52	0.50	380
weighted avg	0.80	0.83	0.81	380

2.	Tune the Model	Hyperparameter Tuning - Validation Method -	Hyperparameter tuning and cross validation 

```
[40] # automatic nested cross-validation for random forest on a classification dataset
     from numpy import mean
     from numpy import std
     from sklearn.datasets import make_classification
     from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import KFold
     from sklearn.model_selection import GridSearchCV
     from sklearn.ensemble import RandomForestClassifier
     # create dataset

     # configure the cross-validation procedure
     cv_inner = KFold(n_splits=3, shuffle=True, random_state=1)
     # define the model
     model = RandomForestClassifier(random_state=1)
     # define search space
     space = dict()
     space['n_estimators'] = [10, 100, 500]
     space['max_features'] = [2, 4, 6]
     # define search
     search = GridSearchCV(model, space, scoring='accuracy', n_jobs=1, cv=cv_inner, refit=True)
     # configure the cross-validation procedure
     cv_outer = KFold(n_splits=10, shuffle=True, random_state=1)
     # execute the nested cross-validation
     scores = cross_val_score(search, X, Y, scoring='accuracy', cv=cv_outer, n_jobs=-1)
     # report performance
     print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))

     Accuracy: 0.869 (0.021)
```