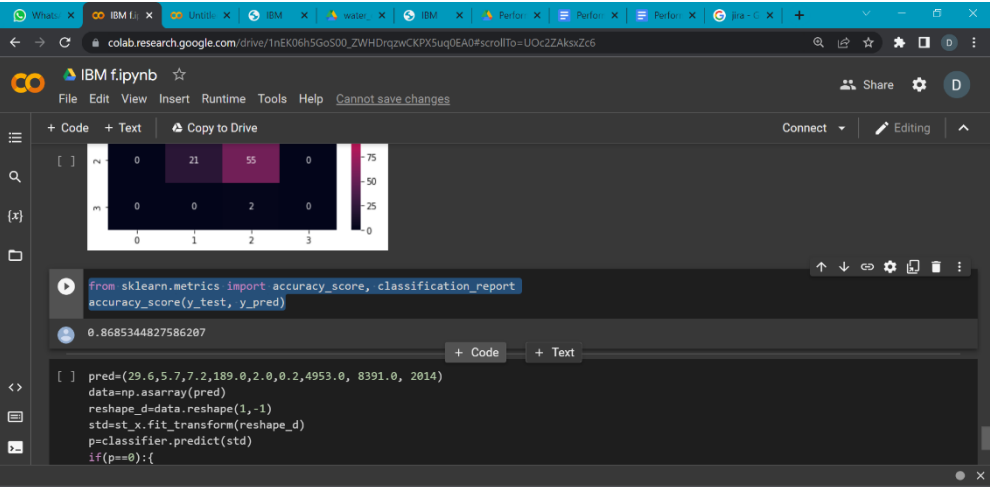
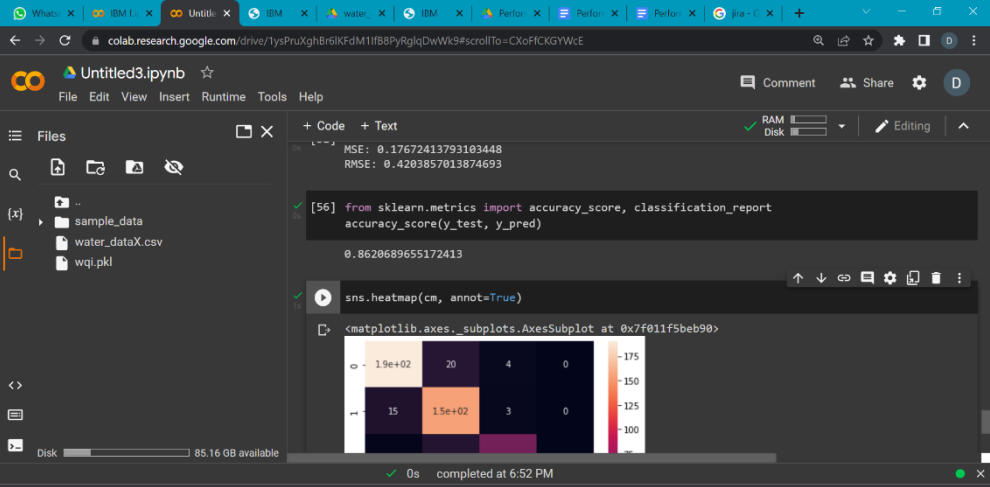


Project Development Phase Model Performance Test

Date	16 November 2022
Team ID	PNT2022TMID14644
Project Name	Efficient Water Quality Analysis and Prediction Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Metrics	<p>Regression Model: MAE - , MSE - , RMSE - , R2 score -</p> <p>Classification Model: Confusion Matrix - , Accuracy Score- & Classification Report -</p>	<p>Regression model:</p>  <p>Classification Model:</p> 

Classification Model :

The screenshot shows a Google Colab notebook interface. The top bar includes tabs for 'What's up', 'IBM f.ipynb', 'water_data.txt', 'Performance', and 'IBM'. The address bar shows the URL: colab.research.google.com/drive/1nEK06h5Go500_ZWHDqzqCKPXSuq0EA0#scrollTo=mg4uNMbcxOIG. The notebook title is 'IBM f.ipynb'. The code editor contains the following Python code:

```
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion='entropy')
classifier.fit(x_train, y_train)

RandomForestClassifier(criterion='entropy', n_estimators=10)

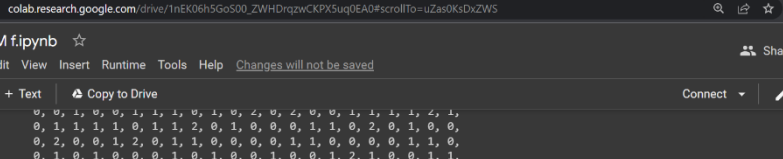
[ ] from sklearn.metrics import accuracy_score

[ ] train_acc=classifier.predict(x_train)
accuracy_score(train_acc,y_train)

0.9949712643678161

[ ] #Predicting the test set result
y_pred= classifier.predict(x_test)
```

The bottom of the interface shows a file explorer with 'Performance_Test_1.docx' and 'water_data1.txt'.



The screenshot shows a Google Colab notebook titled "water_data1.txt". The code in the notebook is as follows:

```
from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_test, y_pred)

0.8685344827586207

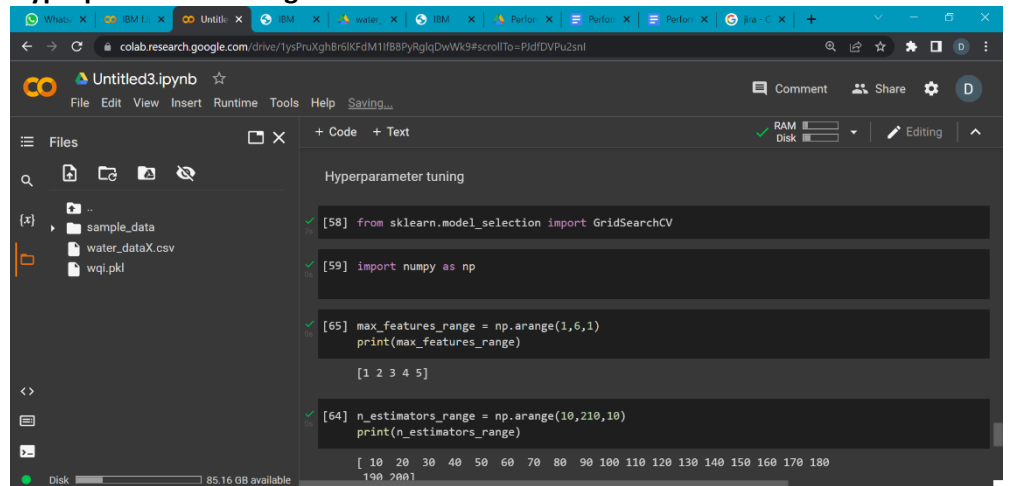
[ ] pred=(29.6,5.7,7.2,189.0,2.0,0.2,4953.0, 8391.0, 2014)
data=np.asarray(pred)
reshape_d=data.reshape(1,-1)
std=st_x.fit_transform(reshape_d)
p=classifier.predict(std)
if(p==0):{
    print("The water quality is Very Bad")
}
if(p==1):{
    print("The water quality is Poor ")
}
if(p==2):{
    print("The water quality is Good")
}
```

The notebook interface includes a file explorer on the left, a code editor in the center, and a file manager at the bottom. The file manager shows two files: "Performance Test....docx" and "water_data1.txt".

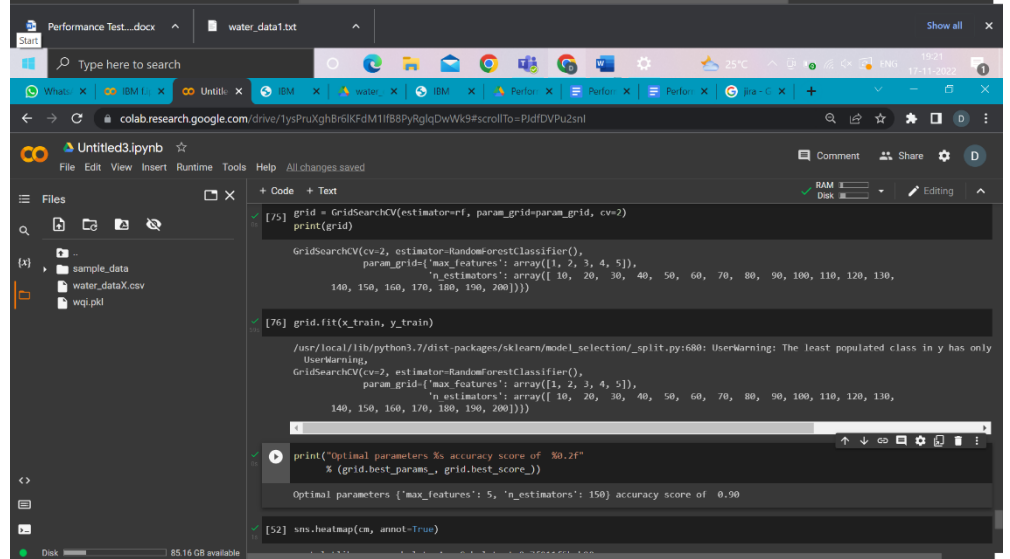
2. Tune the Model

Hyperparameter Tuning - Validation Method -

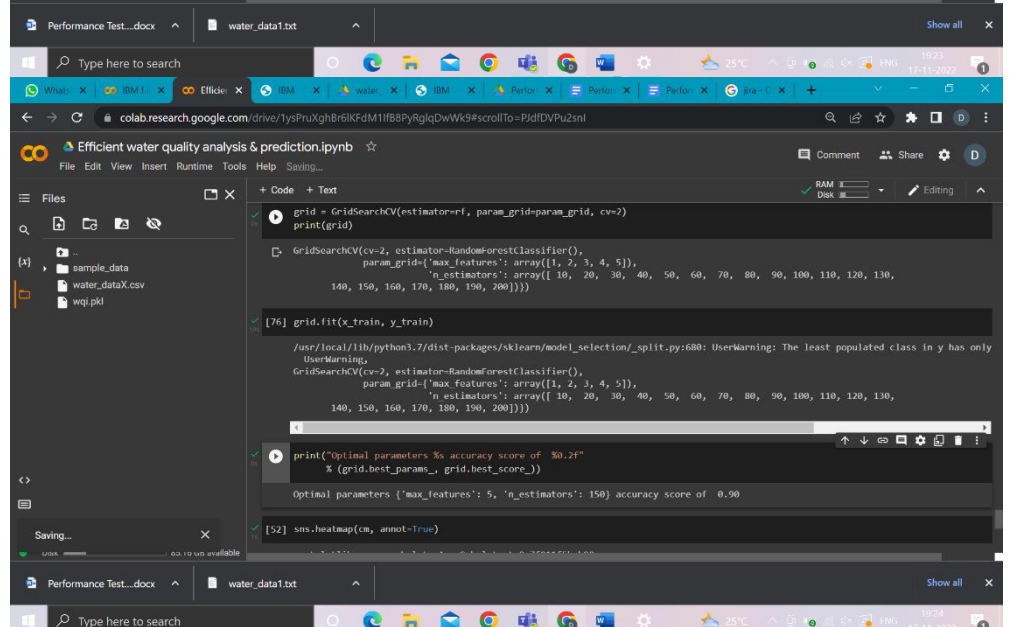
Hyperparameter tuning :



```
[58] from sklearn.model_selection import GridSearchCV
[59] import numpy as np
[65] max_features_range = np.arange(1,6,1)
print(max_features_range)
[1 2 3 4 5]
[64] n_estimators_range = np.arange(10,210,10)
print(n_estimators_range)
[ 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180
 190 200]
```

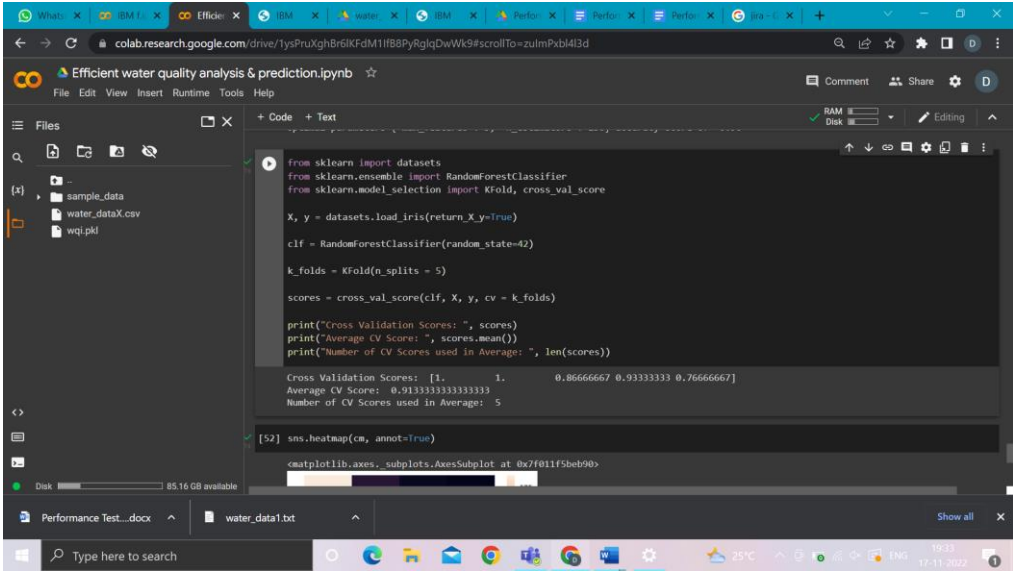


```
[75] grid = GridSearchCV(estimator=rf, param_grid=param_grid, cv=2)
print(grid)
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
              param_grid={'max_features': array([1, 2, 3, 4, 5]),
                          'n_estimators': array([ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130,
140, 150, 160, 170, 180, 190, 200])})
[76] grid.fit(x_train, y_train)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only
UserWarning:
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
              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])})
[77] print("Optimal parameters %s accuracy score of %0.2f"
          % (grid.best_params_, grid.best_score_))
Optimal parameters {'max_features': 5, 'n_estimators': 150} accuracy score of 0.90
[52] sns.heatmap(cm, annot=True)
```



```
[75] grid = GridSearchCV(estimator=rf, param_grid=param_grid, cv=2)
print(grid)
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
              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])})
[76] grid.fit(x_train, y_train)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only
UserWarning:
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
              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])})
[77] print("Optimal parameters %s accuracy score of %0.2f"
          % (grid.best_params_, grid.best_score_))
Optimal parameters {'max_features': 5, 'n_estimators': 150} accuracy score of 0.90
[52] sns.heatmap(cm, annot=True)
```

Validation method :



```
from sklearn import datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold, cross_val_score

X, y = datasets.load_iris(return_X_y=True)

clf = RandomForestClassifier(random_state=42)

k_folds = KFold(n_splits = 5)

scores = cross_val_score(clf, X, y, cv = k_folds)

print("Cross Validation Scores: ", scores)
print("Average CV Score: ", scores.mean())
print("Number of CV Scores used in Average: ", len(scores))

Cross Validation Scores: [1. 1. 0.86666667 0.93333333 0.76666667]
Average CV Score: 0.9133333333333333
Number of CV Scores used in Average: 5

[52] sns.heatmap(ca, annot=True)

<matplotlib.axes._subplots.AxesSubplot at 0x7f01f5beb90>
```

