PROJECT DEVELOPMENT PHASE - SPRINT III

Assignment Date	09-11-2022
Team ID	PNT2022TMID06599
Project Name	Efficient Water Quality Analysis and Prediction using Machine Learning
Maximum Marks	8 Mark

Train and Develop the Model

Data Collection:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.patches as patches
from matplotlib.patches import ConnectionPatch
from collections import OrderedDict
from matplotlib.gridspec import GridSpec
%matplotlib inline

df = pd.read_csv('Final.csv')
df
```

Exploratory Data Analysis:

```
df.shape

df.isnull().sum()

df.info()

df.describe()

df.fillna(df.mean(), inplace=True)
df.isnull().sum()
```

```
df.Potability.value_counts()

sns.countplot(df['Potability'])
plt.show()

sns.distplot(df['ph'])
plt.show()

df.hist(figsize=(14,14))
plt.show()


plt.figure(figsize=(13,8))
sns.heatmap(df.corr(),annot=True,cmap='terrain')
plt.show()

df.boxplot(figsize=(14,7))

X = df.drop('Potability',axis=1)
Y= df['Potability']

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size= 0.2,
random_state=101,shuffle=True)
```

Train Decision Tree Classifier and check accuracy:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
dt=DecisionTreeClassifier(criterion= 'gini', min_samples_split= 10, splitter= 'best')
dt.fit(X_train,Y_train)


prediction=dt.predict(X_test)
print(f"Accuracy Score = {accuracy_score(Y_test,prediction)*100}")
print(f"Confusion Matrix =\n {confusion_matrix(Y_test,prediction)}")
print(f"Classification Report =\n {classification_report(Y_test,prediction)}")
```

```
df.head

res =
dt.predict([[7.408985467,0.57139761,40,6.505923139,311.4526625,504.1459941,
11.53214401,81.10693773,3.772420928,0.0,100,0.0,16.5,0.0,11.24]])[0]
res
```

Apply Hyper Parameter Tuning:

```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV

# define models and parameters
model = DecisionTreeClassifier()
criterion = ["gini", "entropy"]
splitter = ["best", "random"]
min_samples_split = [2,4,6,8,10,12,14]

# define grid search
grid = dict(splitter=splitter, criterion=criterion,
min_samples_split=min_samples_split)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search_dt = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1,
cv=cv,
                          scoring='accuracy',error_score=0)
grid_search_dt.fit(X_train, Y_train)
```

```
print(f"Best: {grid_search_dt.best_score_:.3f} using
{grid_search_dt.best_params_}")
means = grid_search_dt.cv_results_['mean_test_score']
stds = grid_search_dt.cv_results_['std_test_score']
params = grid_search_dt.cv_results_['params']
```

```
for mean, stdev, param in zip(means, stds, params):
    print(f"{mean:.3f} ({stdev:.3f}) with: {param}")

print("Training Score:",grid_search_dt.score(X_train, Y_train)*100)
print("Testing Score:", grid_search_dt.score(X_test, Y_test)*100)
```

Modelling:

```
df.head(20)

df.tail(5)

df['Potability'].value_counts().to_frame()

df_filtered = df[df['Turbidity'].isin(["1,2,3,4,5,6,7,8,9,10"])]

print(df_filtered.head(15))

print(df_filtered.shape)
```

Model Evaluation

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
print('R Squared=',r2_score(X_train,Y_test))
print('MAE=',mean_absolute_error(X_train,Y_test))
print('MSE=',mean_squared_error(X_train,Y_tes

import joblib
joblib.dump(dt, 'classifier.pkl')


!pip install -U ibm-watson-machine-learning
```

```
from ibm_watson_machine_learning import APIClient
import json
import numpy as np


wml_credentials =
{"apikey":"nFFWACn7pVNTQWlnb7pusoXVa63g0vFEq_8Y2x2pxZSE","url":
            "https://us-south.ml.cloud.ibm.com" }

wml_client = APIClient(wml_credentials)
wml_client.spaces.list()

SPACE_ID = "3255cdbd-d2f9-4a9d-b816-efff2d706372"

wml_client.set.default_space(SPACE_ID)

wml_client.software_specifications.list(500)

import sklearn
sklearn.__version__

MODEL_NAME = 'wqi'
DEPLOYMENT_NAME = 'Model'
DEMO_MODEL = dt


# Set Python Version
software_spec_uid =
wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')


# Setup model meta
model_props = {
   wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
   wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
   wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
}
```

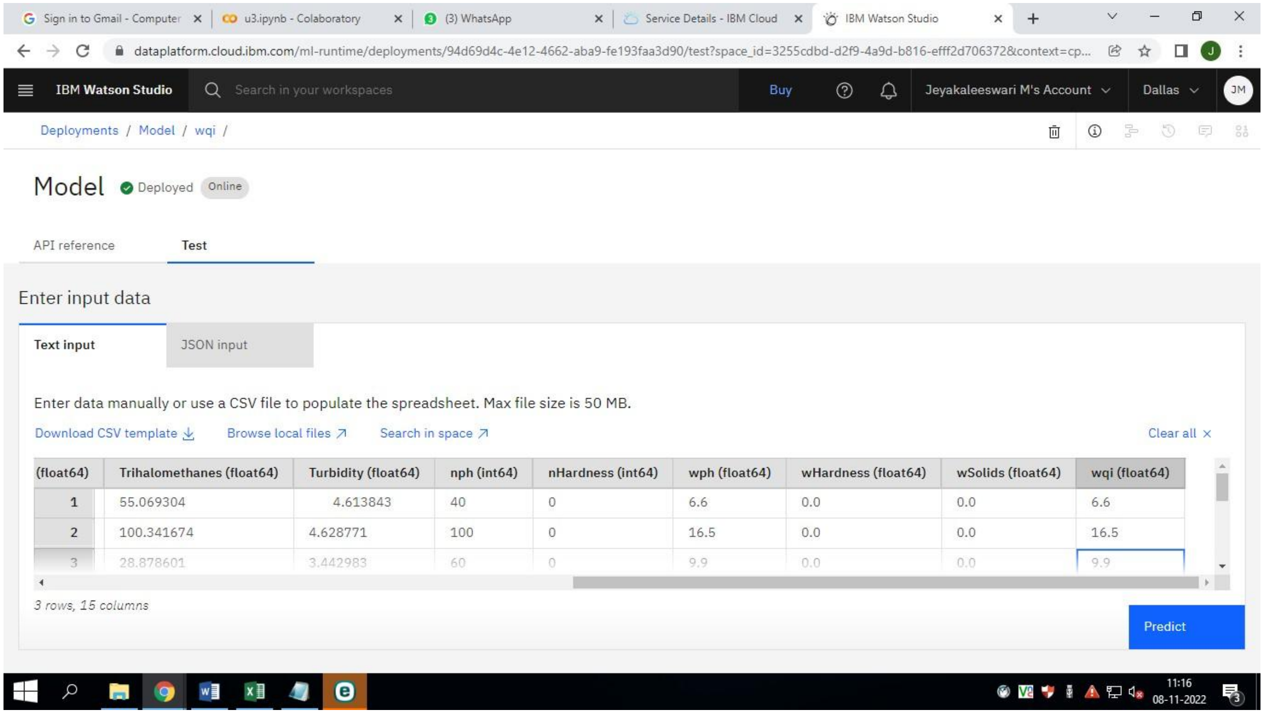
SAVE THE MODEL:

```
#Save model
model_details =
   wml_client.repository.store_model(model=DEM
   O_MODEL, meta_props=model_props,
   training_data=X_train,
   training_target=Y_train
)


model_details


model_id = wml_client.repository.get_model_id(model_details)

model_id

# Set meta
deployment_props = {

wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NA
ME,
   wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

DEPLOY:

```
# Deploy
deployment =
   wml_client.deployments.create(artifact_uid
   =model_id, meta_props=deployment_props
)
```

INPUT:



OUTPUT: