PROJECT DEVELOPMENT PHASE

SPRINT – 3

SOURCE CODE

DATE	12-NOV-2022
TEAM ID	PNT2022TMID27775
PROJECT NAME	DEVELOPING A FLIGHT DELAY MODEL USING MACHINE LEARNING
MAXIMUM MARKS	8 MARKS

IMPORT LIBRARIES

```
import numpy as np
import pandas as pd
```

IMPORT LABEL ENCODER

```
from sklearn.preprocessing import
LabelEncoder from sklearn.ensemble import
RandomForestClassifierfrom sklearn.metrics
import classification_report from
sklearn.metrics import jaccard_score

from  sklearn.model_selection import
train_test_split
```

IMPORT DATASET

```
import os, types
import  pandas
as pd
from botocore.client import
Configimport ibm_boto3
```

```
def_iter_(self): return 0

@hidden_cell

The following code accesses a file in your IBM Cloud Object Storage


cos_client = ibm_boto3.client(service_name='s3',
   ibm_api_key_id='BmleA4MV5fW02WAmF6zCBnBmBBkh7otufBwtC7V
   84yVO',
   ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
   config=Config(signature_version='oauth'),
   endpoint_url='https://s3.private.us.cloud-object-
   storage.appdomain.cloud')

bucket = 'randommodel-donotdelete-pr-
jpkful51t7p3nj'object_key =
'Processed_data15.csv'

body   =   cos_client.get_object(Bucket=bucket,Key=object_key)['Body']


Add missing_iter__method, so pandas accepts body as file-like object

if not hasattr(body, "__iter__"): body.__iter= types.MethodType(iter , body )

df =
pd.read_csv(body)
df.head()

df.head(90)

columns=  ['carrier','dest',
'origin']le=LabelEncoder()
for i in columns:
    df[i]=le.fit_transform(df[i])

df['carrier'].unique()
```

```
df['origin'].unique()

df['dest'].uniq

ue()

df.head(90)
```

FROM COLUMN(YEARS) TO COLUMN(DISTANCE)

```
X = df.iloc[:,
0:6].valuesX[0:5]

y = df['delayed']
y.head().to
frame()
for i in range(0, 20):
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
   random_state=i)
```

CREATING RANDOM FOREST CLASSIFIER

```
clf =
   RandomForestClassifier(random_stat
   e=i)clf.fit(X_train, y_train)
```

DETERMINING THE SCORE

```
train_score = clf.score(X_train,
   y_train)test_score =
   clf.score(X_test, y_test)
   print("Test: {}, Train: {} and Random State: {}".format(test_score, train_score, i))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=18)clf = RandomForestClassifier(random_state=18)
clf.fit(X_train, y_train)

print("Train set: ", clf.score(X_train,

y_train))print("Test set: ",
```

```
clf.score(X_test, y_test))
```

PREDICTING THE TRAINED CLASSIFIER TO THE TEST

```
yhat = clf.predict(X_test)
```

VIEWING THE PREDICTED PROBABILITIES OF FIRST 10 OBSERVATIONS

```
yhat_prob =

clf.predict_proba(X_test)[:10]

print(classification_report(y_test,

yhat))

import joblib
joblib.dump(clf, '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":"MAmvQGzuqmoDN0P9M8ziexwNLRu_aJTZrHq4pWlkY67k",
            "url": "https://us-south.ml.cloud.ibm.com" }
wml_client =
APIClient(wml_credentials)
wml_client.spaces.list()

SPACE_ID  =  "7c5663ee-671c-49d2-a415-a27bac157d6d"

wml_client.set.default_space(SPACE

_ID)

wml_client.software_specifications.li

st(500)
```

SAVE AND DEPLOY THE MODEL

```
import sklearn
sklearn.__version__

MODEL_NAME = 'Flight'
DEPLOYMENT_NAME =
'model_deploy'DEMO_MODEL = clf
```

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 MODEL

```
model_details =
   wml_client.repository.store_model(
   model=DEMO_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

```
deployment =
   wml_client.deployments.create(
   artifact_uid=model_id,
   meta_props=deployment props )
```