PROJECT DEVELOPMENT PHASE

SPRINT – 3

SOURCE CODE

|  |  |
| --- | --- |
| Date | 09-Nov-2022 |
| Team ID | PNT2022TMID17753 |
| 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 RandomForestClassifier from 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 Config import 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**=**'BmleA4MV5fW02WAmF6zCBnBmBBkh7otufBwtC7V84yVO', 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']**.**unique() df**.**head(90)

# FROM COLUMN(YEARS) TO COLUMN(DISTANCE)

X **=** df**.**iloc[:, 0:6]**.**values X[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\_state**=**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.list(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\_NAME, wml\_client**.**deployments**.**ConfigurationMetaNames**.**ONLINE: {}

}

# DEPLOY

deployment **=** wml\_client**.**deployments**.**create( artifact\_uid**=**model\_id, meta\_props**=**deployment props )