PROJECT DEVELOPMENT PHASE

Sprint - III

Date	09-Nov-2022
Team ID	PNT2022TMID49460
Project Name	Developing a Flight Delay Model Using Machine Learning
Maximum Marks	4 Marks

Training the model on IBM

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

You might want to remove those credentials before you share the notebook

```
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":"gyOvc0l0Hde4zdTmNc47N4Vh1zmMTFh7FlK8BEcKPADB",
              "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
)
```