

SPRINT- 4 : DEPLOYMENT

IBM Cloud Pak for Data

Search in your workspaces

Buy

saranya's Account

Dallas

SS

Projects / Detecting Parkinsons Disease us... / Parkinson_Disease_Prediction

File Edit View Insert Cell Kernel Help

Trusted Python 3.9

In [1]: !pip install imutils
Collecting imutils
 Downloading imutils-0.5.4.tar.gz (17 kB)
 Building wheels for collected packages: imutils
 Building wheel for imutils (setup.py) ... done
 Stored in directory: /tmp/.cache/pip/wheels/4b/a5/2d/4a070a801d3a3d93f033d3ee9728f470f514826e89952df3ea
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4

In [2]: from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from skimage import feature
from imutils import build_montages
from imutils import paths

```
import numpy as np
import cv2
import os
import pickle

In [3]:
import os, types
import pandas as pd
from boto3.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. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='0RRDOP2_khkK1GlrM4V7Yb4Shtk1VICnqe6IXRXUpce',
    ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='auth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'detectingparkinsonsdiseasesingma-donotdelete-pr-zkzewifj4ve7i'
object_key = 'dataset.zip'

streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a boto3.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

```
In [4]: from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_2.read()), 'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

```
In [5]: pwd
```

```
Out[5]: '/home/wsuser/work'
```

Path for train and test data

```
In [6]: trainingpath="/home/wsuser/work/dataset/spiral/training"
testingpath="/home/wsuser/work/dataset/spiral/testing"
```

Quantifying Images

```
In [7]: def quantify_image(image):
features = feature.hog(image, orientations=9,
pixels_per_cell=(10, 10),
cells_per_block=(2, 2),
transform_sqrt=True,
block_norm="L1")
return features
```

Loading Train Data and Test Data

```
In [8]: def load_split(path):
        imagePaths = list(paths.list_images(path))
        data = []
        labels = []

        for imagePath in imagePaths:
            label = imagePath.split(os.path.sep)[-2]

            image = cv2.imread(imagePath)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            image = cv2.resize(image, (200, 200))

            image=cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

            features = quantify_image(image)

            data.append(features)
            labels.append(label)

        return (np.array(data), np.array(labels))
```

Load the train and test data

```
In [9]: print("[INFO] loading data...")
        (X_train, y_train) = load_split(trainingpath)
        (X_test, y_test) = load_split(testingpath)

        [INFO] loading data...
```

Label Encoding

```
In [10]: le = LabelEncoder()
        y_train = le.fit_transform(y_train)
        y_test = le.transform(y_test)
        print(X_train.shape, y_train.shape)

        (72, 12996) (72,)
```

Model Building

Training The Model

```
In [11]: print("[INFO] training model")
        model = RandomForestClassifier(n_estimators=100)
        model.fit(X_train, y_train)

        [INFO] training model

Out[11]: RandomForestClassifier()
```

Testing The Model

```
In [12]: testingpath=list(paths.list_images(testingpath))
        idxs=np.arange(0,len(testingpath))
        idxs=np.random.choice(idxs,size=(25,),replace=False)
        images=[]
```

```
In [13]: for i in idxs:
        image=cv2.imread(testingpath[i])
        output=image.copy()

        # Load the input image,convert to grayscale and resize
```

```
In [13]: for i in idxs:
        image=cv2.imread(testingpath[i])
        output=image.copy()

        # Load the input image,convert to grayscale and resize

        output=cv2.resize(output,(128,128))
        image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        image=cv2.resize(image,(200,200))
        image=cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

        #quantify the image and make predictions based on the extracted feature using last trained random forest
        features=quantify_image(image)
        preds=model.predict([features])
        label=le.inverse_transform(preds)[0]
        #the set of output images
        if label=="healthy":
            color=(0,255,0)
        else:
            color=(0,0,255)

        cv2.putText(output,label,(3,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,color,2)
        images.append(output)

        #creating a montage
        montage=build_montages(images,(128,128),(5,5))[0]
        cv2.imshow("Output",montage)
        cv2.waitKey(0)
```

Model Evaluation

```
In [34]: predictions = model.predict(X_test)

cm = confusion_matrix(y_test, predictions).flatten()
print(cm)
(tn, fp, fn, tp) = cm
accuracy = (tp + tn) / float(cm.sum())
print(accuracy)

[14  1  3 12]
0.8666666666666667
```

Save The Model

```
In [15]: pickle.dump(model, open('parkinson.pkl', 'wb'))
```

Deployment

```
In [16]: !pip install -U ibm-watson-machine-learning

Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.8.9)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.26.7)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (21.3)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (4.8.2)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.26.0)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.3.4)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2022.9.24)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (1.20.3)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (1.15.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (3.3)
Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from importlib-metadata->ibm-watson-machine-learning) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm-watson-machine-learning) (3.0.4)
```

```
In [17]: # Now connect notebook ml service with api key and url

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

Authenticate and Set Space

```
In [18]: wml_credentials = {
    "apikey": "RYa2JTVisfgzBUBvFXnCYWUXLBDrntWzc9KGstjrtCS",
    "url": "https://us-south.ml.cloud.ibm.com" #For Dallas region
}
```

```
In [19]: wml_client = APIClient(wml_credentials)
```

```
In [20]: # Check the available deployments
```

```
wml_client.spaces.list()

Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
```

ID	NAME	CREATED
efa48345-def9-4aa5-b19f-4dd7d5f766ce	ParkinsonDiseaseDetection	2022-11-06T10:09:49.894Z

```
In [21]: SPACE_ID = "efa48345-def9-4aa5-b19f-4dd7d5f766ce"
```

```
In [22]: # Space id created default one
```

```
wml_client.set_default_space(SPACE_ID)
```

```
Out[22]: 'SUCCESS'
```

```
In [23]: # To check the environment
```

```
wml_client.software_specifications.list()

runtime-22.1-py3.9 12b83a17-2d48-5082-900f-0ab31bfbd3cb base
scikit-learn-0.22-py3.6 154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base
default_r3.6 1b70a6c3-ab34-4b87-8aa0-a4a3c8296a36 base
pytorch-onnx-1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880db0e7 base
kernel-spark3.3-py3.6 1c9e5454-f216-59dd-a20e-474a5c0f5988 base
pytorch-onnx_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bd37f base
tensorflow_2.1-py3.6 1eb25b84-d0ed-5dde-b6a5-3fbdf1665666 base
spark-mllib_3.2 20047f72-0a98-58c7-9ff5-a77b012eb8f5 base
tensorflow_2.4-py3.8-horovod 217c16f6-178f-56bf-92da-b19f20564c49 base
runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-a1b0-da66306ce58 base
do_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base
autoai-ts_3.8-py3.8 2aa0c932-798f-5ae9-abde-15e0c2402fb5 base
kernel-spark3.3-py3.6 2b73a275-7cbf-420b-a912-eae7f436e0bc base
kernel-spark3.3-py3.9 2b7961e2-e3b1-5abc-a491-4b2c8368839a base
pytorch_1.2-py3.6 2c8ef57d-2687-4b7d-acce-01f94976dac1 base
spark-mllib_2.3 2e51f700-bca0-4b0d-88dc-5c6791338875 base
pytorch-onnx-1.1-py3.6-edt 32983cea-3f32-4400-8965-dde874a8d67e base
spark-mllib_3.0-py37 36507ebe-8770-55ba-ab2a-eafe787690e9 base
spark-mllib_2.4 390d21f8-e58b-4fac-9c55-d7ceda621326 base
```

Save and Deploy the Model

```
In [24]: import sklearn
sklearn.__version__

Out[24]: '1.0.2'

In [25]: MODEL_NAME = "ParkinsonDiseaseDetection_DeployedModel"
DEPLOYMENT_NAME = "ParkinsonDiseaseDetection"

In [26]: # Set Python default version

software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
```

Create Model Properties to deploy the model

```
In [27]: # 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
}
```

```
In [28]: # Save Model

model_details = wml_client.repository.store_model(
    model = model,
    meta_props = model_props,
    training_data = X_train,
    training_target = y_train
)
```

```
In [29]: model_details

{'schemas': {'input': [{'fields': [{'name': 'f0', 'type': 'float'},
{'name': 'f1', 'type': 'float'},
{'name': 'f2', 'type': 'float'},
{'name': 'f3', 'type': 'float'},
{'name': 'f4', 'type': 'float'},
{'name': 'f5', 'type': 'float'},
{'name': 'f6', 'type': 'float'},
{'name': 'f7', 'type': 'float'},
{'name': 'f8', 'type': 'float'},
{'name': 'f9', 'type': 'float'},
{'name': 'f10', 'type': 'float'},
{'name': 'f11', 'type': 'float'},
{'name': 'f12', 'type': 'float'},
{'name': 'f13', 'type': 'float'},
{'name': 'f14', 'type': 'float'},
{'name': 'f15', 'type': 'float'},
{'name': 'f16', 'type': 'float'},
{'name': 'f17', 'type': 'float'},
{'name': 'f18', 'type': 'float'},
{'name': 'f19', 'type': 'float'}]}
```

```
In [30]: model_id = wml_client.repository.get_model_id(model_details)
model_id
```

```
Out[30]: '7d936b97-a55f-403a-9624-5ad06e18e6b0'
```

Deploy in props

```
In [31]: # Set meta

deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME: DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

```
In [32]: # Deploy

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

#####

Synchronous deployment creation for uid: '7d936b97-a55f-403a-9624-5ad06e18e6b0' started

#####

initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready

-----
successfully finished deployment creation, deployment_uid='cbe26007-da09-4ca5-919f-3b00aa88f433'
-----
```