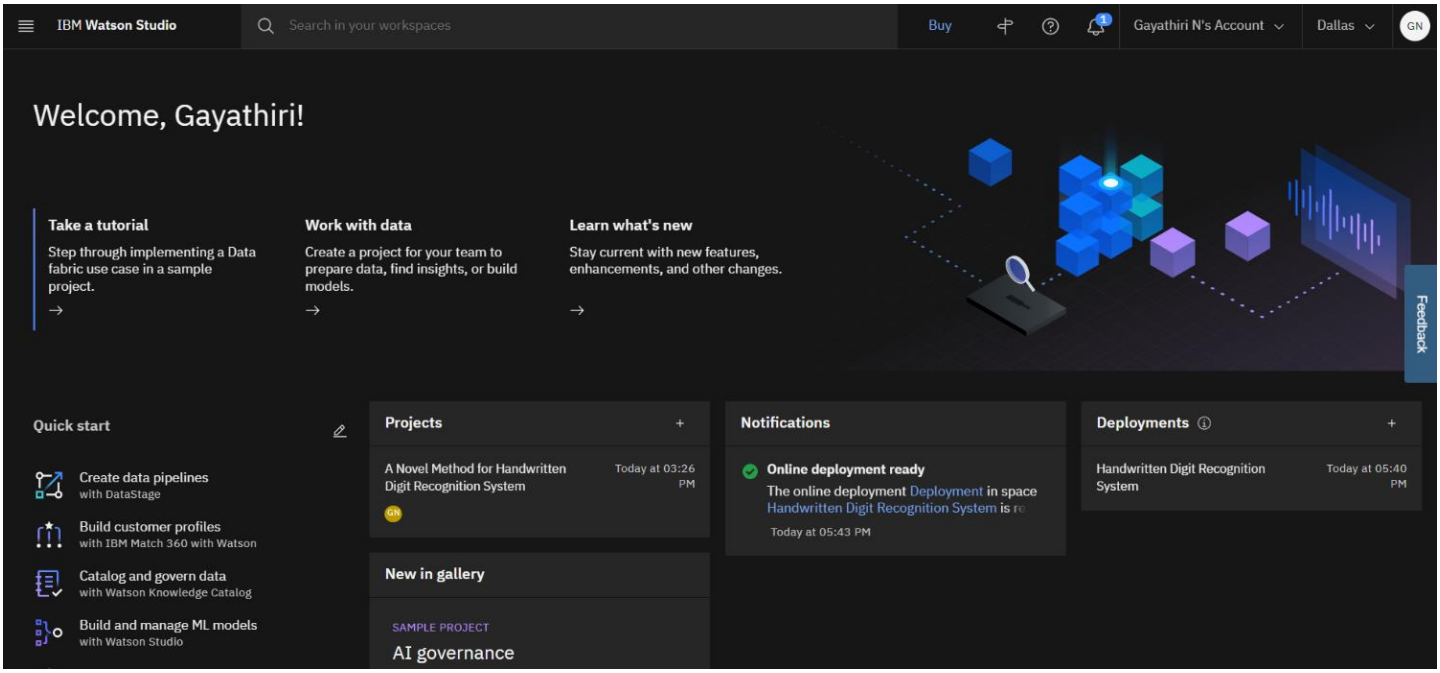
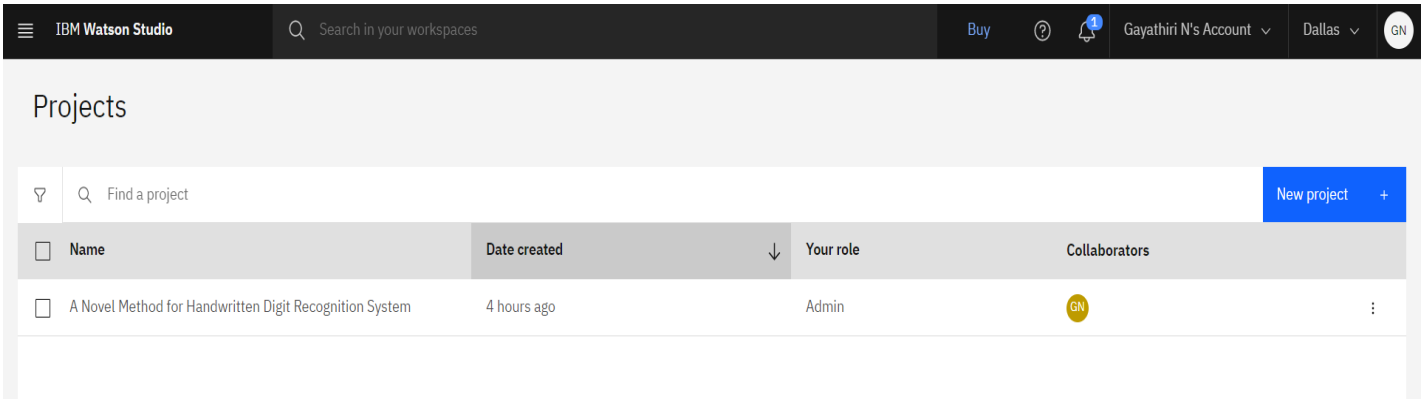
A Novel Method for Handwritten Digit Recognition

Team ID: PNT2022TMID35466

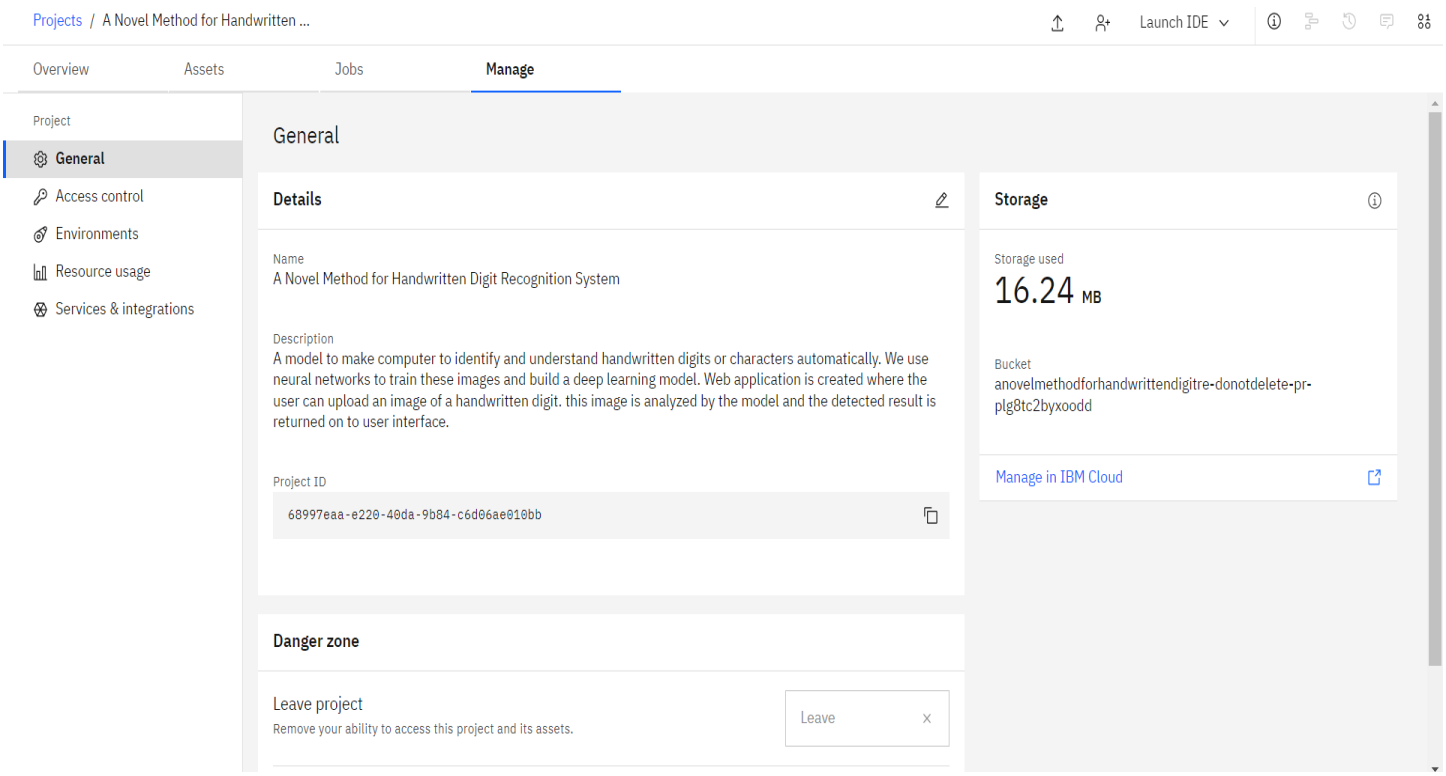
IBM CLOUD HOME PAGE



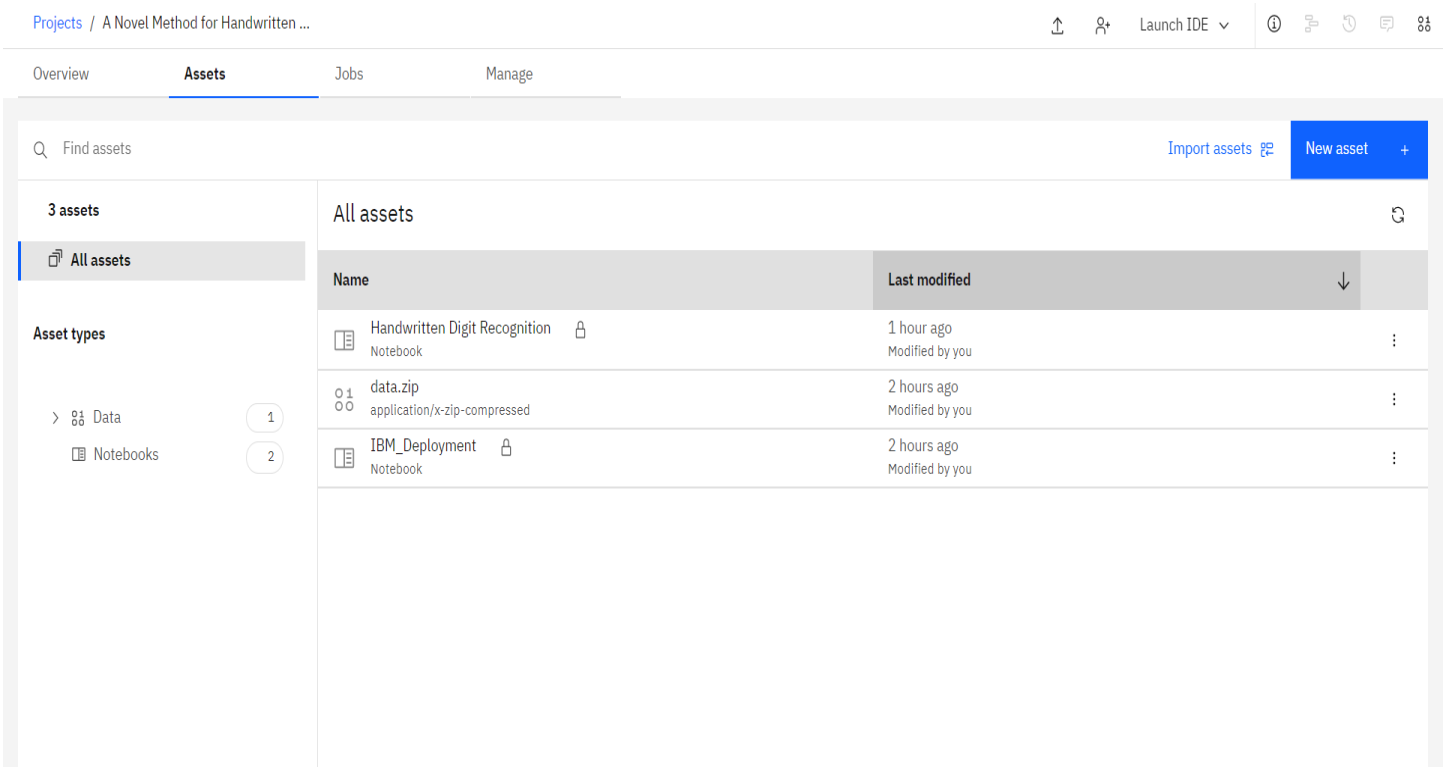
PROJECT DETAILS



PROJECT DESCRIPTION



PROJECT ASSETS



IMPORTING REQUIRED LIBRARIES:

Importing Required Libraries

In [1]:
```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import keras
from keras import layers
keras.backend.set_image_data_format('channels_last')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

LOADING INPUT DATASET

In [2]:
```
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. 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='l1SLNKuyTOTvyxuymCK02Yj3GQDrlAxjSLgrZRYD5R2G',
    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 = 'handwrittendigitrecognition-donotdelete-pr-yvxxug6r0c2f8y'
object_key = 'data.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.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 [3]:
```
from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

CNN MODEL:

```
def CNN():
    model = keras.Sequential()
    # CONV > CONV > BN > RELU > MAXPOOLING > DROPOUT
    model.add(layers.Conv2D(32, (3, 3), (1, 1), padding='valid', input_shape=(28, 28, 1), name='conv2d_1_1'))
    model.add(layers.Conv2D(32, (3, 3), (1, 1), padding='same', name='conv2d_1_2'))
    model.add(layers.BatchNormalization(name='bn_1'))
    model.add(layers.Activation('relu', name='relu_1'))
    model.add(layers.MaxPooling2D((2, 2), (2, 2), padding='valid', name='mp2d_1'))
    model.add(layers.Dropout(0.2, name='drop_1'))
    # CONV > CONV > BN > RELU > MAXPOOLING > DROPOUT
    model.add(layers.Conv2D(64, (3, 3), (1, 1), padding='valid', name='conv2d_2_1'))
    model.add(layers.Conv2D(64, (3, 3), (1, 1), padding='same', name='conv2d_2_2'))
    model.add(layers.BatchNormalization(name='bn_2'))
    model.add(layers.Activation('relu', name='relu_2'))
    model.add(layers.MaxPooling2D((2, 2), (2, 2), padding='valid', name='mp2d_2'))
    model.add(layers.Dropout(0.2, name='drop_2'))
    # FLATTEN > DENSE > CLASSIFICATION
    model.add(layers.Flatten())
    model.add(layers.Dense(100, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))

    return model
```

```
model = CNN()
```

```
model.compile(optimizer = "Adam", loss = "CategoricalCrossentropy", metrics = "accuracy")
```

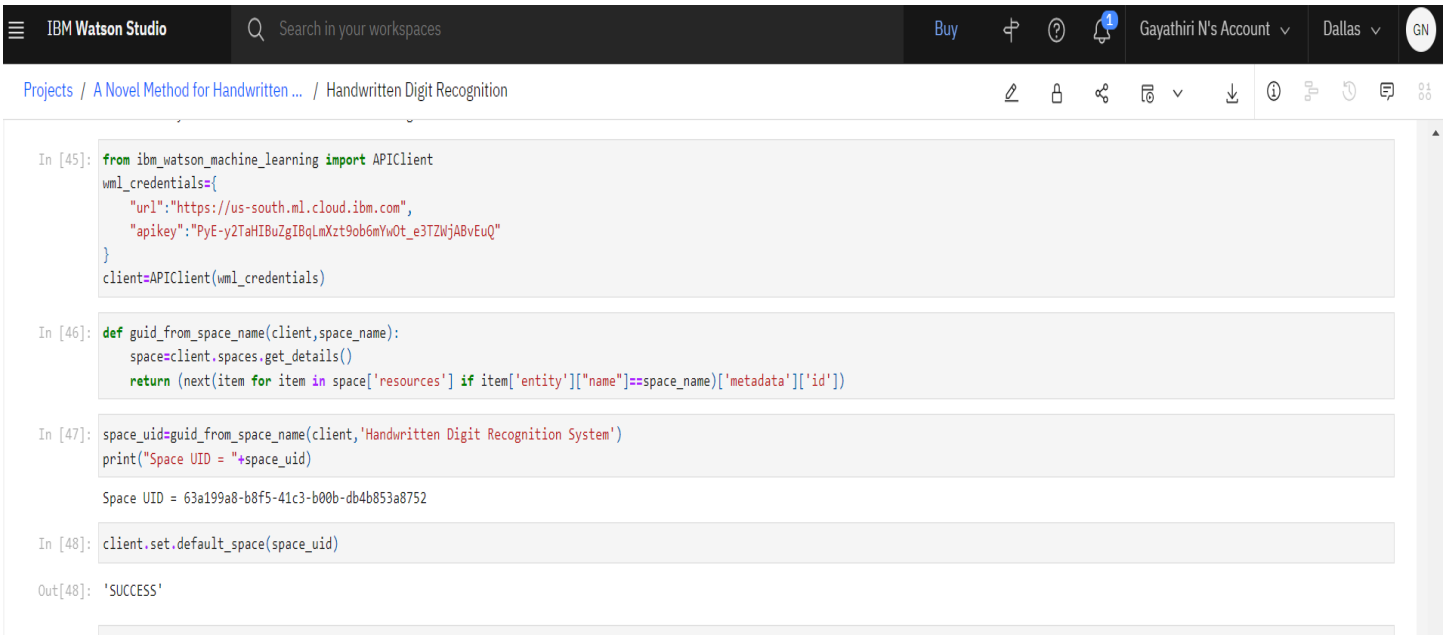
```
model.summary()
```

TRAINING THE MODEL

```
_________________________________________________________________
```

```
n [36]: training = model.fit(X_train, Y_train, validation_data = (X_val, Y_val), batch_size = 64, epochs = 5, verbose = 1 )
```

```
Epoch 1/5
525/525 [==============================] - 90s 170ms/step - loss: 0.2151 - accuracy: 0.9339 - val_loss: 0.0720 - val_accuracy: 0.9771
Epoch 2/5
525/525 [==============================] - 89s 170ms/step - loss: 0.0715 - accuracy: 0.9778 - val_loss: 0.0762 - val_accuracy: 0.9767
Epoch 3/5
525/525 [==============================] - 88s 167ms/step - loss: 0.0538 - accuracy: 0.9829 - val_loss: 0.0660 - val_accuracy: 0.9805
Epoch 4/5
525/525 [==============================] - 89s 169ms/step - loss: 0.0458 - accuracy: 0.9850 - val_loss: 0.0416 - val_accuracy: 0.9889
Epoch 5/5
525/525 [==============================] - 89s 169ms/step - loss: 0.0388 - accuracy: 0.9878 - val_loss: 0.0578 - val_accuracy: 0.9832
```



TESTING THE MODEL: