PES UNIVERSITY	**PES UNIVERSITY** **(Established under Karnataka Act No. 16 of 2013)** **100 Ft. Road, BSK III Stage, Bengaluru – 560 085** **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING** **SESSION: Jan-May 2021**

Course Title: Stochastic Models and Machine Learning	
Course code: UE20CS504	
Semester : I Sem	**Team Id: 6**
SRN: PES1PG20CS009	**Name: Bhavana G**
SRN: PES1PG20CS028	**Name: Pawan Kumar Doddamani**

ASSIGNMENT REPORT

Problem 1: Implement a Naive Bayes Classifier(NBC) , of English newspaper Head Lines into Politics, Sports, Education, Healthcare, Finance(5-Class Labels)

Description:

Given a sample headline we have to predict which of the above 5 class labels it belongs to.

Dataset Details:

1) The dataset consists of 6 columns and about 2,00,853 data entries.
2) The columns include a category, headline, authors, link, short description and date where category is the label indicating which category the headline belongs to.
3) Split the data into training and test datasets
4) Encoding of the tags needs to be done in order to convert the text tags into comparable tokens.

Steps of Implementation:

1) We have collected dataset from Kaggle who has collected it from HuffingtonPost.
2) The dataset has multiple categories so first we removed the irrelevant data.
3) Then we split the data into training and test datasets.
4) First we run CountVectorizer to give tokens to each word and then count no of times that token has occurred.
5) Then we run TfIdfTransformer where TF is Term Frequency(this summarizes how often a given word appears within a document) and IDF is Inverse Document Frequency(this downscales words that appear a lot across documents).

6) Finally our data is ready to be input to our model.

7) We use the Multinomial NB algorithm to train our model which is suitable with discrete features (e.g. word count for text classification).

8) Check the accuracy of our model from the test data using metrics.

Output Screenshots:

```
In [1]: import pandas as pd

        file_data = pd.read_json('News_Category_Dataset_v2.json', lines=True)
        file_data.head()
```

Out[1]:

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26

```
In [2]: file_data.shape
Out[2]: (200853, 6)
```

Fig 1: Importing Dataset into Pandas DataFrame

```
In [7]: # rename WELLNESS to HEALTHCARE and MONEY to FINANCE

        df = relevant_rows
        df['category'] = df['category'].replace(['WELLNESS', 'MONEY'], ['HEALTHCARE', 'FINANCE'])
        df.tail()
```

```
<ipython-input-7-4db0a047c698>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['category'] = df['category'].replace(['WELLNESS', 'MONEY'], ['HEALTHCARE', 'FINANCE'])
```

Out[7]:

	category	headline	short_description
200805	HEALTHCARE	This Is Only the Beginning: Surprising Advice ...	My great-aunt Ida loves to say, "This is only ...
200849	SPORTS	Maria Sharapova Stunned By Victoria Azarenka I...	Afterward, Azarenka, more effusive with the pr...
200850	SPORTS	Giants Over Patriots, Jets Over Colts Among M...	Leading up to Super Bowl XLVI, the most talked...
200851	SPORTS	Aldon Smith Arrested: 49ers Linebacker Busted ...	CORRECTION: An earlier version of this story i...
200852	SPORTS	Dwight Howard Rips Teammates After Magic Loss ...	The five-time all-star center tore into his te...

```
In [8]: df.shape
Out[8]: (58161, 3)
```

Fig 2: DataFrame after Preprocessing

```
In [8]: x = df.drop('category', axis='columns')
        y = df['category']

        x.head()
```

Out[8]:

	headline	short_description
13	Trump's Crackdown On Immigrant Parents Puts Mo...	Last month a Health and Human Services officia...
14	'Trump's Son Should Be Concerned': FBI Obtaine...	The wiretaps feature conversations between Ale...
15	Edward Snowden: There's No One Trump Loves Mor...	But don't count on Robert Mueller to nail him,...
16	Booyah: Obama Photographer Hilariously Trolls ...	Just a peeping minute.
17	Ireland Votes To Repeal Abortion Amendment In ...	Irish women will no longer have to travel to t...

```
In [10]: y.head()

Out[10]: 13    POLITICS
         14    POLITICS
         15    POLITICS
         16    POLITICS
         17    POLITICS
         Name: category, dtype: object
```

Fig 3: Separate Features in x and Labels in y

```
In [10]: from sklearn.model_selection import train_test_split

         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

         x_train.shape

Out[10]: (46528, 2)

In [11]: x_test.shape

Out[11]: (11633, 2)
```

Fig 4: Split Test and Training Data

```
In [9]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.pipeline import Pipeline

        text_clf = Pipeline([('vector', TfidfVectorizer()), ('clf', MultinomialNB())])
```

Fig 5: Create a Pipeline to run TfidfVectorizer and MultinomialNB in sequence

```
In [45]: import time

         # train the model
         start_time = time.time()
         text_clf.fit(x_train['headline'], y_train)
         print("Training time(in seconds)", time.time() - start_time)

         # test the model
         predicted = text_clf.predict(x_test['headline'])

         Training time(in seconds) 0.7834486961364746
```

Fig 6: Train and Test the Model

```
In [26]: sample_prediction = text_clf.predict(['10 ways to beat stress in 10 minutes or less'])
         sample_prediction

Out[26]: array(['HEALTHCARE'], dtype='<U10')
```

Fig 7: Sample Prediction

Interpretation of efficiency:

1) The model took 0.78 seconds to train.

```
In [45]: import time

         # train the model
         start_time = time.time()
         text_clf.fit(x_train['headline'], y_train)
         print("Training time(in seconds)", time.time() - start_time)

         # test the model
         predicted = text_clf.predict(x_test['headline'])

         Training time(in seconds) 0.7834486961364746
```

2) Accuracy score when checked with the test data was 84.83%

```
In [13]: from sklearn import metrics
         from sklearn.metrics import accuracy_score

         print(accuracy_score(y_test, predicted))

         0.8483624172612396
```

3) Confusion matrix of trained data is as below in the following order for rows and column indexes

   0 – EDUCATION

   1 – FINANCE

   2 – HEALTHCARE

   3 – POLITICS

   4 – SPORTS

```
In [31]: metrics.confusion_matrix(y_test, predicted)

Out[31]: array([[   0,    0,   58,  170,    0],
                [   0,    1,   79,  238,    0],
                [   0,    0, 3106,  534,    1],
                [   0,    0,  109, 6335,    0],
                [   0,    0,   96,  565,  341]], dtype=int64)
```

End of Problem 1

Problem 2: For the same Dataset in Problem 1 apply One-Versus-Rest SVM and classify.

Description:

Given a sample headline we have to predict which of the above 5 class labels it belongs to.

Dataset Details:

1) The dataset consists of 6 columns and about 2,00,853 data entries.
2) The columns include a category, headline, authors, link, short description and date where category is the label indicating which category the headline belongs to.
3) Split the data into training and test datasets
4) Encoding of the tags needs to be done in order to convert the text tags into comparable tokens.

Steps of Implementation:

1) The pre processed and split data from Problem 1 is used.
2) First we run TfIdfVectorizer which does the work of both CountVectorizer and TfIdfTransformer together.
3) Then our data is ready to be input to our model.
4) We use the Linear SVC to train our model. Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
5) Check the accuracy of our model from the test data using metrics.

Output Screenshots:

```
In [46]: from sklearn.svm import SVC

         model = Pipeline([('vect', TfidfVectorizer()), ('svm', SVC(kernel='linear'))])

         start_time = time.time()
         model.fit(x_train['headline'], y_train)
         print("Training time(in seconds)", time.time() - start_time)

         Training time(in seconds) 280.57390904426575

In [17]: model.score(x_test['headline'], y_test)
Out[17]: 0.9100833834780366
```

Fig 1: Training and testing the model

```
In [40]: sample_pred = model.predict(['10 ways to beat stress in 10 minutes or less'])
         sample_pred
Out[40]: array(['HEALTHCARE'], dtype=object)
```

Fig 2: Sample Prediction

Interpretation of efficiency:

1) The model took 120 seconds to train.

```
In [46]: from sklearn.svm import SVC

         model = Pipeline([('vect', TfidfVectorizer()), ('svm', SVC(kernel='linear'))])

         start_time = time.time()
         model.fit(x_train['headline'], y_train)
         print("Training time(in seconds)", time.time() - start_time)

         Training time(in seconds) 280.57390904426575
```

2) Accuracy score when checked with the test data was 91.00%

```
In [17]: model.score(x_test['headline'], y_test)

Out[17]: 0.9100833834780366
```

3) Confusion matrix of trained data is as below with

   0 – EDUCATION

   1 – FINANCE

   2 – HEALTHCARE

   3 – POLITICS

   4 – SPORTS

```
In [37]: from sklearn import metrics

         y_pred = model.predict(x_test['headline'])
         metrics.confusion_matrix(y_test, y_pred)

Out[37]: array([[  82,    6,   56,   82,    2],
                [   5,  156,   70,   81,    6],
                [   7,   22, 3387,  189,   36],
                [  14,   18,  253, 6113,   46],
                [   1,    5,   72,  138,  786]], dtype=int64)
```

End of Problem 2

Problem 3: Implement a Multi Layer (One Input, One Output and One or more Hidden Layers)ANN for handwritten digit classification using MNIST dataset.

Description:

Given a handwritten digit should be able to identify which digit it is using ANN. We are required to design and use a multilayer ANN or CNN to solve this problem.

Dataset Details:

1) We have imported the MNIST dataset from keras library.

2) The dataset has 60,000 trained different handwritten digit images and 10,000 different handwritten test images.

Structure of ANN:

We are capable of using many different layers in a convolutional neural network. However, convolution, pooling, and fully connect layers are the most important ones.

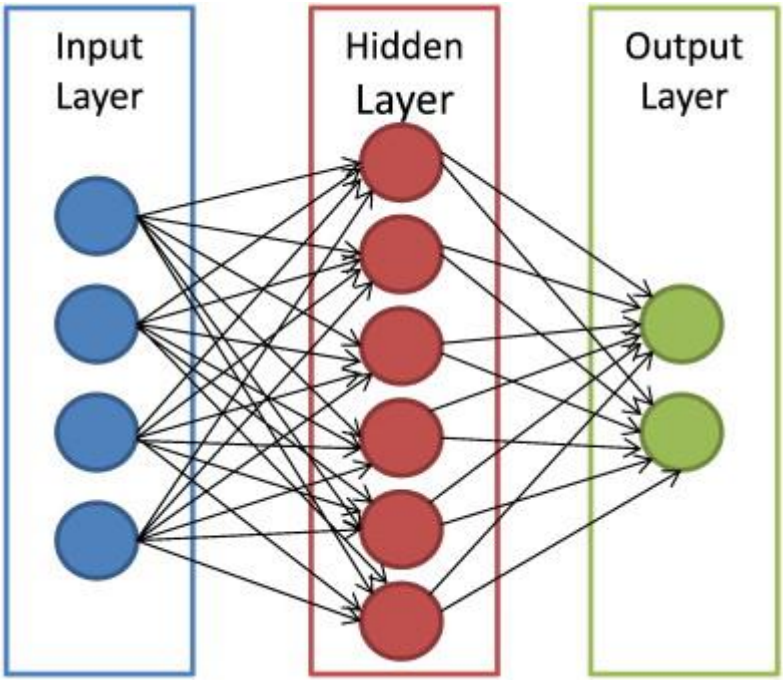

Fig 1: Structure of ANN

Convolutional Layers:

Convolutional layer is the very first layer where we extract features from the images in our datasets. Due to the fact that pixels are only related with the adjacent and close pixels, convolution allows us to preserve the relationship between different parts of an image. Convolution is basically filtering the image with a smaller pixel filter to decrease the size of the image without losing the relationship between pixels.

Pooling Layer:

When constructing CNNs, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the representation to reduce the parameter counts which reduces the computational complexity. In addition, pooling layers also helps with the overfitting problem. Basically, we select a pooling size to reduce the amount of the parameters by selecting the maximum, average, or sum values inside these pixels. Max Pooling, one of the most common pooling techniques, may be demonstrated as follows:

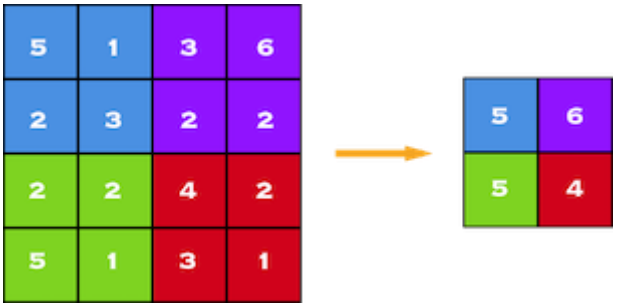

Fig 2: Max Pooling by 2 x 2

A Set of Fully Connected Layer:

A fully connected network is our RegularNet where each parameter is linked to one another to determine the true relation and effect of each parameter on the labels. Since our time-space complexity is vastly reduced thanks to convolution and pooling layers, we can construct a fully connected network in the end to classify our images. A set of fully connected layers looks like this:

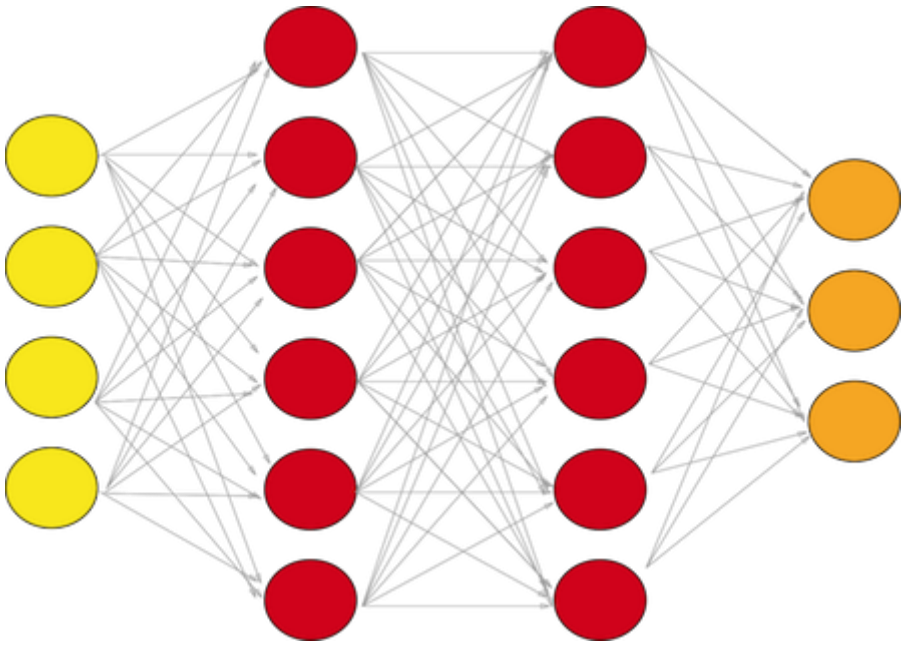

Fig 3: A fully connected layer with two hidden layers

Implementation of ANN:

In [21]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
```

In [22]:
```
model=Sequential()
# first convolution layer
model.add(Conv2D(64,(3,3), input_shape= x_trainr.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

# second convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

# third convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

#fully connected layer 1
model.add(Flatten())
model.add(Dense(64))
model.add(Activation("relu"))

#fully connected layer2
model.add(Dense(32))
model.add(Activation("relu"))


#last fully connected layer
model.add(Dense(10))
model.add(Activation('softmax'))
```

In [23]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 64)        640
_________________________________________________________________
activation (Activation)      (None, 26, 26, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
activation_1 (Activation)    (None, 11, 11, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
activation_2 (Activation)    (None, 3, 3, 64)          0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                330
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0
=================================================================
Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0
_________________________________________________________________
```

Steps of Implementation:

1) We have first imported the python libraries and the dataset is loaded.

2) Then the dataset is split into training set and test set.

3) Then we scaled to the standard size 28x28 pixels and it is converted to gray scale.

4) And the model is built by using sequential which is directly obtained from keras.

5) By using sequential model the layers are build one after the other in a sequence.

6) Conv2D layer converts the images into 2D matrix and there are 62 nodes in the first layer to 3x3 window size and we are reducing it to 2x2 and so on using Maxpooling.

7) The activation function transforms the weighted sum input into the output from the nodes in a layer.

8) The model is trained for 5 iterations and we can also observe that accuracy increases for every iteration.

9) MNIST digit seven is predicted when test data is applied as input to the trained model and the model correctly predicts the image.

Output Screenshots:

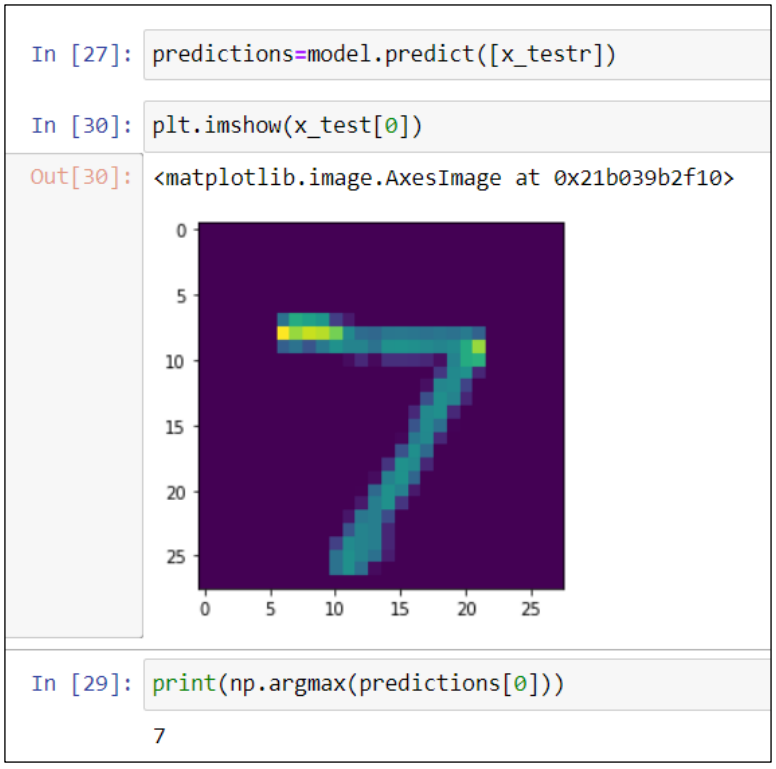

Fig 1: Prediction of digit 7

In the above snapshot we can see that, from the MNIST digit seven is predicted when test data is applied as input to the trained model and the model correctly predicts the image.

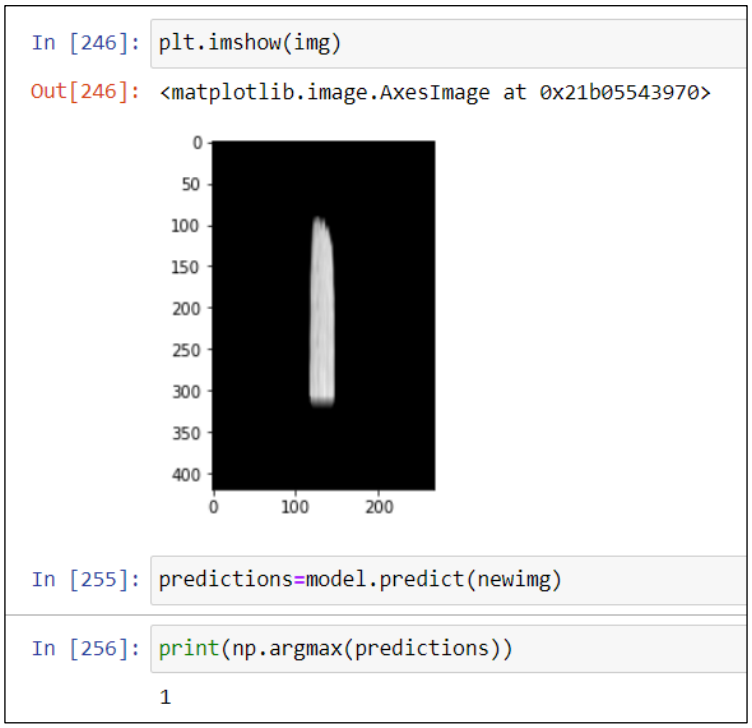

Fig 2: Prediction of digit 1

In the above snapshot we can see that, digit one written in MS paint and is given as input to the model and the model correctly predicts the image.

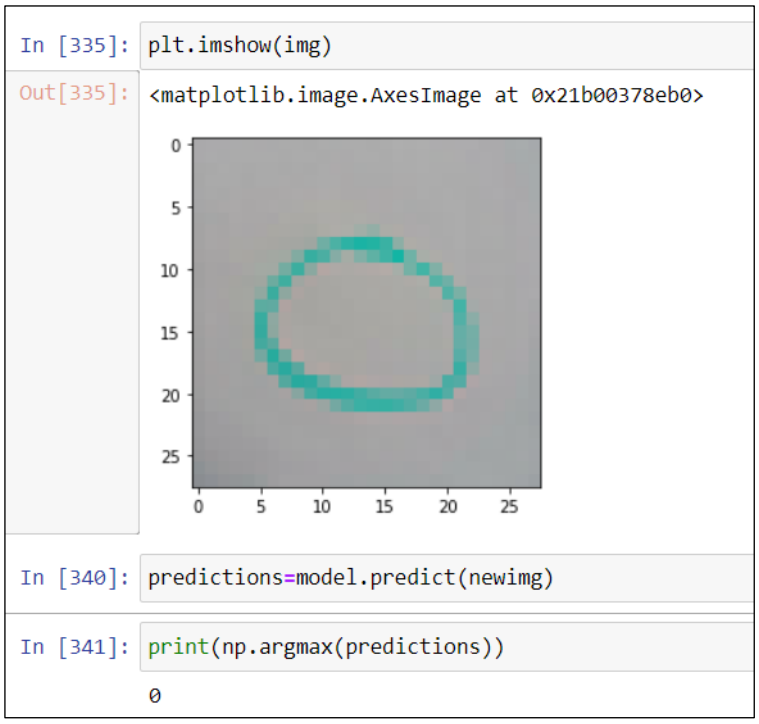

Fig 3: Prediction of digit 0

In the above snapshot we can see that, digit zero written on paper is given as input to the model and the model correctly predicts the image.

Interpretation of efficiency:

On training data upto 5 iterations

```
In [25]: model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=['accuracy'])
         model.fit(x_trainr,y_train,epochs=5,validation_split=0.3)  #training the model
         # categorical cross entropy. It compares the predicted label and true label and calculates the loss

         Epoch 1/5
         1313/1313 [==============================] - 40s 28ms/step - loss: 0.6877 - accuracy: 0.7783 - val_loss: 0.1315 - val_accuracy:
         0.9566
         Epoch 2/5
         1313/1313 [==============================] - 35s 27ms/step - loss: 0.1076 - accuracy: 0.9666 - val_loss: 0.0866 - val_accuracy:
         0.9717
         Epoch 3/5
         1313/1313 [==============================] - 37s 29ms/step - loss: 0.0757 - accuracy: 0.9768 - val_loss: 0.0731 - val_accuracy:
         0.9782
         Epoch 4/5
         1313/1313 [==============================] - 38s 29ms/step - loss: 0.0569 - accuracy: 0.9827 - val_loss: 0.0818 - val_accuracy:
         0.9727
         Epoch 5/5
         1313/1313 [==============================] - 42s 32ms/step - loss: 0.0416 - accuracy: 0.9866 - val_loss: 0.0574 - val_accuracy:
         0.9812
```

Accuracy of the model is 98.26%

```
In [26]: #Evaluating on test data set
         test_loss, test_acc=model.evaluate(x_testr,y_test)
         print("Test loss on 10,000 test samples",test_loss)
         print("Validation Accuracy on 10,000 test samples",test_acc)

         313/313 [==============================] - 2s 6ms/step - loss: 0.0614 - accuracy: 0.9826
         Test loss on 10,000 test samples 0.06137567758560181
         Validation Accuracy on 10,000 test samples 0.9825999736785889
```

Learning Outcomes:

1) Learnt Pandas
2) Learnt how Naïve Bayes Classifier works
3) Understood where Support Vector Machines can be applied and how Kernel tricks work
4) Leant how to visualize our dataset
5) Learnt how to measure the outcome of learned model using metrics
6) Learnt image recognition by OpenCV library
7) Learnt how to Scale and create CNN layers
8) Learnt how to train the Model

References:

1) YouTube
2) Stack overflow and Medium to help solve a few errors.

End of Problem 3