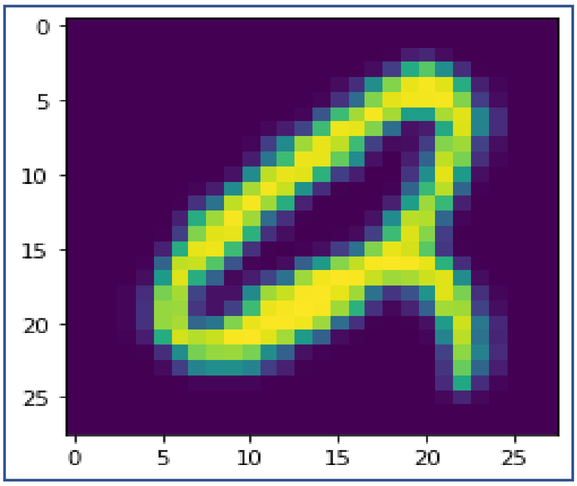
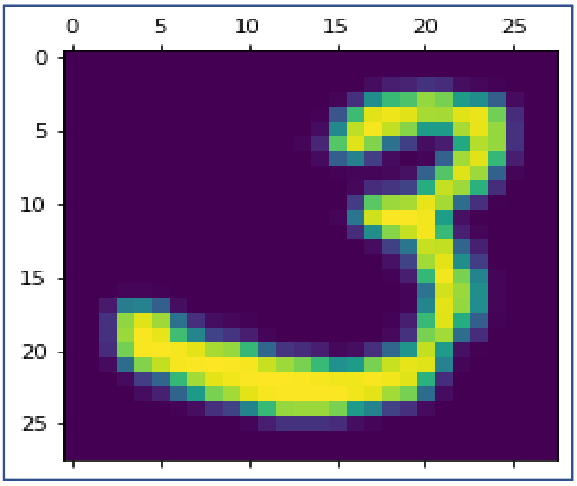
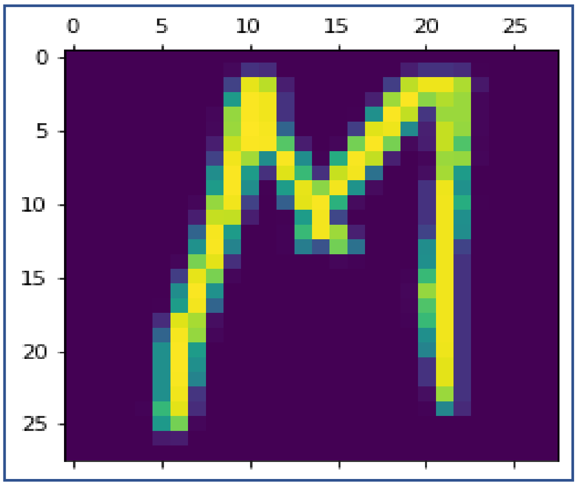
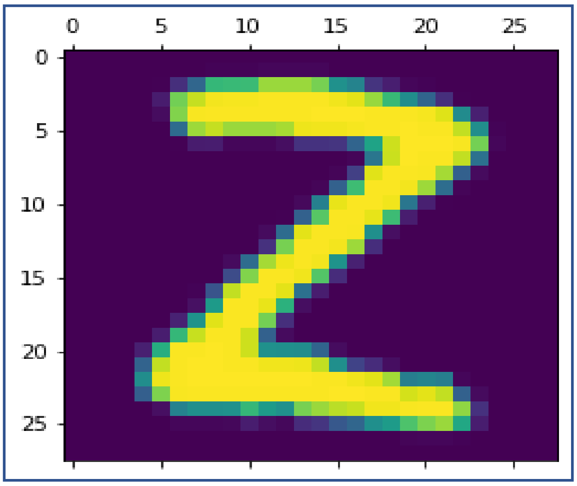
ALPHANUMERIC RECOGNITION

PROJECT FOR LEARNING PURPOSE









SUBMITTED BY: PRATYUSH SINGH

SUBMITTED TO: MISS MALLIKA SRIVASTAVA

BATCH: 21DAT-063

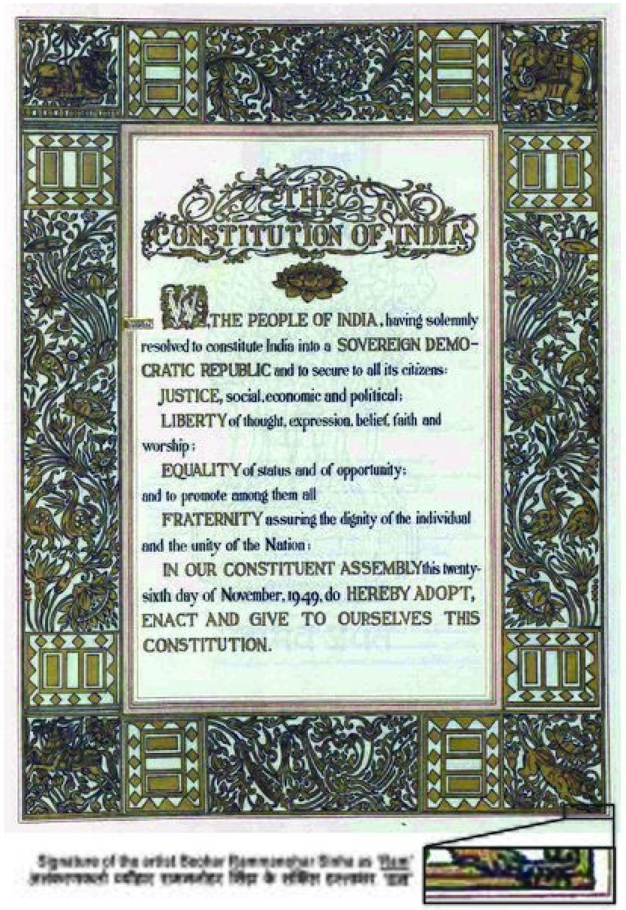
CONTENT TABLE:

IMAGE AND TABLE CONTENT:

ABSTRACT: This project is small ML, which in intended to help others(on opensource) to get the basic taste of ML related projects.

PROJECT SUMMARY: This project is based on Alphanumeric recognition capability using EMNIST byclass DATASET using Sklearn Logistic regression model using saga solver optimisation.

OBJECTIVES:



THE CONSTITUTION OF INDIA

WE, THE PEOPLE OF INDIA, having solemnly resolved to constitute India into a SOVEREIGN DEMOCRATIC REPUBLIC and to secure to all its citizens:

JUSTICE, social, economic and political;

LIBERTY of thought, expression, belief, faith and worship;

EQUALITY of status and of opportunity;

and to promote among them all

FRATERNITY assuring the dignity of the individual and the unity of the Nation;

IN OUR CONSTITUENT ASSEMBLY this twenty-sixth day of November, 1949, do HEREBY ADOPT, ENACT AND GIVE TO OURSELVES THIS CONSTITUTION.

Signature of the artist Beohar Rammanohar Sinha as 'Ram'
[illegible]

- ✓ Idea of making this project in my mind come, because I feel it is very difficult in ML, to get proper data and logics for preparing ML model.

- ✓ So to help people in opensource I decided to work on this small project, so that they can get a help while exploring the basics of the ML courses

- ✓ I have decided to upload the project in opensource(Github) that others can take help in understanding basics of python ML.

Details of Process:

A. IMPORTING DATASET: The Dataset used for this project is EMNIST byclass dataset, i.e these 4 give files:

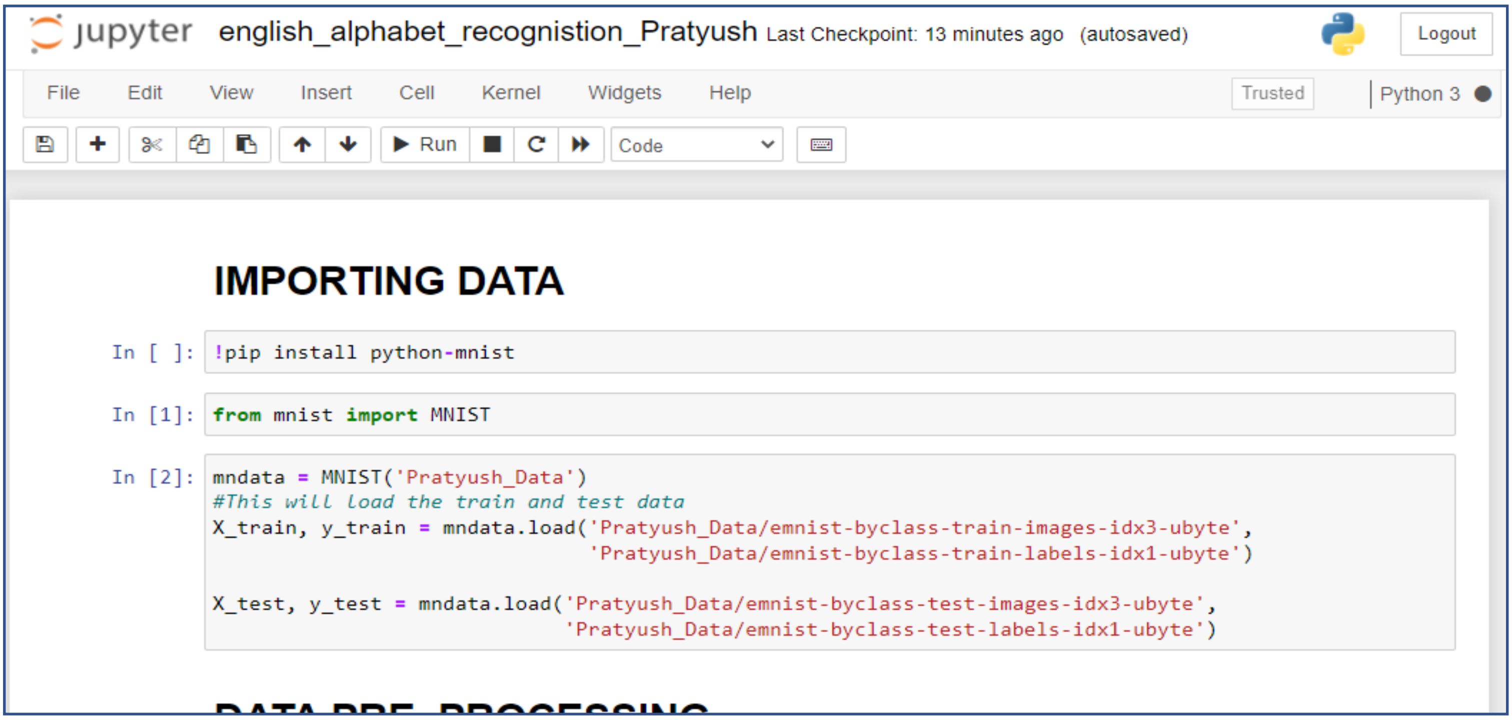
- emnist-byclass-train-images-idx3-ubyte
- emnist-byclass-train-labels-idx1-ubyte

→ For training purpose

- emnist-byclass-test-images-idx3-ubyte
- emnist-byclass-test-labels-idx1-ubyte

→ For testing purpose

For reading these datasets, I used python-mnist package as shown in figure:


jupyter
english_alphabet_recognistion_Pratyush Last Checkpoint: 13 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

IMPORTING DATA

```
In [ ]: !pip install python-mnist
```

```
In [1]: from mnist import MNIST
```

```
In [2]: mndata = MNIST('Pratyush_Data')
        #This will load the train and test data
        X_train, y_train = mndata.load('Pratyush_Data/emnist-byclass-train-images-idx3-ubyte',
                                       'Pratyush_Data/emnist-byclass-train-labels-idx1-ubyte')

        X_test, y_test = mndata.load('Pratyush_Data/emnist-byclass-test-images-idx3-ubyte',
                                     'Pratyush_Data/emnist-byclass-test-labels-idx1-ubyte')
```

DATA PRE-PROCESSING

FIGURE 1 IMPORTINF DATA

Here, I saved stored all my 4 datasets in a folder and named it randomly with name 'Pratyush_Data'.

And stored the information in X_train, y_train, X_test, y_test respectively.

The numerical encoding of EMNIST dataset is as follows(here I am informing about the meaning of values written in y_train and y_test):

NUMERICAL ENCODING	RESPECTIVE DIGITS/ALPHABETS
0	0
1	1
2	2
3	3
.	.
.	.
9	9
10	A
11	B
12	C
13	D
.	.
.	.
35	Z
36	a
37	b
38	c
39	d
.	.
.	.
60	y
61	z

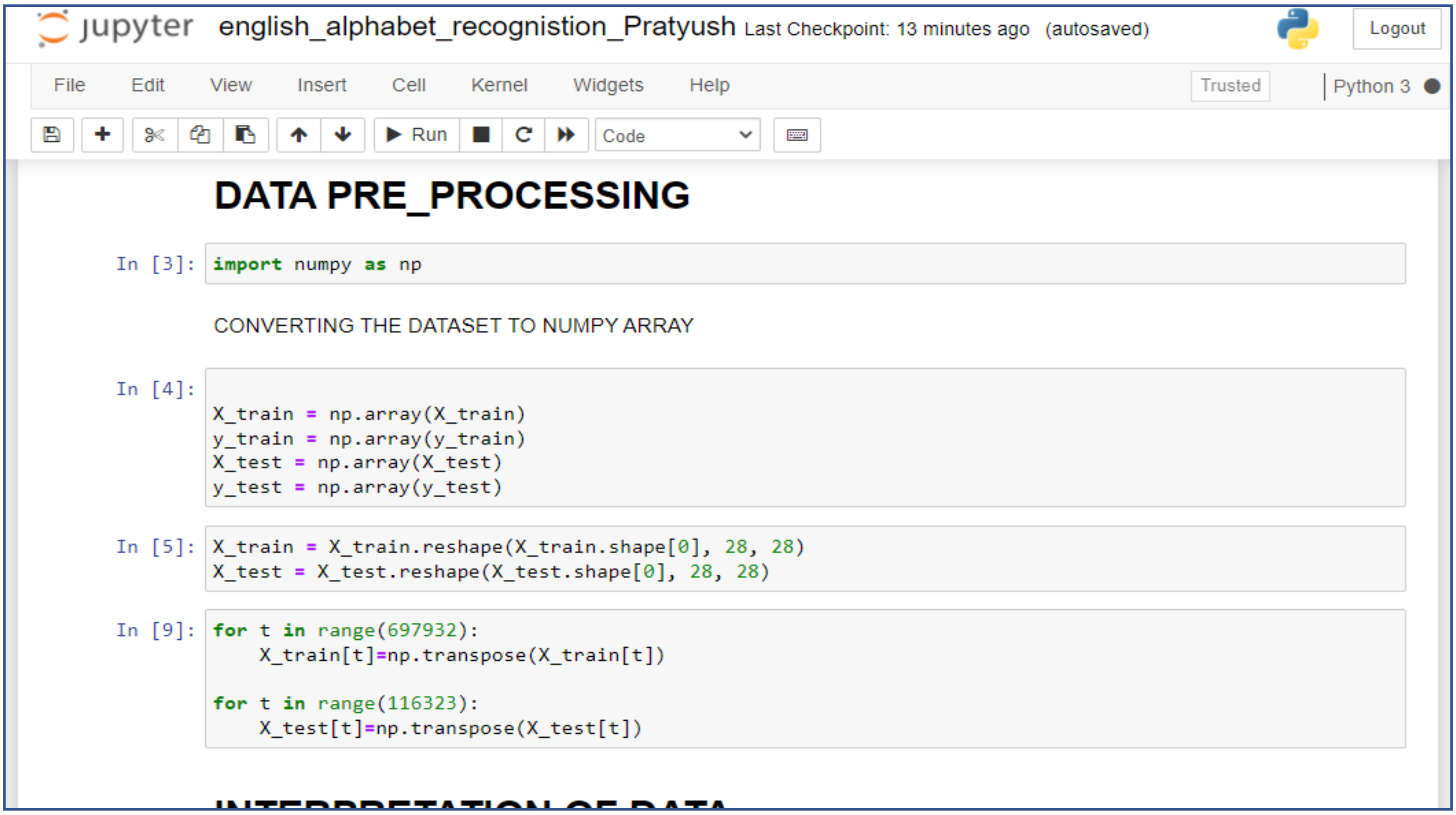
TABLE-1: EXPAINING EMNIST BYCLASS DATASET ENCODING

B. DATA PRE-PROCESSING (OPTIMISING DATA FOR MODEL): The pre-processing required following steps:

a. Converting X_train and X_test (i.e images) to NumPy arrays.

b. Reshaping all images into 28*28 matrix.

c. Taking Transpose of 2-D array for reversing and rotating all train and test images to proper orientation.


jupyter
english_alphabet_recognistion_Pratyush Last Checkpoint: 13 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

DATA PRE_PROCESSING

```
In [3]: import numpy as np
```

CONVERTING THE DATASET TO NUMPY ARRAY

```
In [4]:
        X_train = np.array(X_train)
        y_train = np.array(y_train)
        X_test = np.array(X_test)
        y_test = np.array(y_test)

In [5]: X_train = X_train.reshape(X_train.shape[0], 28, 28)
        X_test = X_test.reshape(X_test.shape[0], 28, 28)

In [9]: for t in range(697932):
            X_train[t]=np.transpose(X_train[t])

        for t in range(116323):
            X_test[t]=np.transpose(X_test[t])
```

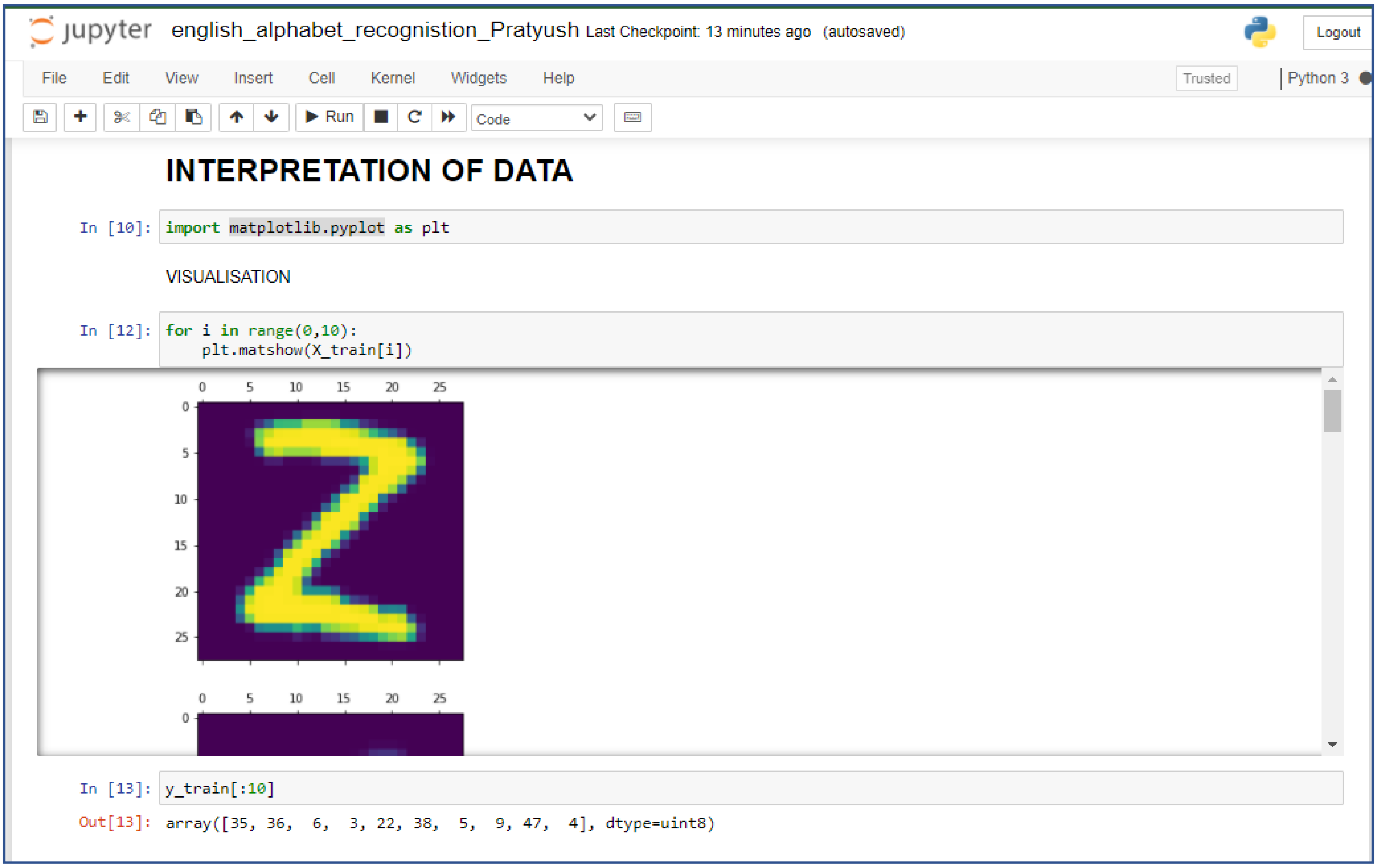
INTERPRETATION OF DATA

FIGURE 2: DATA PRE-PROCESSING

C. **INTERPRETATION OF DATA:** Now using matplotlib and NumPy ,we will check whether out data is properly pre-processed or not. We will do this as following:

a. Using matplotlib.pyplot.matshow, we will check our X_train dataset

b. And using NumPy we will check our y_train dataset.



INTERPRETATION OF DATA

```
In [10]: import matplotlib.pyplot as plt
```

VISUALISATION

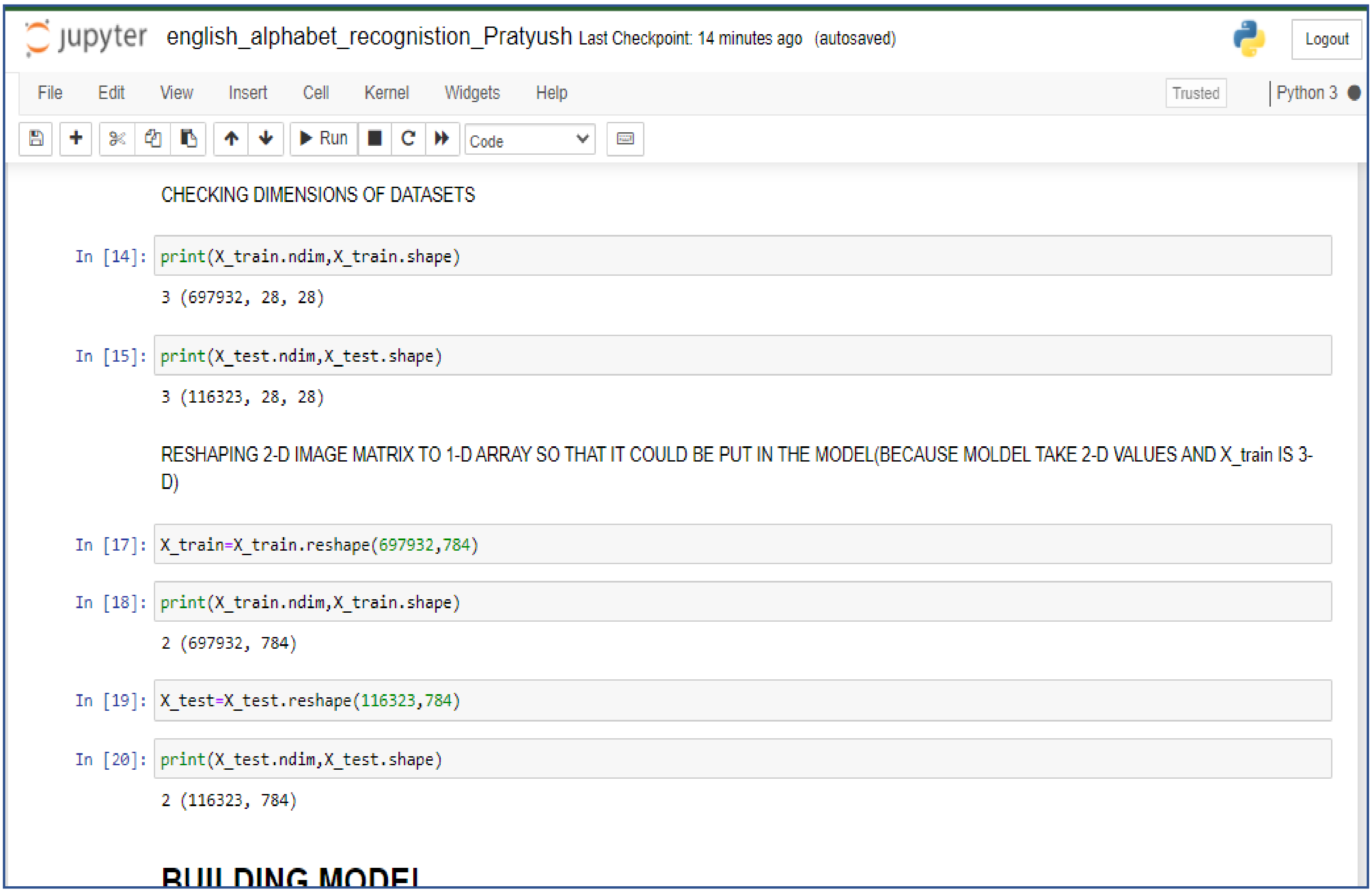
```
In [12]: for i in range(0,10):
             plt.matshow(X_train[i])
```

```
In [13]: y_train[:10]
Out[13]: array([35, 36,  6,  3, 22, 38,  5,  9, 47,  4], dtype=uint8)
```

FIGURE 3: INTERPRETATION OF DATA

And from the code-snippet (provided in the link) we can clearly see that out training data set is complete fine.

But we know that logistic regression take a 2-D array as its first parameter (that is our X_train) and 1-D array as its second parameter(that is our y_train).
But our X_train is at present group of images that is a 3-D array. So first we have to convert 2-D image matrices to !-D array as shown:


jupyter english_alphabet_recognistion_Pratyush Last Checkpoint: 14 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

CHECKING DIMENSIONS OF DATASETS

```
In [14]: print(X_train.ndim,X_train.shape)
         3 (697932, 28, 28)

In [15]: print(X_test.ndim,X_test.shape)
         3 (116323, 28, 28)
```

RESHAPING 2-D IMAGE MATRIX TO 1-D ARRAY SO THAT IT COULD BE PUT IN THE MODEL(BECAUSE MOLDEL TAKE 2-D VALUES AND X_train IS 3-D)

```
In [17]: X_train=X_train.reshape(697932,784)

In [18]: print(X_train.ndim,X_train.shape)
         2 (697932, 784)

In [19]: X_test=X_test.reshape(116323,784)

In [20]: print(X_test.ndim,X_test.shape)
         2 (116323, 784)
```

BUILDING MODEL

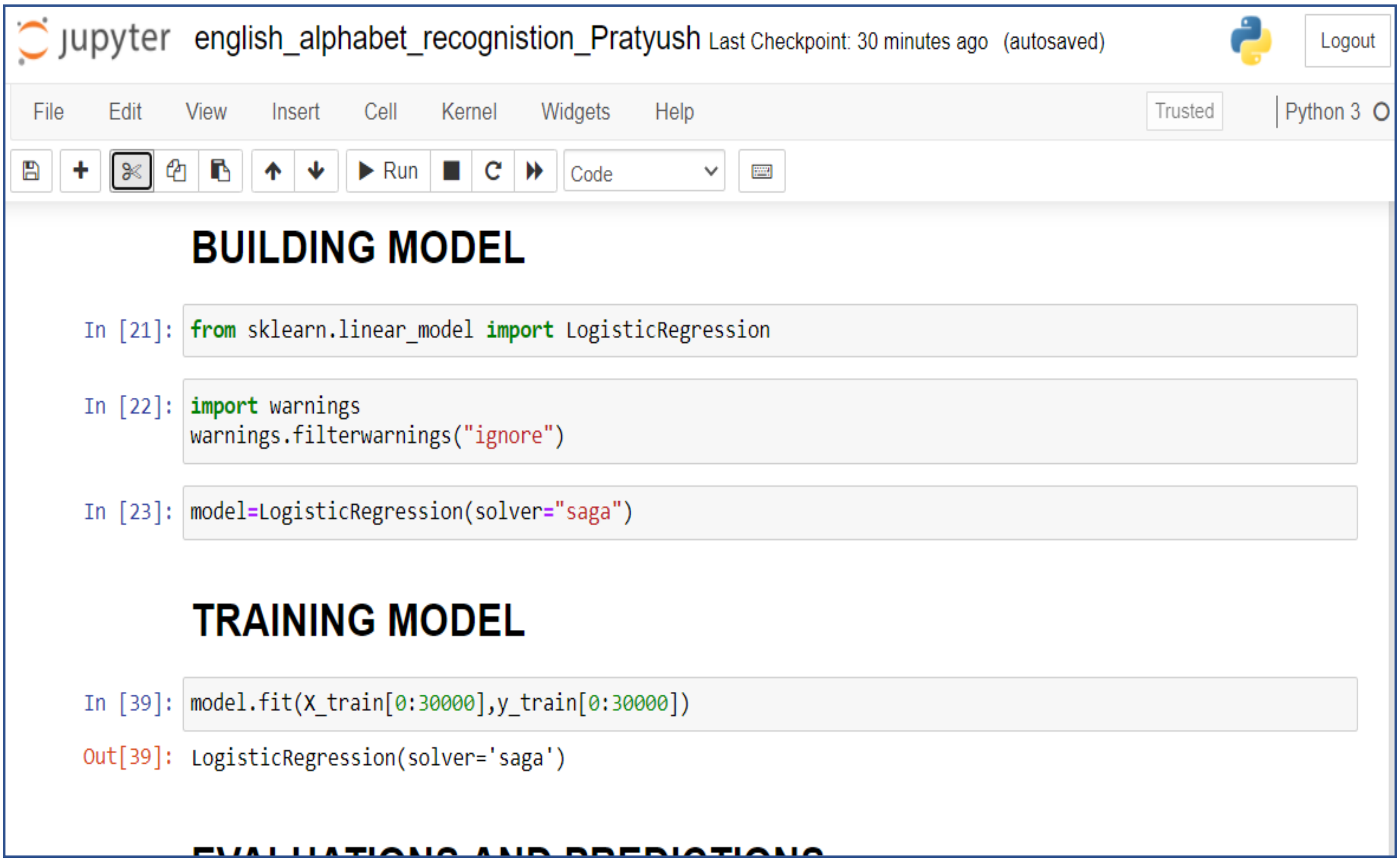
FIGURE 4: 2-D IMAGE MATRIX TO 1-D ARRAY CONVERSION

Now our X_train and Y_train are perfectly ready for training model.

D. BUILDING AND TRAINING MODEL:

Here for defining data model, I have used Logistic regression. We know that Logistic Regression is the Supervised Learning Algorithm for solving classification problems. Hence can be used in this problem.

In backend this regression solver has many parameters that train the model like Regularization parameter, hypothesis function, theta - fitting parameter etc.


jupyter
english_alphabet_recognistion_Pratyush Last Checkpoint: 30 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

BUILDING MODEL

```
In [21]: from sklearn.linear_model import LogisticRegression

In [22]: import warnings
         warnings.filterwarnings("ignore")

In [23]: model=LogisticRegression(solver="saga")
```

TRAINING MODEL

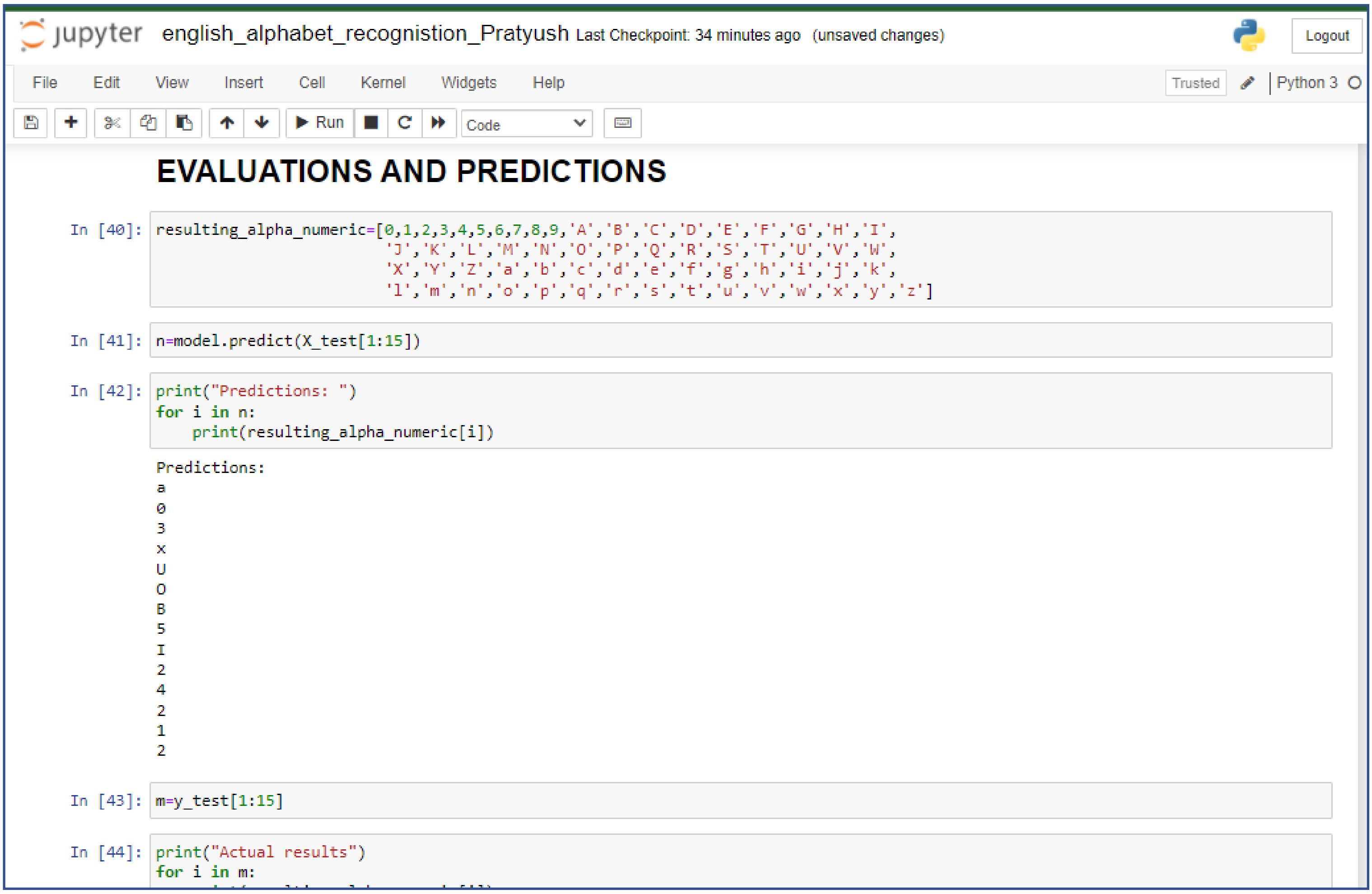
```
In [39]: model.fit(X_train[0:30000],y_train[0:30000])

Out[39]: LogisticRegression(solver='saga')
```

FIGURE 5: BUILDING AND TRAINING MODEL

Even X_train contains 697932 images(1-D arrays for model) but I used first 30000 images, because training with all the images was taking a lot of time.

Here solver is Algorithm used in the optimization problem, And for multiclass large data classification ‘saga’ is preferred.

E. TESTING AND EVALUATION:

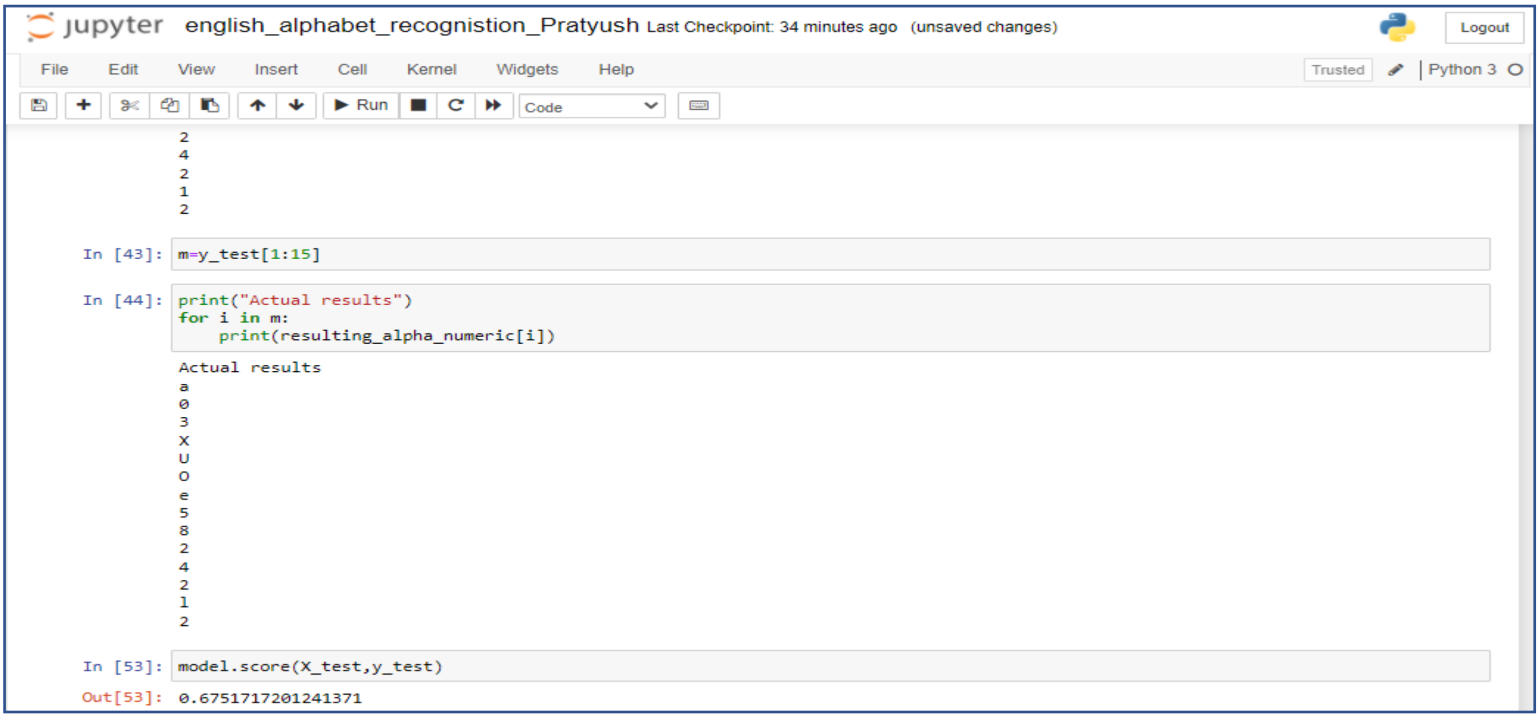

jupyter english_alphabet_recognistion_Pratyush Last Checkpoint: 34 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

EVALUATIONS AND PREDICTIONS

```
In [40]: resulting_alpha_numeric=[0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F','G','H','I',
                                  'J','K','L','M','N','O','P','Q','R','S','T','U','V','W',
                                  'X','Y','Z','a','b','c','d','e','f','g','h','i','j','k',
                                  'l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

In [41]: n=model.predict(X_test[1:15])

In [42]: print("Predictions: ")
         for i in n:
             print(resulting_alpha_numeric[i])

         Predictions:
         a
         0
         3
         x
         U
         O
         B
         5
         I
         2
         4
         2
         1
         2

In [43]: m=y_test[1:15]

In [44]: print("Actual results")
         for i in m:
```

FIGURE 6: TESTING AND EVALUATION

Here, I have used a list of alphanumeric to convert numbers of predictions and y_test to their respective digits/alphabets.


jupyter english_alphabet_recognistion_Pratyush Last Checkpoint: 34 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
         2
         4
         2
         1
         2

In [43]: m=y_test[1:15]

In [44]: print("Actual results")
         for i in m:
             print(resulting_alpha_numeric[i])

         Actual results
         a
         0
         3
         X
         U
         O
         e
         5
         8
         2
         4
         2
         1
         2

In [53]: model.score(X_test,y_test)
Out[53]: 0.6751717201241371
```

And we can from both ‘Predictions’ and ‘Actual results’ that, model has done fairly nice. But It sometimes confuses between:

- O(capital o) ,o(small o),0(zero)
- G, C, c
- U, 0, O, u ,etc

It has a score of approximately 70% on test data, which is fairly nice.

REQUIREMENTS:

- Jupyter notebook or collab (PYTHON 3.8+ KERNEL)
- NumPy, matplotlib, mnist, sklearn library packages of python
- EMNIST byclass Dataset.

MY CODE AND DATASET FOLDER LINK:

https://drive.google.com/drive/folders/1e8Fo6lbo7gtZBo5FWwr1XDjluCBjVHWJ?usp=sharing