Topic	**Digit Recognition-1**
Class Description	**Students learn about image detection.** **Students create their own prediction algorithm.**
Class	**C122**
Class time	**45 mins**
Goal	● Learn and create a model which can detect the digits from an image.
Resources Required	● Teacher Resources ○ Colab Notebook ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ● Student Resources ○ Colab Notebook ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen

Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

CONTEXT

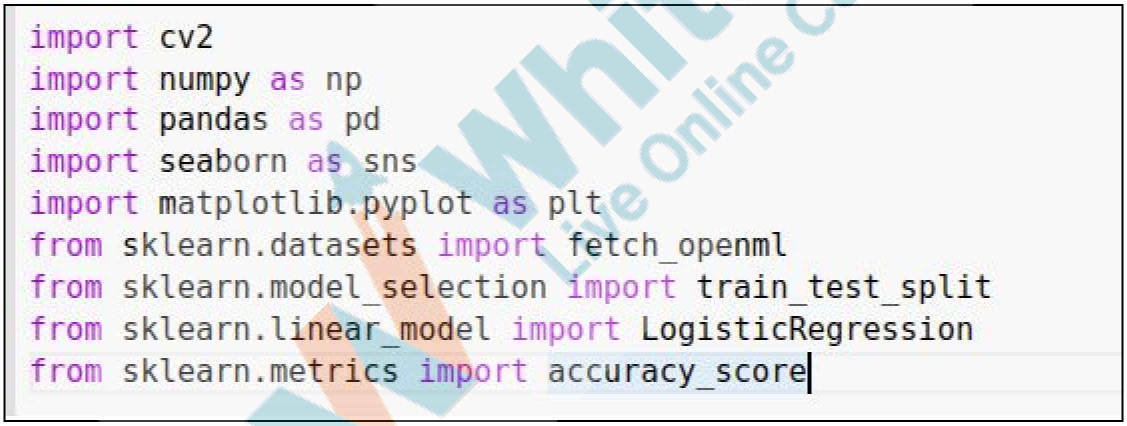
- **Introducing the concept of image detection.**

Class Steps	**Teacher Action**	**Student Action**
Step 1: **Warm Up** **(5 mins)**	Hi <Student Name>! How are you doing today? Can you quickly revise what we did in last class?	*The student revises the concepts covered in the previous class.*

	So we used the camera and created an invisibility cloak. Few classes ago we learned about logistic regression and we saw that it was used to separate the outcomes. We learned about binary logistic regression where the outcome is always in True or False. But what if the outcome is not binary and something else?	**ESR:** varied
	Today we are going to see one such example where the outcome is not binary. In the previous class we got the computer to detect the colors this time we'll get the computer to detect the hand written numbers. Sounds exciting?	**ESR:** Yes
Teacher Initiates Screen Share		
CHALLENGE ● **Use the OpenML library for data** ● **Prepare the data to create the logistic regression**		
Step 2: **Teacher-led Activity** **(15 min)**	*Teacher opens the Google Colab notebook from Teacher activity 1.*	-
	Let's first import all the libraries that we'll need. *Teacher asks the student to guess*	

	what the library is used for as she imports them. *<Teacher codes to import all the libraries required>* Code:- **import cv2** **import numpy as np** **import pandas as pd** **import seaborn as sns** **import matplotlib.pyplot as plt** **from sklearn.datasets import fetch_openml** **from sklearn.model_selection import train_test_split** **from sklearn.linear_model import LogisticRegression** **from sklearn.metrics import accuracy_score** *<Teacher explains what the libraries are and what they are used for>* 1.**import cv2** - This is the library with which we are going to use our computer's camera. 2.**import numpy as np** - This is so that we can perform complex mathematical/list operations. 3.**import pandas as pd** - This is so that we can treat our data as DataFrames. We already know how helpful they are. 4.**import seaborn as sns** - This is a python module to prettify the charts that we draw with matplotlib. We have used it a couple of times. 5.**import matplotlib.pyplot as plt** -	*Student guesses what the library is used for as the teacher imports the library.*

	This library is used to draw the charts. 6.**from sklearn.datasets import fetch_openml** - This function allows us to retrieve a data set by name from OpenML, a public repository for machine learning data and experiments. 7.**from sklearn.model_selection import train_test_split** - This is to split our data into training and testing. 8.**from sklearn.linear_model import LogisticRegression** - This is for creating a Logistic Regression Classifier. 9.**from sklearn.metrics import accuracy_score** - This is to measure the accuracy score of the model.	



```
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

	Here we are going to be using a dataset from OpenML where we have 70,000 images of hand-written digits. *Teacher codes to use the dataset from OpenML library.* Code: **X, y = fetch_openml('mnist_784', version=1, return_X_y=True)**	*Student observes and asks questions.*

```
print(pd.Series(y).value_counts())
classes = ['0', '1', '2','3', '4','5', '6', '7', '8', '9']
nclasses = len(classes)
```

Let's go through this code line by line.

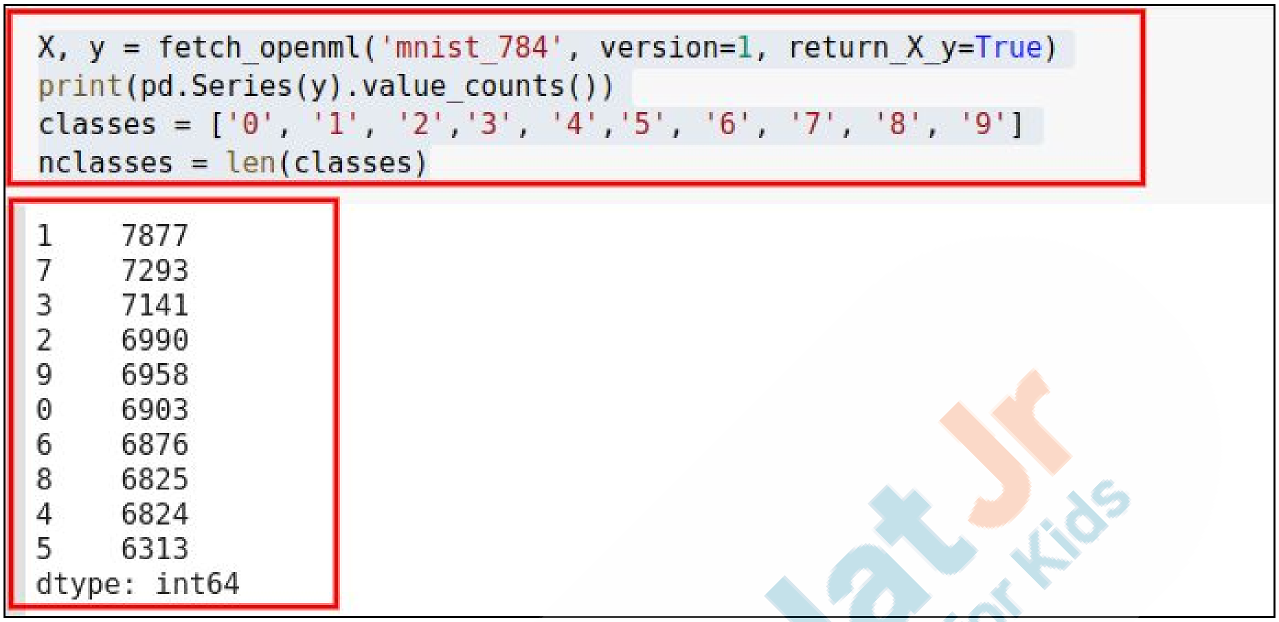
As we know, we can represent any given image in the form of binary. Here in the first line of code -

X, y = fetch_openml('mnist_784', version=1, return_X_y=True)

We are fetching the dataset of images of handwritten digits from the OpenML datasets. The X here would be the data of images represented in Binary, while y would be the label of that image, i.e - 0, 1, 2, ..., 9.

With the second line of code, **print(pd.Series(y).value_counts())** it tells us the count of samples for each of the labels. Here, we can see that label 1 has 7,877 samples, 2 has 6,990 samples and so on.

Finally, we are creating a list that contains all the labels.



```
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
print(pd.Series(y).value_counts())
classes = ['0', '1', '2','3', '4','5', '6', '7', '8', '9']
nclasses = len(classes)
```

```
1    7877
7    7293
3    7141
2    6990
9    6958
0    6903
6    6876
8    6825
4    6824
5    6313
dtype: int64
```

Now lets see some examples of each label.

Code:

```
samples_per_class = 5
figure = plt.figure(figsize=(nclasses*2,(1+samples_per_class*2)))

idx_cls = 0
for cls in classes:
  idxs = np.flatnonzero(y == cls)
  idxs = np.random.choice(idxs, samples_per_class, replace=False)
  i = 0
  for idx in idxs:
    plt_idx = i * nclasses + idx_cls + 1
    p = plt.subplot(samples_per_class, nclasses, plt_idx);
    p =
```

Student observes and asks questions.

```
sns.heatmap(np.reshape(X[idx],
(28,28)), cmap=plt.cm.gray,
             xticklabels=False,
yticklabels=False, cbar=False);
    p = plt.axis('off');
    i += 1
  idx_cls += 1
```

Now let's look at the code step by step.
With the first line of code - We are saying that we want to display 5 samples per label.

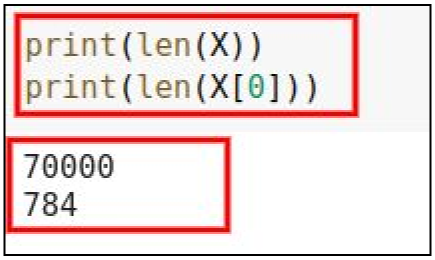
Next, with **figure = plt.figure(figsize=(nclasses*2,(1+samples_per_class*2)))**, we are setting up the total size of the figure that we plotted above.

Now we get into a loop where we are iterating over the classes that we created earlier, which was a list of all the labels (0, 1, 2, ..., 9).

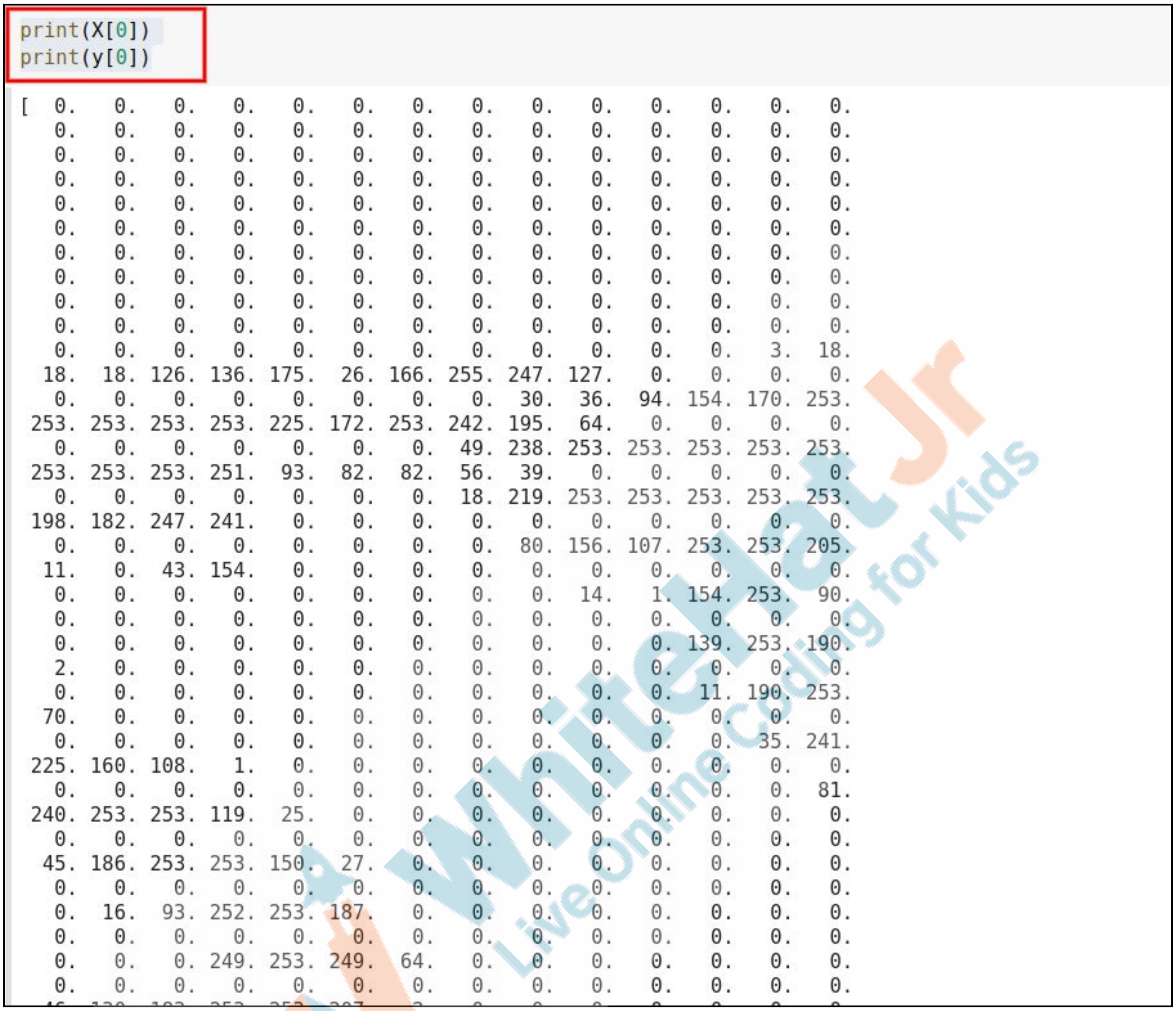
We are first filtering out all the indexes of the elements with value equal to our label with idxs = np.flatnonzero(y == cls) and then we are selecting any 5 random indexes with np.random.choice(idxs, samples_per_class, replace=False).

	We are then iterating over these random indexes for the given label. First, we are doing **plt_idx = i * nclasses + idx_cls + 1** to define the position of the given label. Here, the **idx_cls** acts as the column while i acts as the rows. For all the samples of label 0 that we are plotting, the column idx_cls would remain to be the same while the row i will change. This helps us form a grid of samples while plotting. Now, we have the index of the random sample of the label, but how are we plotting it? We are creating a heatmap to plot the image with. **p = sns.heatmap(np.reshape(X[idx], (28,28)), cmap=plt.cm.gray, xticklabels=False, yticklabels=False, cbar=False);** Here, we are taking the index idx and fetching its element from X, and we are reshaping it in a 28*28 grid. Remember, images here are represented as binary.	
	As we know that the images here are represented by the binary. Let's verify that for X.	

	Code:
	print(len(X))
	print(len(X[0]))
	
	What can we see here?
		ESR:
		We can see that X has 70,000 image data and each image has 784 pixels of data.


`print(len(X))`
`print(len(X[0]))`
`70000`
`784`		

	And that is equivalent to 28*28. Now let's check the array of the particular image. Code: **print(X[0])** **print(y[0])** What do we see here?	**ESR:** We can see, the given label we printed is 5, and we can also make out 5 from the array printed above. The maximum values we have in this array could be 255 (Array consists of numbers from 0 to 255).



```
print(X[0])
print(y[0])
```

```
[   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   3.  18.
   18.  18. 126. 136. 175.  26. 166. 255. 247. 127.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.  30.  36.  94. 154. 170. 253.
  253. 253. 253. 253. 225. 172. 253. 242. 195.  64.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.  49. 238. 253. 253. 253. 253. 253.
  253. 253. 253. 251.  93.  82.  82.  56.  39.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.  18. 219. 253. 253. 253. 253. 253.
  198. 182. 247. 241.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.  80. 156. 107. 253. 253. 205.
   11.   0.  43. 154.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.  14.   1. 154. 253.  90.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0. 139. 253. 190.
    2.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  11. 190. 253.
   70.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  35. 241.
  225. 160. 108.   1.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  81.
  240. 253. 253. 119.  25.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
   45. 186. 253. 253. 150.  27.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.  16.  93. 252. 253. 187.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0. 249. 253. 249.  64.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
```

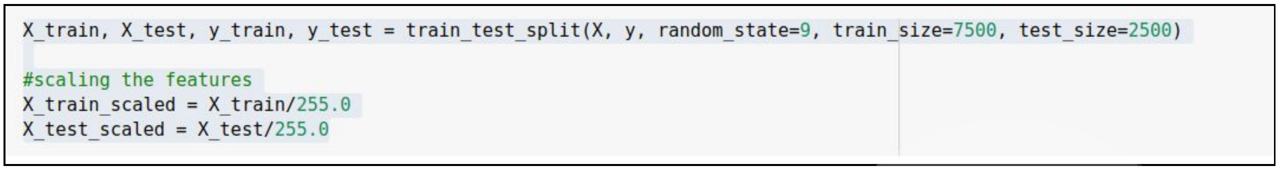
	Alright now we have the data ready, all we need to do is create a logistic regression model and plot the heat map. Can you try doing that and don't worry I'll help you along the way.	**ESR:** yes
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
- **Ask Student to press ESC key to come back to panel**		

- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

ACTIVITY

- **Use the generated data to create a prediction model**
- **Draw the confusion matrix for the data.**

Step 3: **Student-Led Activity** **(15 min)**	*Teacher helps the student to open a New Colab notebook and to update the code till the point taught by the teacher.*	*Student open the new colab notebook using the Student Activity and updates the code till the point taught by the teacher.*
	Now we'll prepare the data by splitting it and train a Logistic Regression Model. We will have to divide it by 255 so that all the features of an image could be represented with values between 0 and 1. We'll only be using 10000 samples as computing 70000 examples will take a lot of computing time. We will have to split our data into 7500 for training and 2500 for testing. Code: **X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=9, train_size=7500, test_size=2500)** **#scaling the features** **X_train_scaled = X_train/255.0** **X_test_scaled = X_test/255.0**	*Student codes to use 1000 samples and split data 7500 for training and 2500 for testing.*

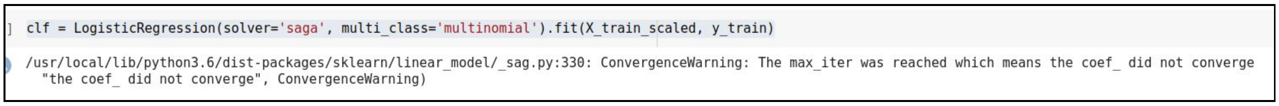


```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=9, train_size=7500, test_size=2500)

#scaling the features
X_train_scaled = X_train/255.0
X_test_scaled = X_test/255.0
```

	Now let's code to fit the data to the model. Code: **clf = LogisticRegression(solver='saga', multi_class='multinomial').fit(X_train_scaled, y_train)** Until now, we were dealing with binary logistic regressions, but here, we have 10 labels, 0 to 9. For this, we write multi_class='multinomial' to specify that this is a multinomial logistic regression. Generally, there is a solver involved in all the logistic regressions, and the default solver is liblinear, which is highly efficient for linear logistic regression. This is also efficient with binary logistic regressions that we learned earlier. For multinomial logistic regression, solver='saga' is highly efficient. It works well with a large number of samples and	*Student codes to fit the data to the model.*

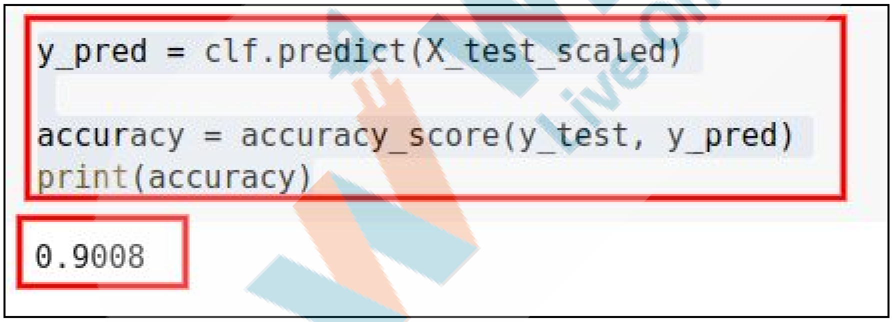
	supports multinomial logistic regressions, like this one.	



```
clf = LogisticRegression(solver='saga', multi_class='multinomial').fit(X_train_scaled, y_train)

/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_sag.py:330: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  "the coef_ did not converge", ConvergenceWarning)
```

	Now that our model is built let's test its accuracy. Code: **y_pred = clf.predict(X_test_scaled)** **accuracy = accuracy_score(y_test, y_pred)** **print(accuracy)** What accuracy do you see?	*Student codes to find the accuracy of the model.* **ESR:** We have an accuracy score of 90.08%.



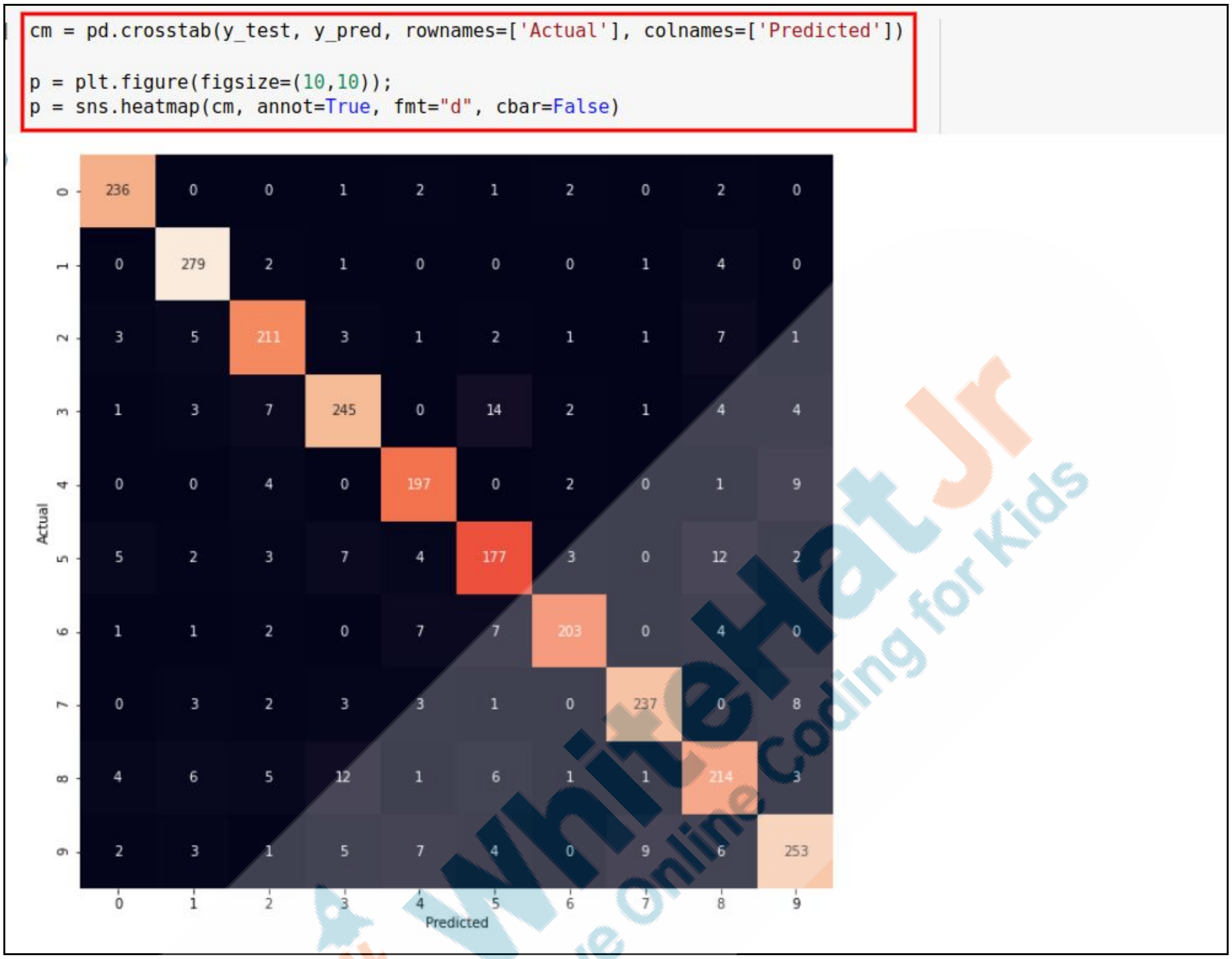
```
y_pred = clf.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

0.9008
```

	This looks like an efficient model. Now let's see how the confusion matrix will look like. Code: **cm = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])**	*Student codes to create the heat map.*

	p = plt.figure(figsize=(10,10)); **p = sns.heatmap(cm, annot=True, fmt="d", cbar=False)** Here, we are creating an array with **cm = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])** and setting its size to be 10*10. We are finally plotting the graph with sns, which is used to prettify the charts that we draw with matplotlib.	



```
cm = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])

p = plt.figure(figsize=(10,10));
p = sns.heatmap(cm, annot=True, fmt="d", cbar=False)
```

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	236	0	0	1	2	1	2	0	2	0
1	0	279	2	1	0	0	0	1	4	0
2	3	5	211	3	1	2	1	1	7	1
3	1	3	7	245	0	14	2	1	4	4
4	0	0	4	0	197	0	2	0	1	9
5	5	2	3	7	4	177	3	0	12	2
6	1	1	2	0	7	7	203	0	4	0
7	0	3	2	3	3	1	0	237	0	8
8	4	6	5	12	1	6	1	1	214	3
9	2	3	1	5	7	4	0	9	6	253

Teacher Guides Student to Stop Screen Share

FEEDBACK

- **Appreciate the student for their efforts**
- **Identify 2 strengths and 1 area of progress for the student**

Step 4: **Wrap-Up** **(5 min)**	Let's quickly revise what we did today.	*Student revises the topic covered in the class.*
	Now we have just predicted the numbers from the image, in the next class we'll see how we can do it using the camera of the system.	**ESR:** Yes

	Sounds exciting? See you in next class.	
Teacher Clicks ✖ End Class		
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? - Describe what happened - Code I wrote ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflection in a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Colab notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true
Teacher Activity 2	Teacher reference code	https://colab.research.google.com/drive/1YR1C5OxuKuxjZ6gR_psvpn09tI2HbwDQ?usp=sharing
Student Activity 1	Colab notebook	https://colab.research.google.com/n

		otebooks/intro.ipynb#recent=true