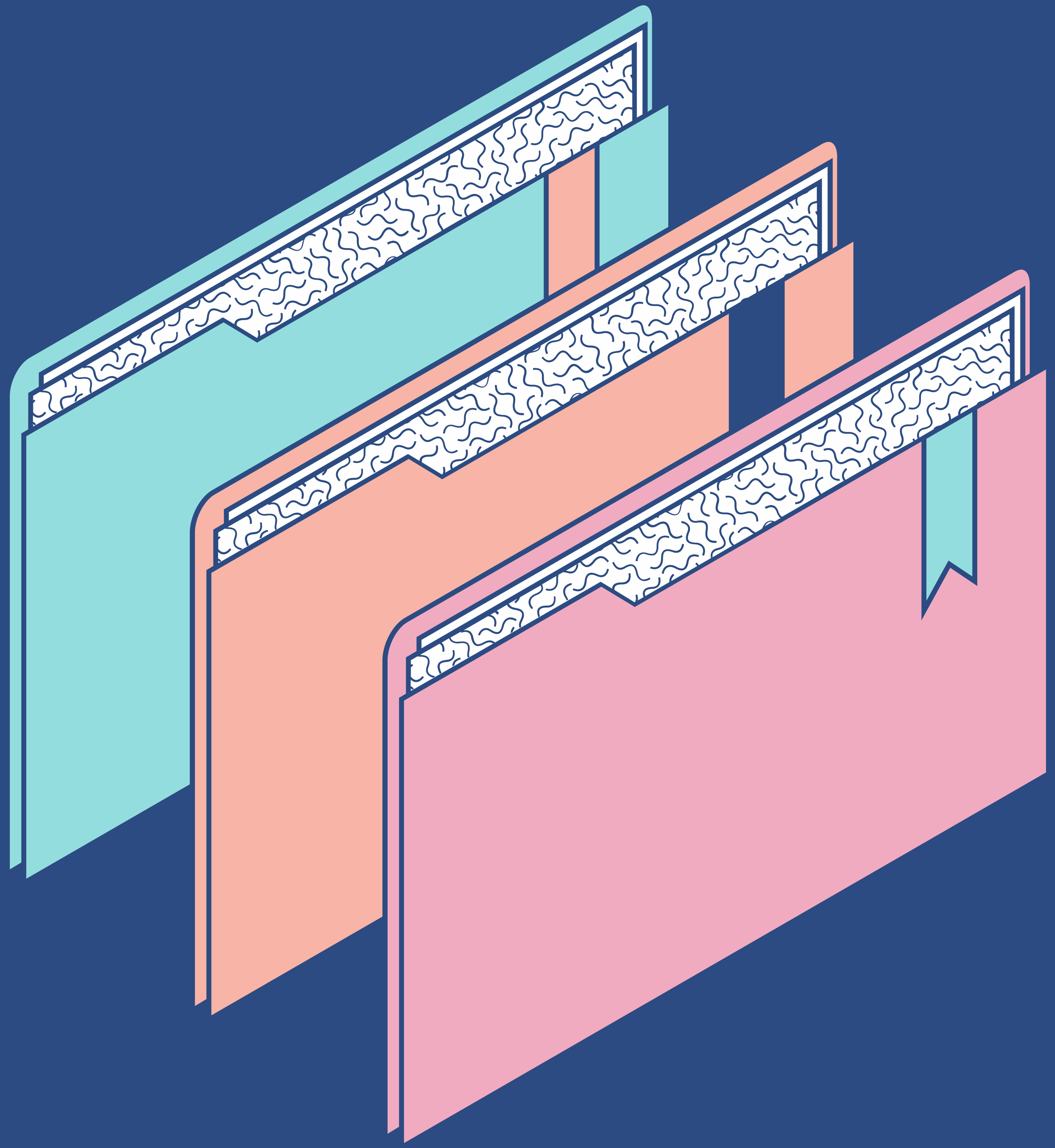
FEMİLDA JOSEPHİN JOSEPH SHOBANA BAI

ENS305 MACHINE LEARNING ASSIGNMENT - 2

ZEYNEP ÖZIŞIL 180722023 COE

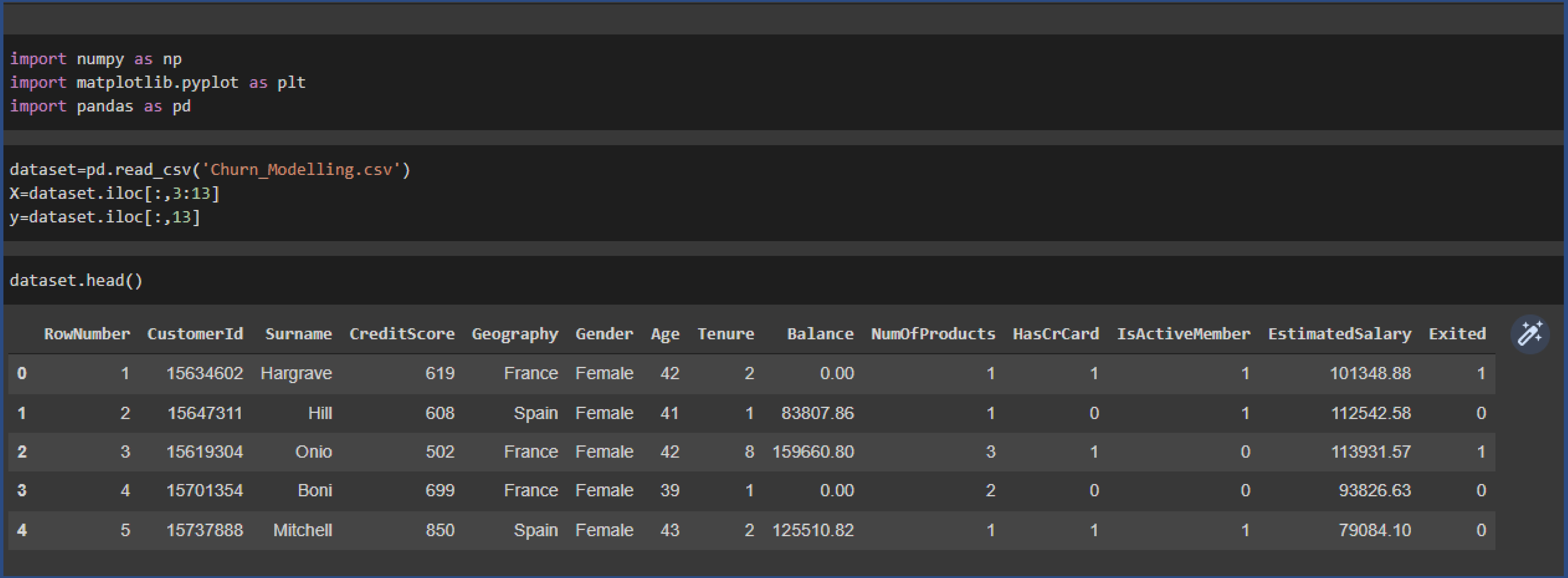


1.

A) FOR THE CHURN MODELLİNG DATASET SHARED İN THE GOOGLE DRİVE LİNK SHARED BELOW USE ARTİFİCİAL NEURAL NETWORKS TO PERFORM CLASSİFİCATİON AND SHOW İTS ACCURACY. B) DRAW THE NEURAL NETWORK ARCHİTECTURE YOU HAVE USED.

Data Preprocessing

Importing the libraries

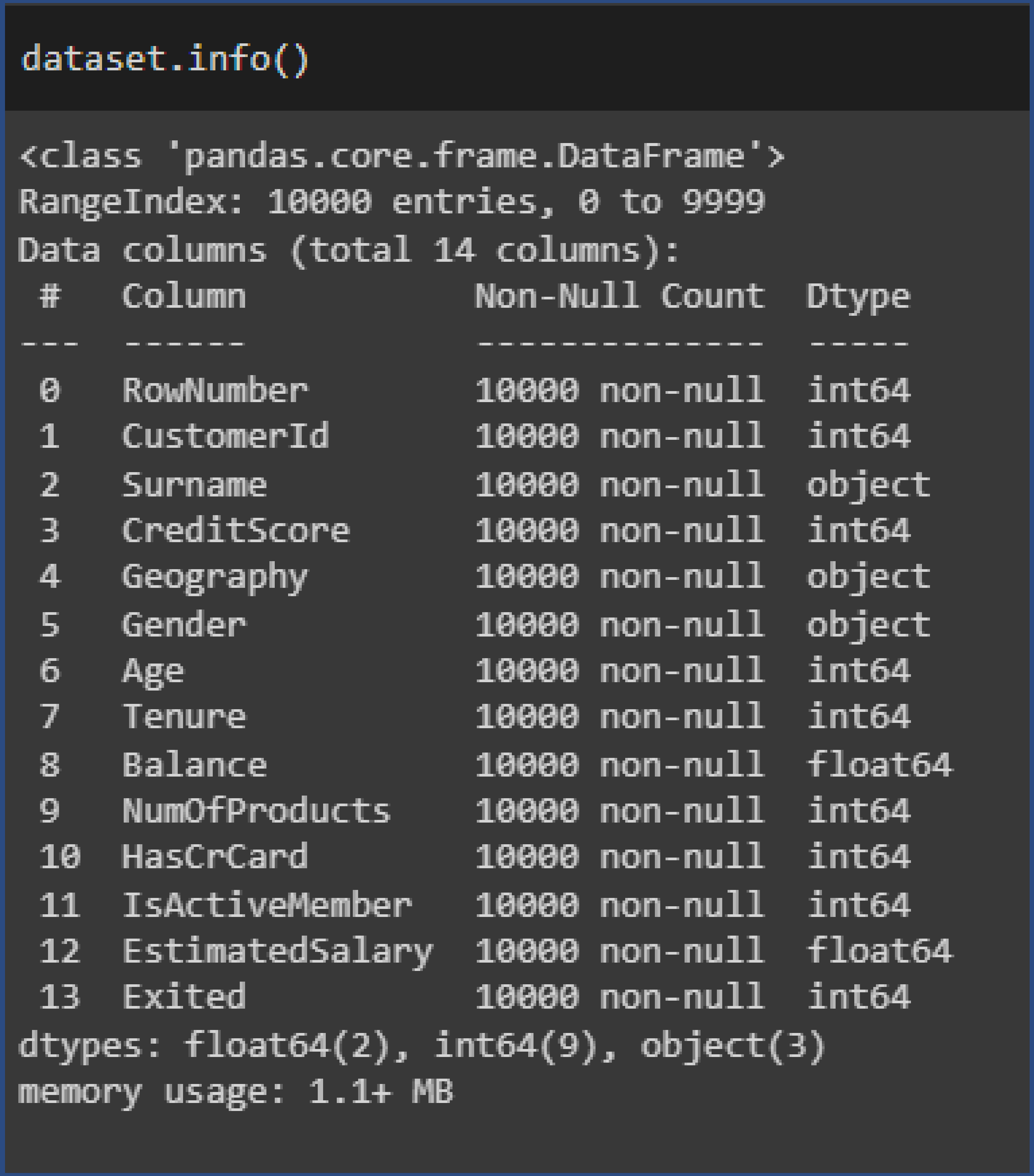


```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```python
dataset=pd.read_csv('Churn_Modelling.csv')
X=dataset.iloc[:,3:13]
y=dataset.iloc[:,13]
```

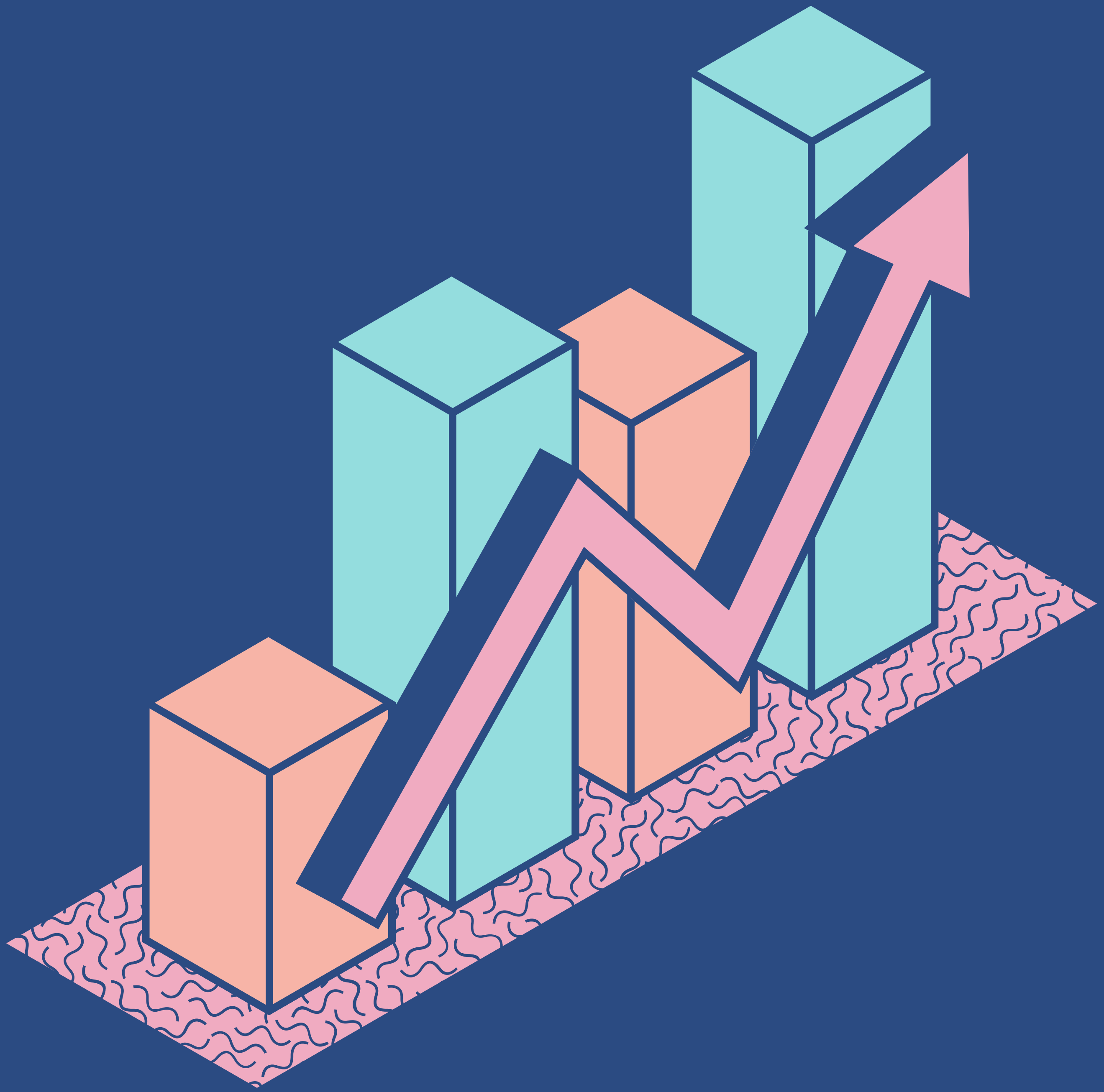
```python
dataset.head()
```

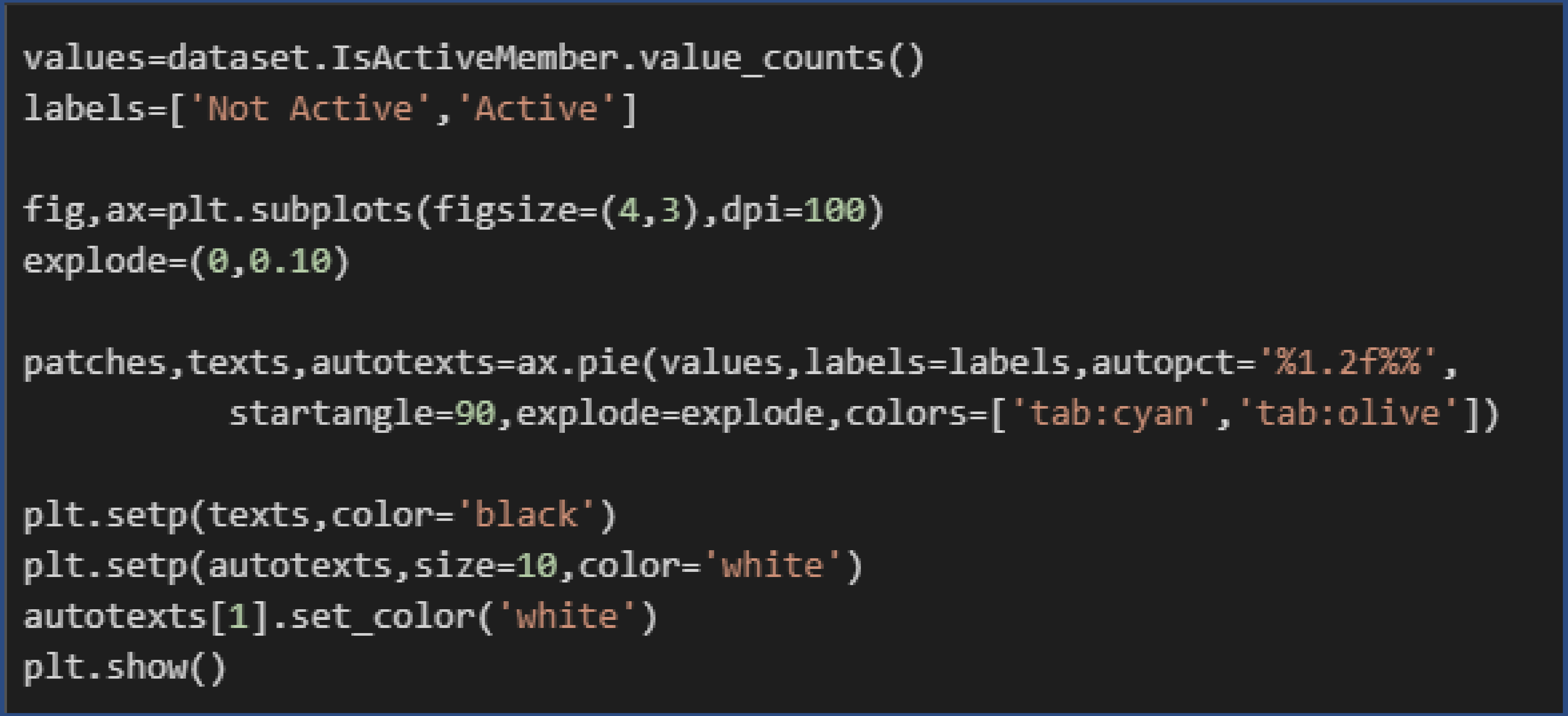
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0



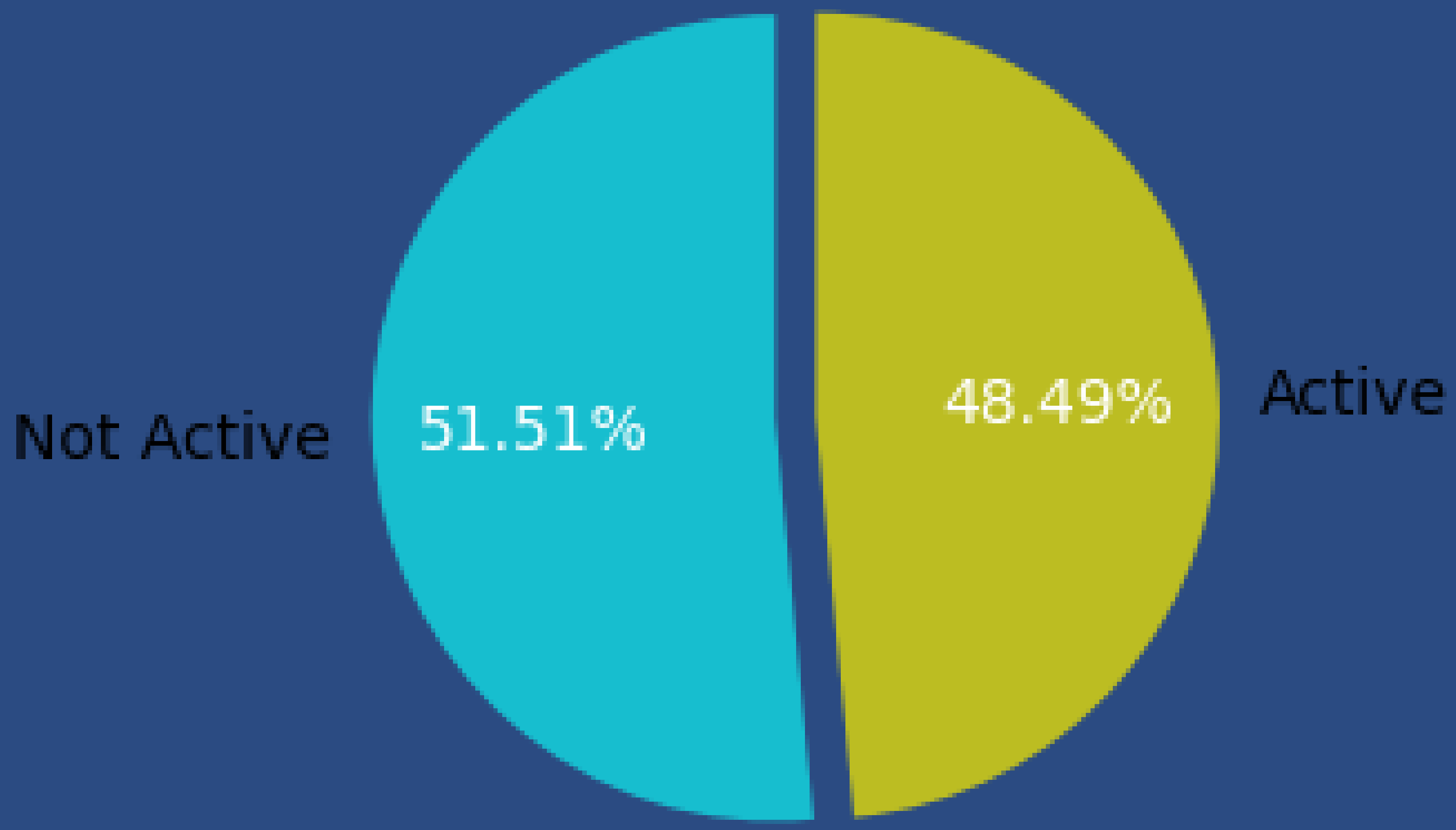
```
dataset.info()
```

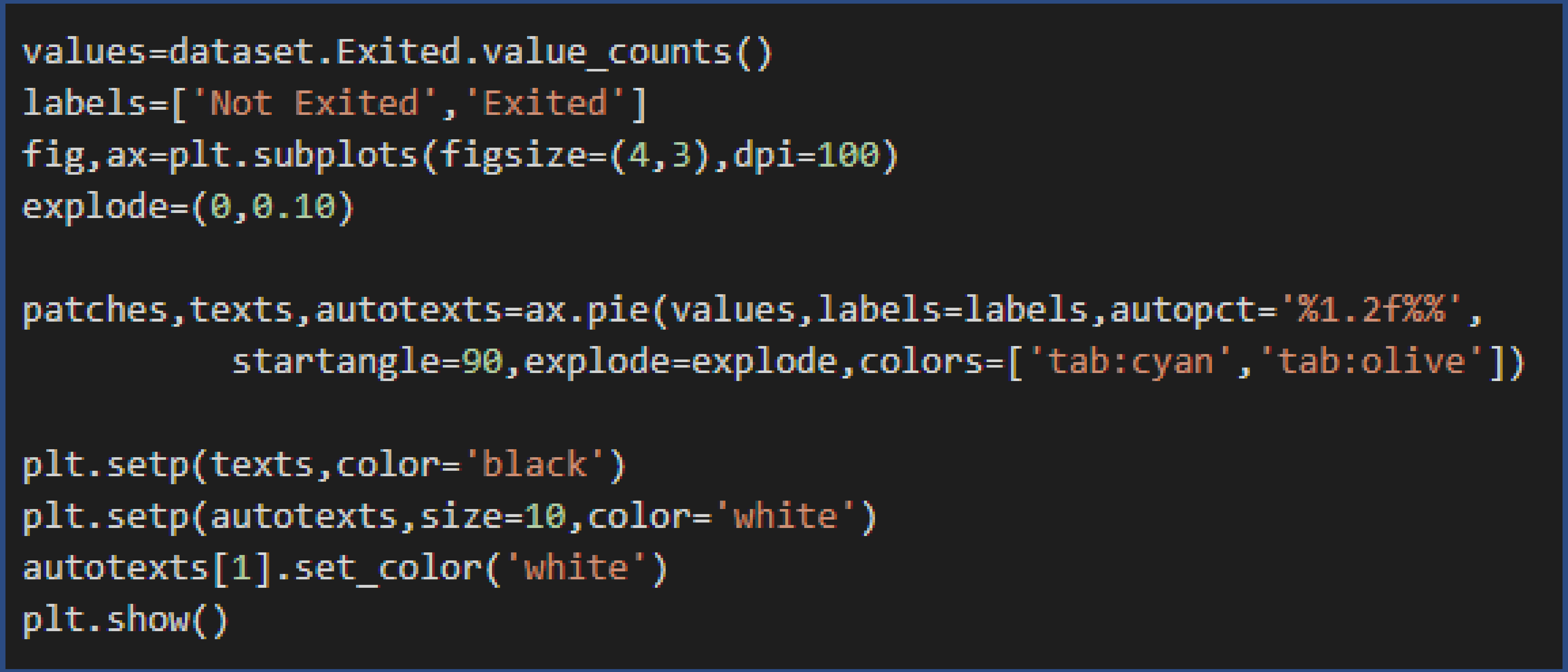
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```



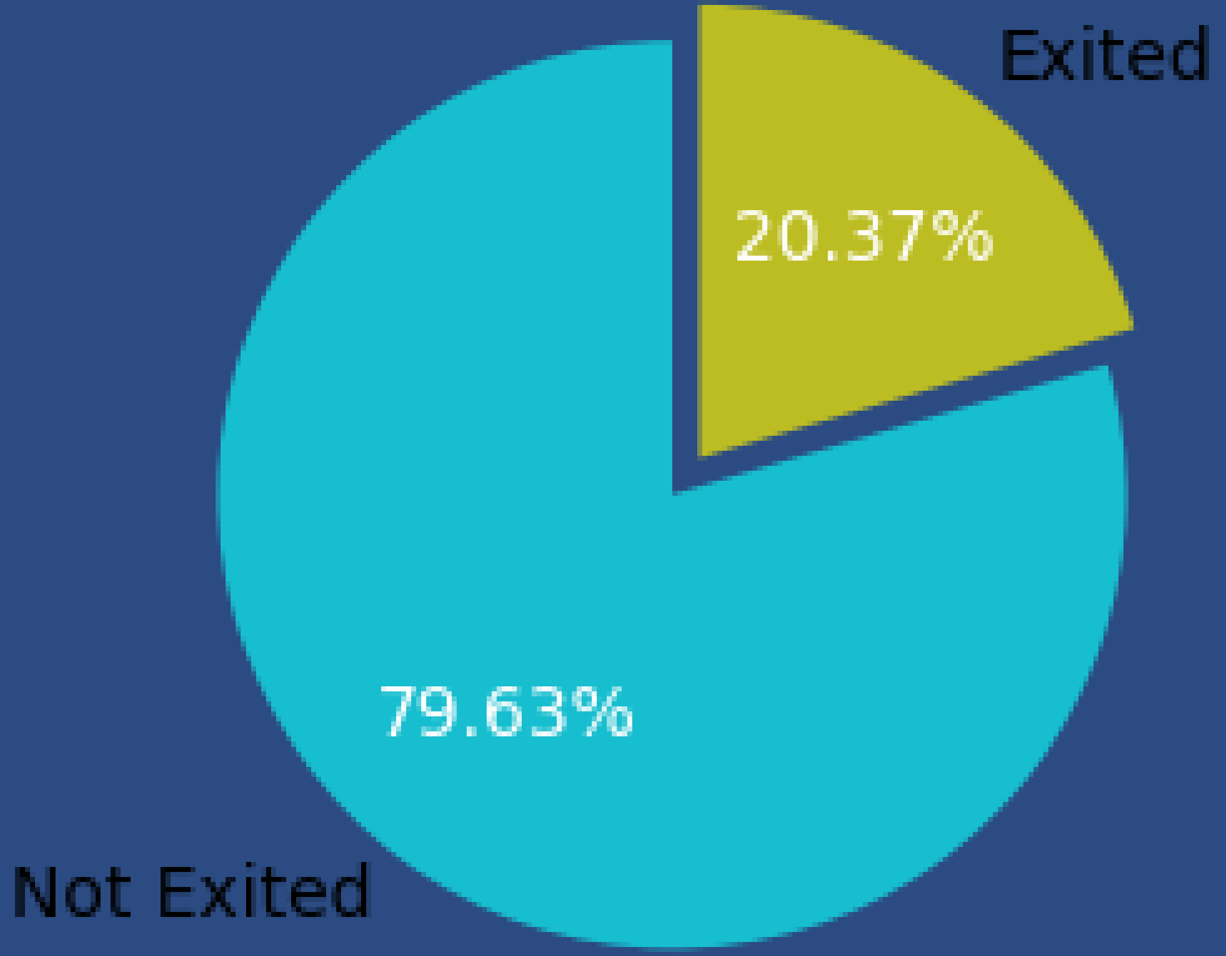


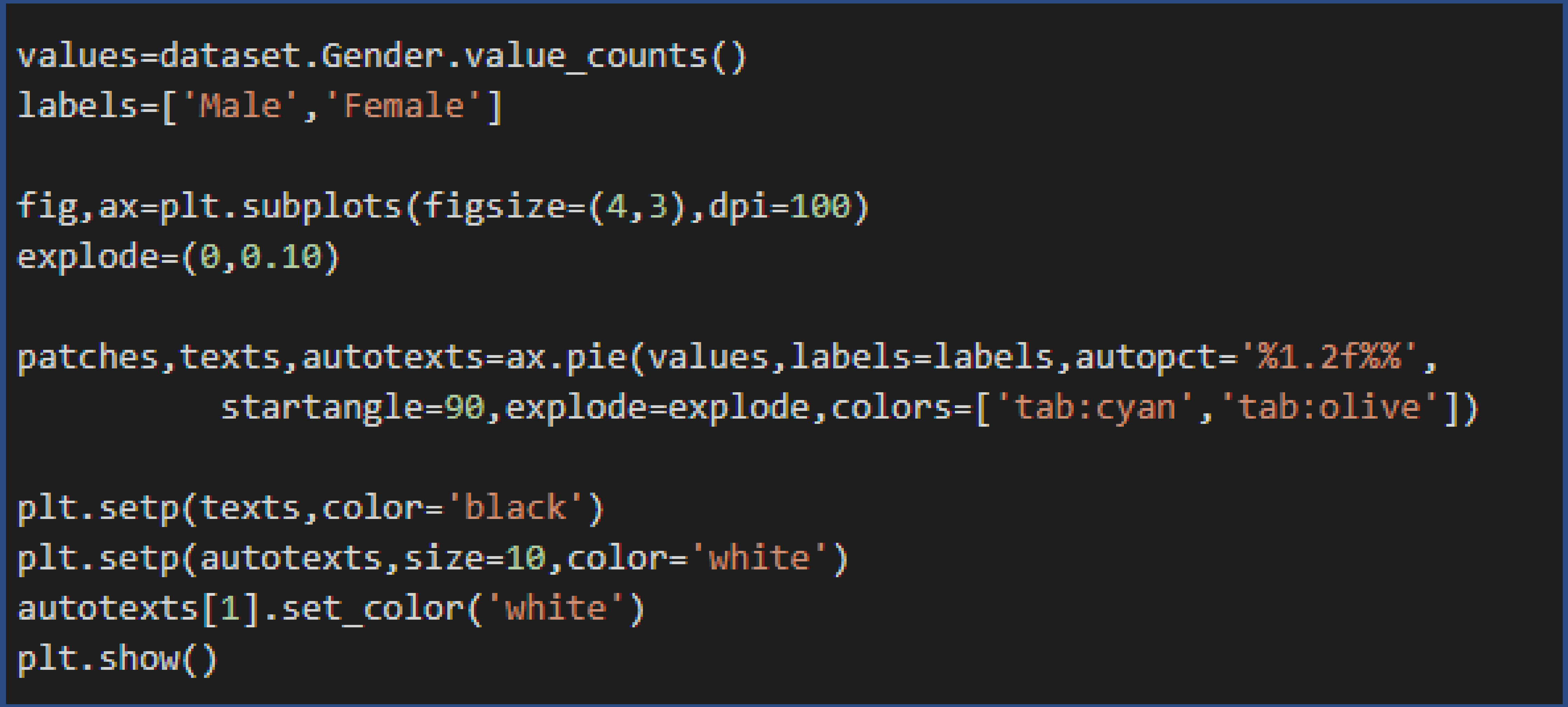
```
values=dataset.IsActiveMember.value_counts()
labels=['Not Active','Active']

fig,ax=plt.subplots(figsize=(4,3),dpi=100)
explode=(0,0.10)

patches,texts,autotexts=ax.pie(values,labels=labels,autopct='%1.2f%%',
          startangle=90,explode=explode,colors=['tab:cyan','tab:olive'])

plt.setp(texts,color='black')
plt.setp(autotexts,size=10,color='white')
autotexts[1].set_color('white')
plt.show()
```



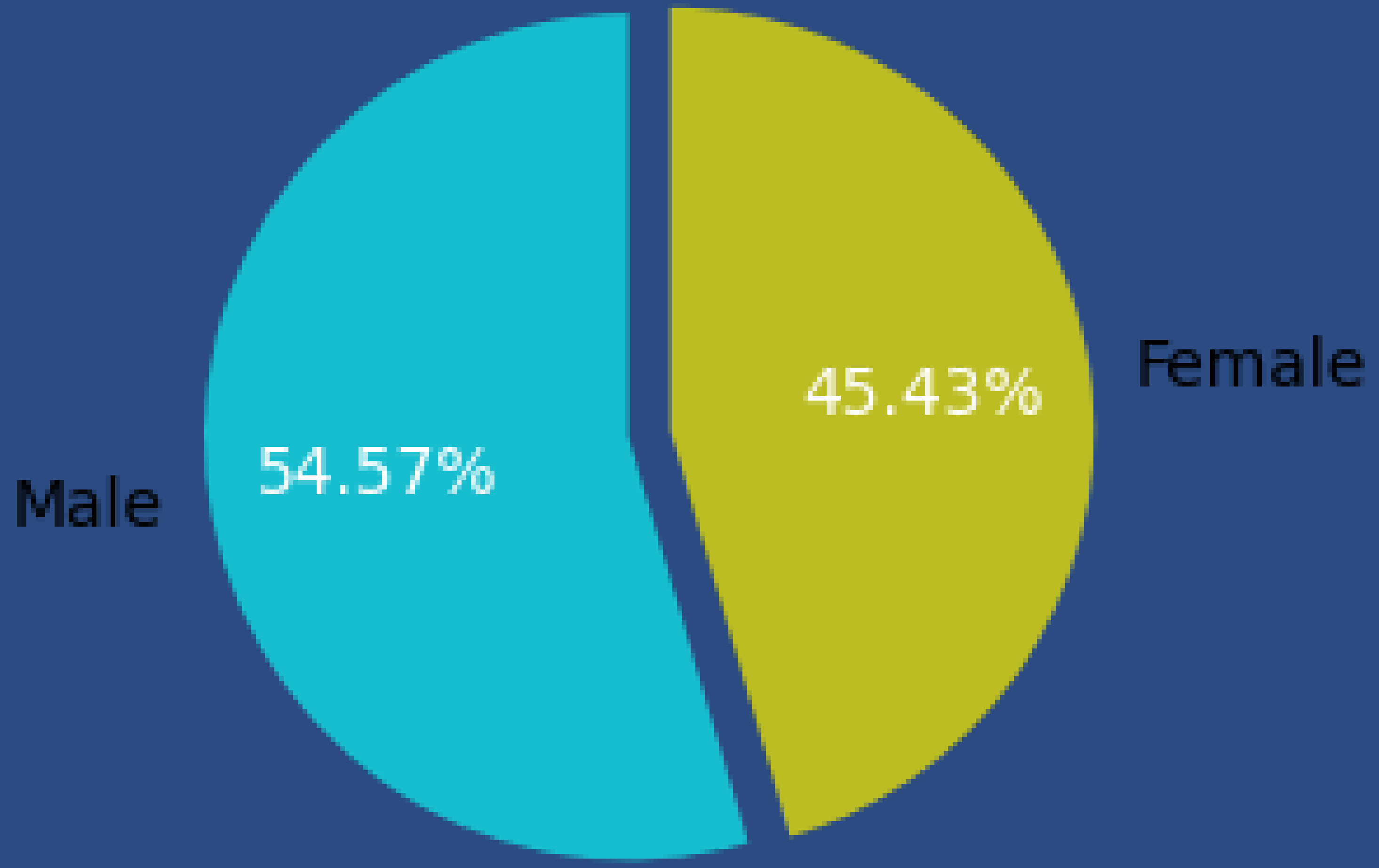


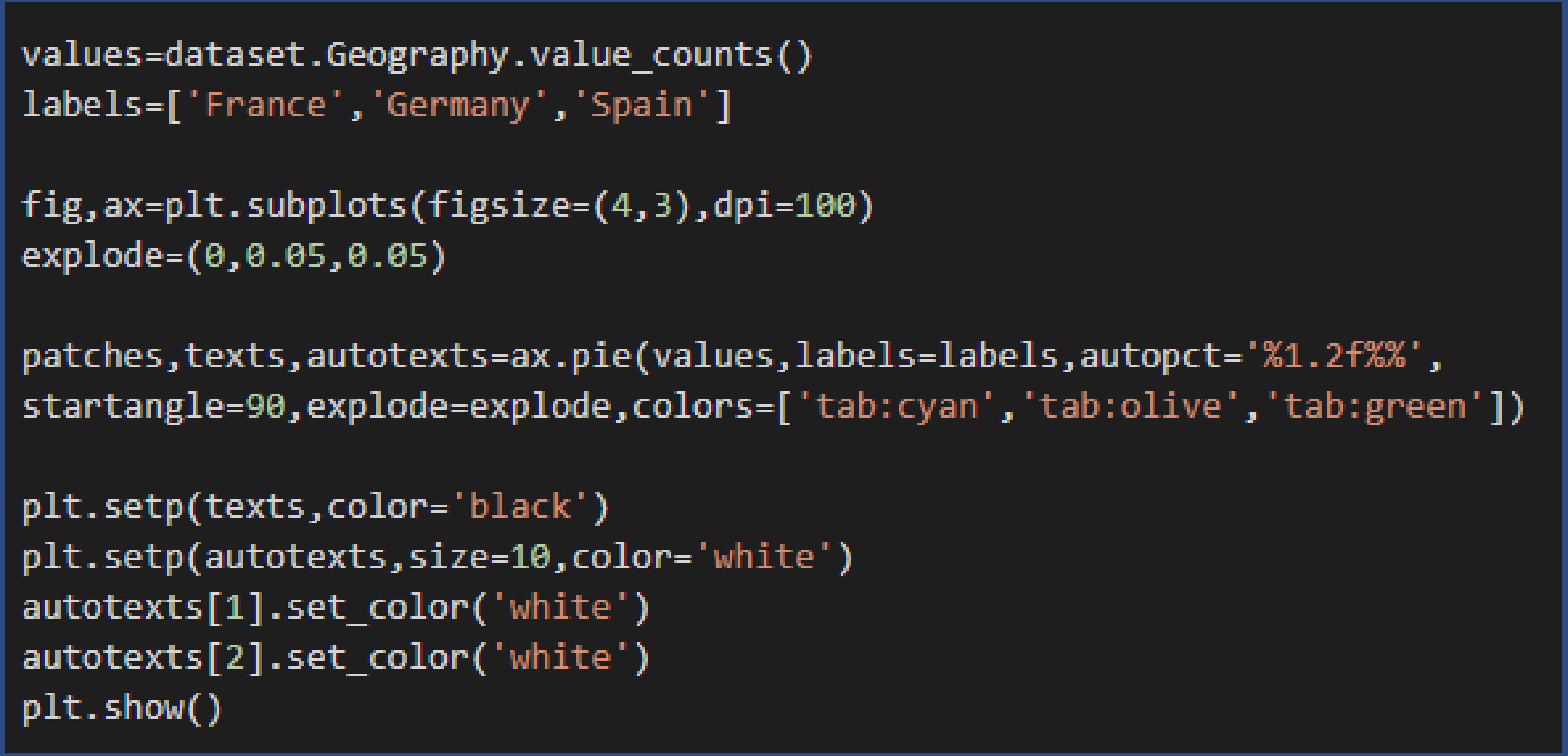
```
values=dataset.Exited.value_counts()
labels=['Not Exited','Exited']
fig,ax=plt.subplots(figsize=(4,3),dpi=100)
explode=(0,0.10)

patches,texts,autotexts=ax.pie(values,labels=labels,autopct='%1.2f%%',
          startangle=90,explode=explode,colors=['tab:cyan','tab:olive'])

plt.setp(texts,color='black')
plt.setp(autotexts,size=10,color='white')
autotexts[1].set_color('white')
plt.show()
```



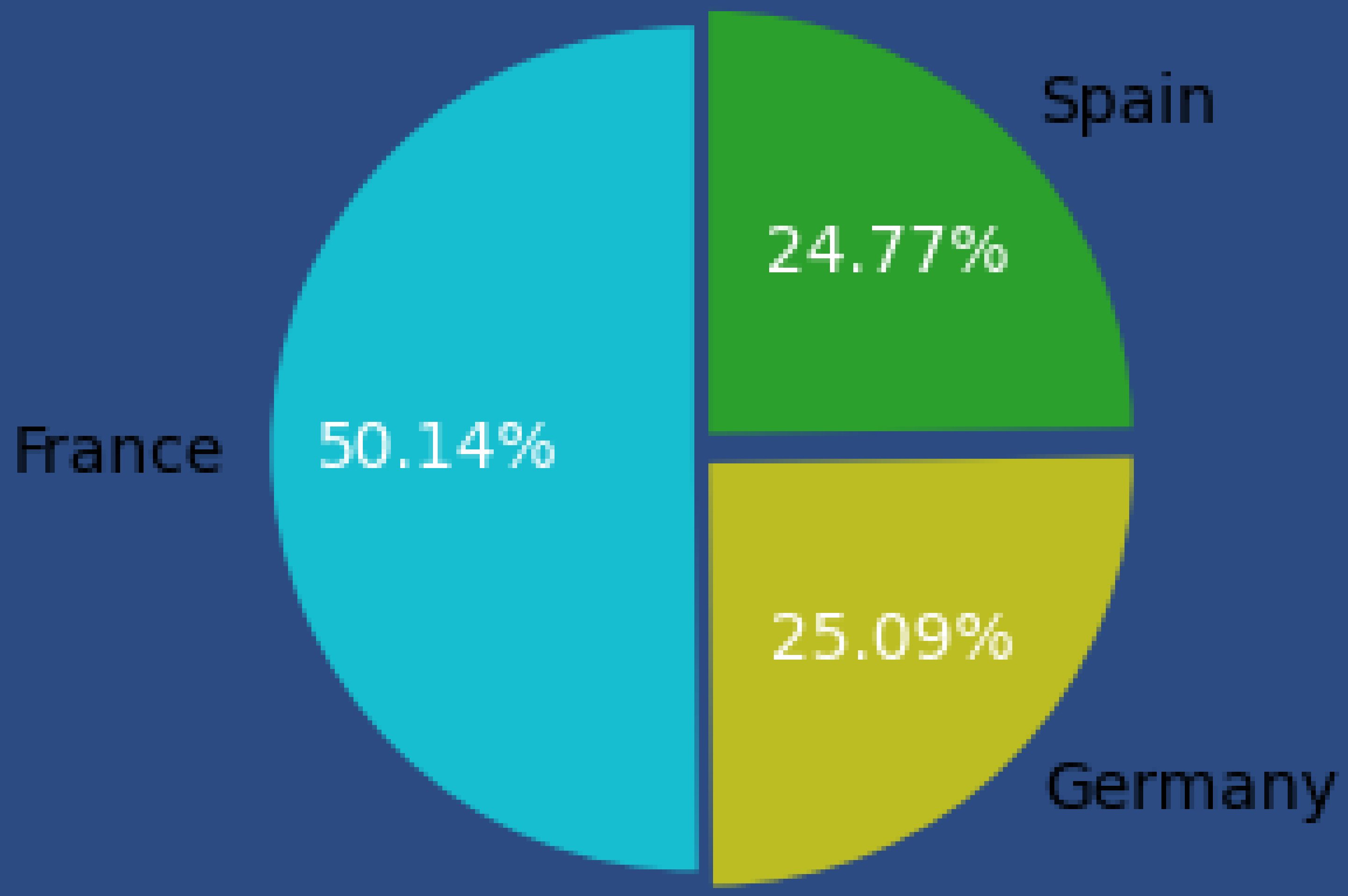


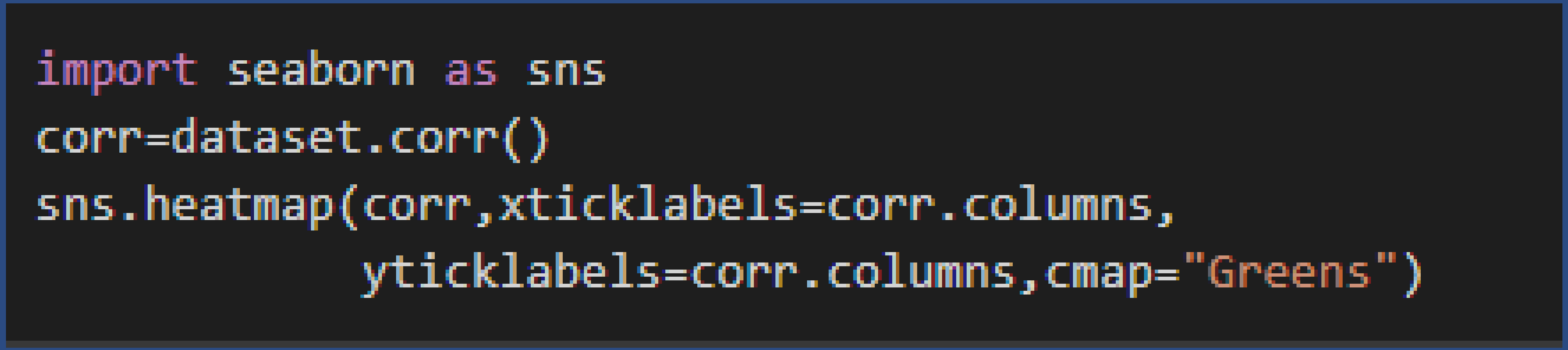
```
values=dataset.Gender.value_counts()
labels=['Male','Female']

fig,ax=plt.subplots(figsize=(4,3),dpi=100)
explode=(0,0.10)

patches,texts,autotexts=ax.pie(values,labels=labels,autopct='%1.2f%%',
          startangle=90,explode=explode,colors=['tab:cyan','tab:olive'])

plt.setp(texts,color='black')
plt.setp(autotexts,size=10,color='white')
autotexts[1].set_color('white')
plt.show()
```



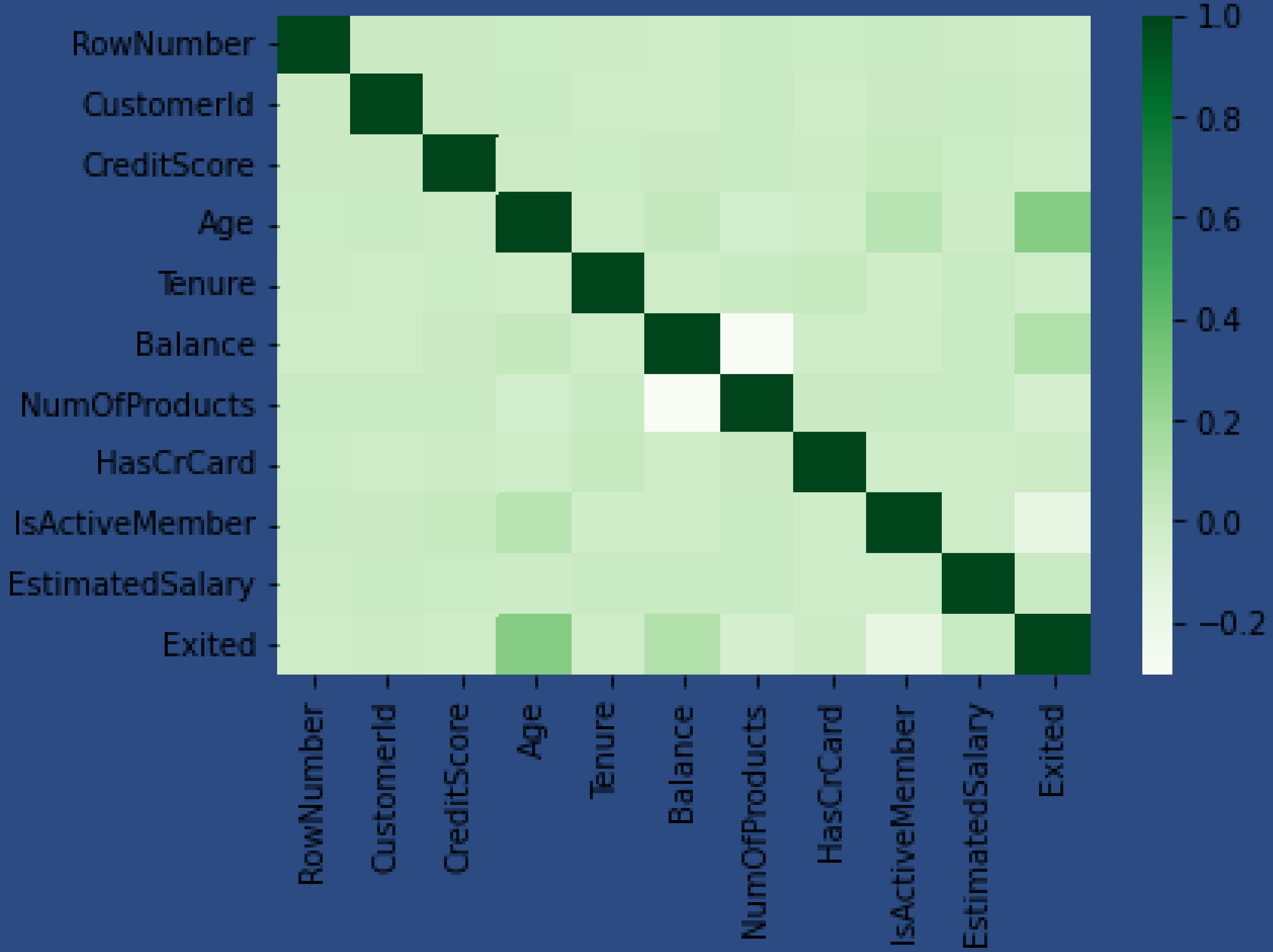


```
values=dataset.Geography.value_counts()
labels=['France','Germany','Spain']

fig,ax=plt.subplots(figsize=(4,3),dpi=100)
explode=(0,0.05,0.05)

patches,texts,autotexts=ax.pie(values,labels=labels,autopct='%1.2f%%',
startangle=90,explode=explode,colors=['tab:cyan','tab:olive','tab:green'])

plt.setp(texts,color='black')
plt.setp(autotexts,size=10,color='white')
autotexts[1].set_color('white')
autotexts[2].set_color('white')
plt.show()
```



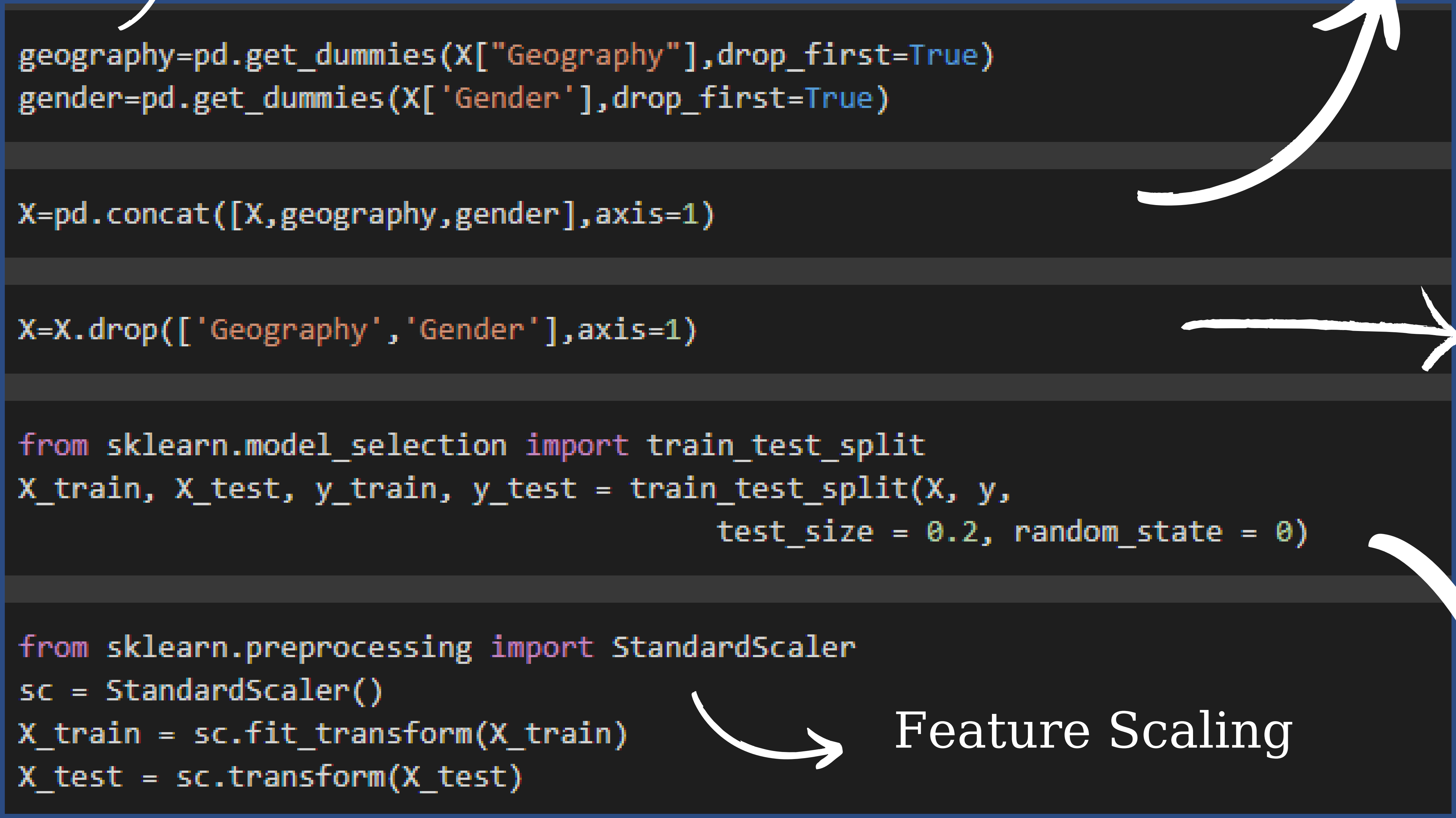


```
import seaborn as sns
corr=dataset.corr()
sns.heatmap(corr,xticklabels=corr.columns,
            yticklabels=corr.columns,cmap="Greens")
```



Create dummy variables

Concatenate the Data Frames



```
geography=pd.get_dummies(X["Geography"],drop_first=True)
gender=pd.get_dummies(X['Gender'],drop_first=True)
```

```
X=pd.concat([X,geography,gender],axis=1)
```

```
X=X.drop(['Geography','Gender'],axis=1)
```

```
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, random_state = 0)
```

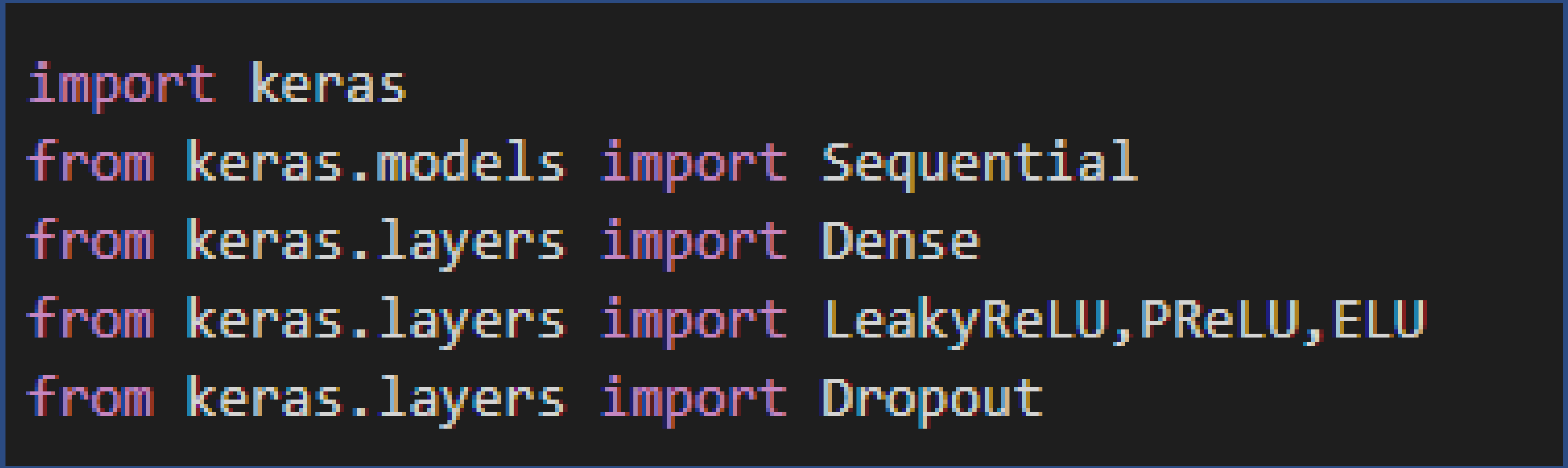
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Drop Unnecessary columns

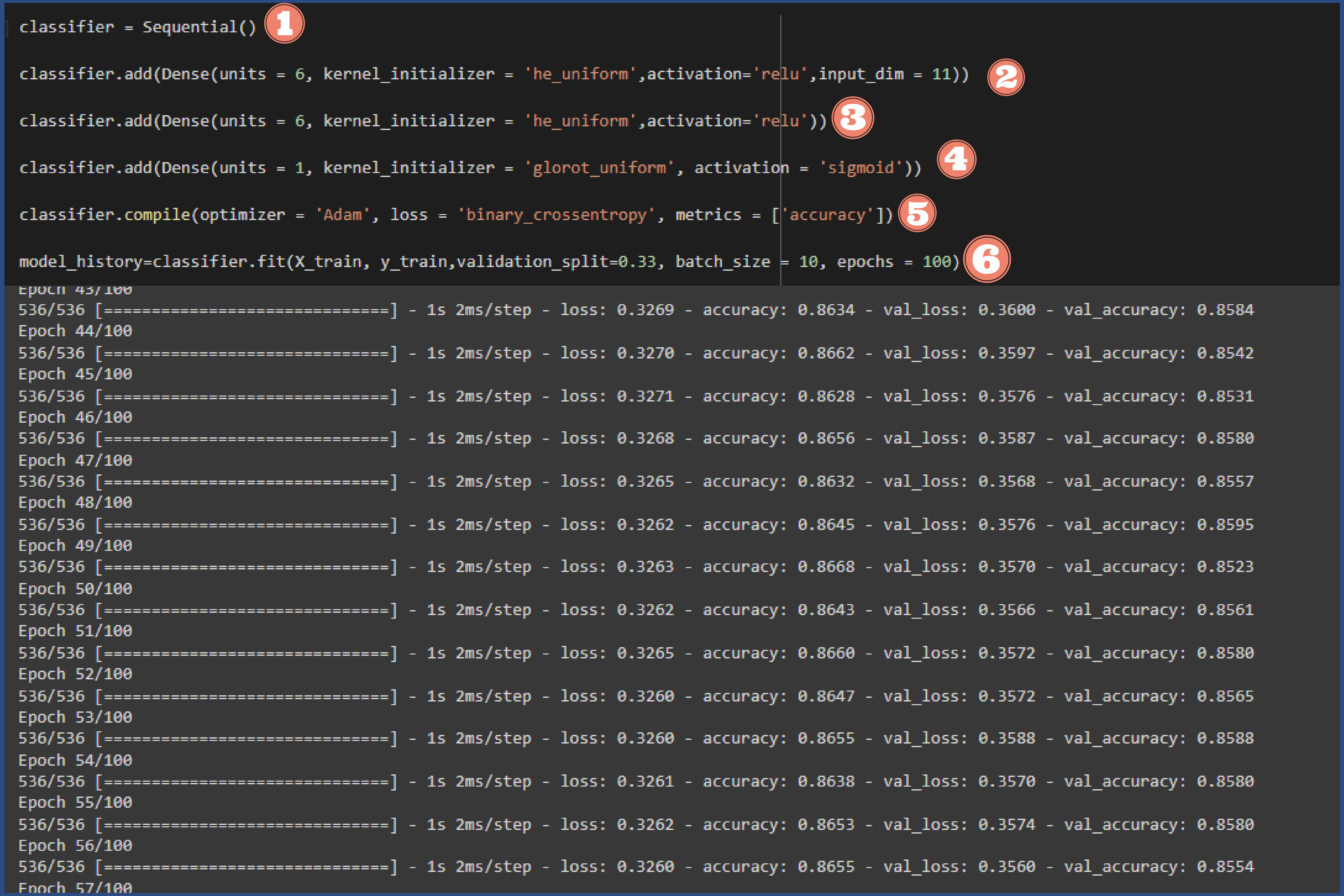
Feature Scaling

Splitting the dataset into the Training set and Test set

Importing the Keras libraries and packages



```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LeakyReLU,PReLU,ELU
from keras.layers import Dropout
```



```
classifier = Sequential()

classifier.add(Dense(units = 6, kernel_initializer = 'he_uniform',activation='relu',input_dim = 11))

classifier.add(Dense(units = 6, kernel_initializer = 'he_uniform',activation='relu'))

classifier.add(Dense(units = 1, kernel_initializer = 'glorot_uniform', activation = 'sigmoid'))

classifier.compile(optimizer = 'Adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

model_history=classifier.fit(X_train, y_train,validation_split=0.33, batch_size = 10, epochs = 100)
```

```
Epoch 43/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3269 - accuracy: 0.8634 - val_loss: 0.3600 - val_accuracy: 0.8584
Epoch 44/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3270 - accuracy: 0.8662 - val_loss: 0.3597 - val_accuracy: 0.8542
Epoch 45/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3271 - accuracy: 0.8628 - val_loss: 0.3576 - val_accuracy: 0.8531
Epoch 46/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3268 - accuracy: 0.8656 - val_loss: 0.3587 - val_accuracy: 0.8580
Epoch 47/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3265 - accuracy: 0.8632 - val_loss: 0.3568 - val_accuracy: 0.8557
Epoch 48/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3262 - accuracy: 0.8645 - val_loss: 0.3576 - val_accuracy: 0.8595
Epoch 49/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3263 - accuracy: 0.8668 - val_loss: 0.3570 - val_accuracy: 0.8523
Epoch 50/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3262 - accuracy: 0.8643 - val_loss: 0.3566 - val_accuracy: 0.8561
Epoch 51/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3265 - accuracy: 0.8660 - val_loss: 0.3572 - val_accuracy: 0.8580
Epoch 52/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3260 - accuracy: 0.8647 - val_loss: 0.3572 - val_accuracy: 0.8565
Epoch 53/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3260 - accuracy: 0.8655 - val_loss: 0.3588 - val_accuracy: 0.8588
Epoch 54/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3261 - accuracy: 0.8638 - val_loss: 0.3570 - val_accuracy: 0.8580
Epoch 55/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3262 - accuracy: 0.8653 - val_loss: 0.3574 - val_accuracy: 0.8580
Epoch 56/100
536/536 [==============================] - 1s 2ms/step - loss: 0.3260 - accuracy: 0.8655 - val_loss: 0.3560 - val_accuracy: 0.8554
Epoch 57/100
```

1)Initialising the ANN

2)Adding the input layer and the first hidden layer

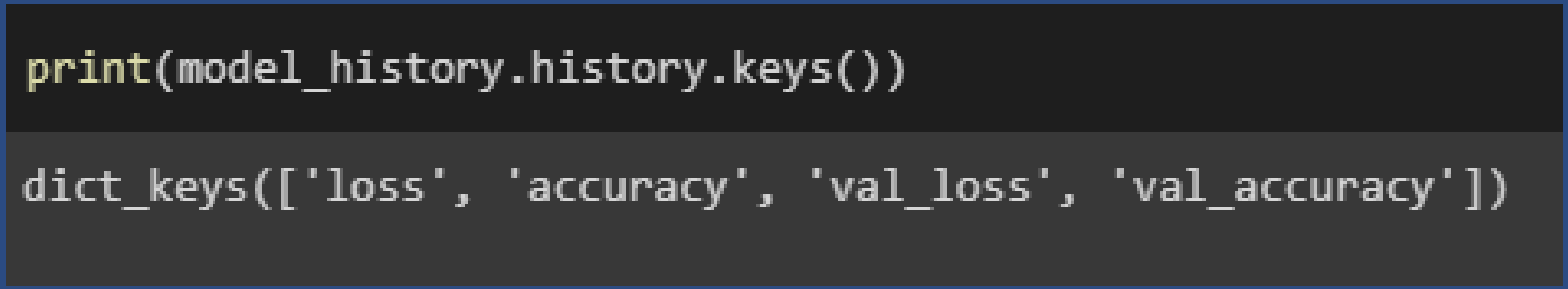
3)Adding the second hidden layer

4)Adding the output layer

5)Compiling the ANN

6)Fitting the ANN to the Training set

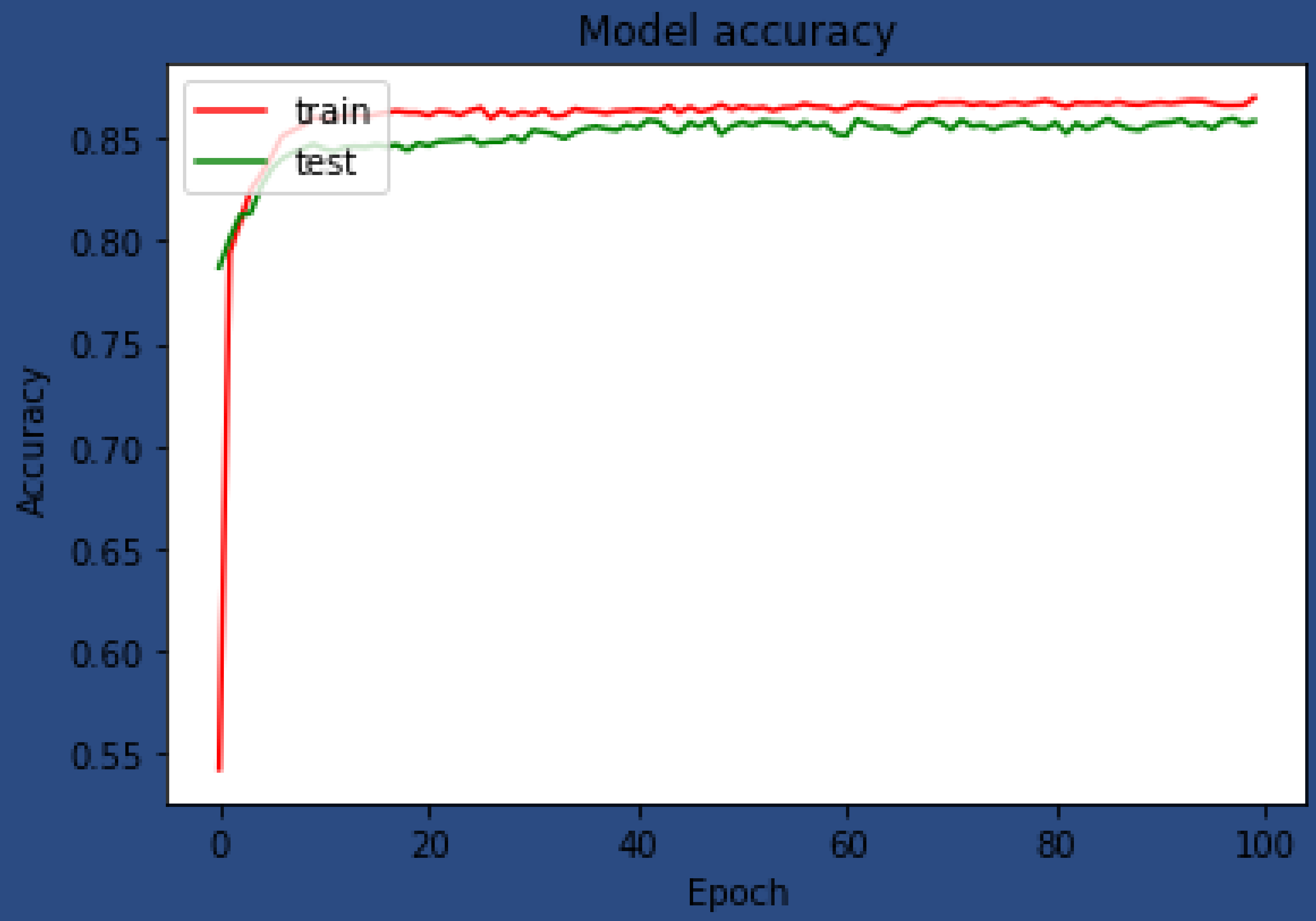
List all data in history

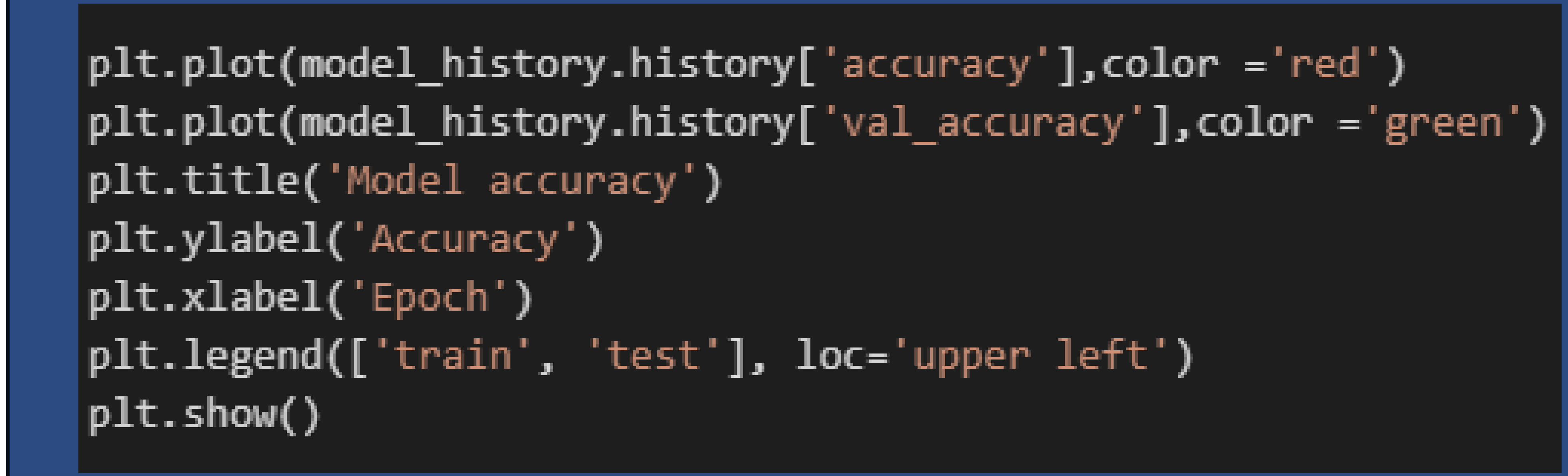


```
print(model_history.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

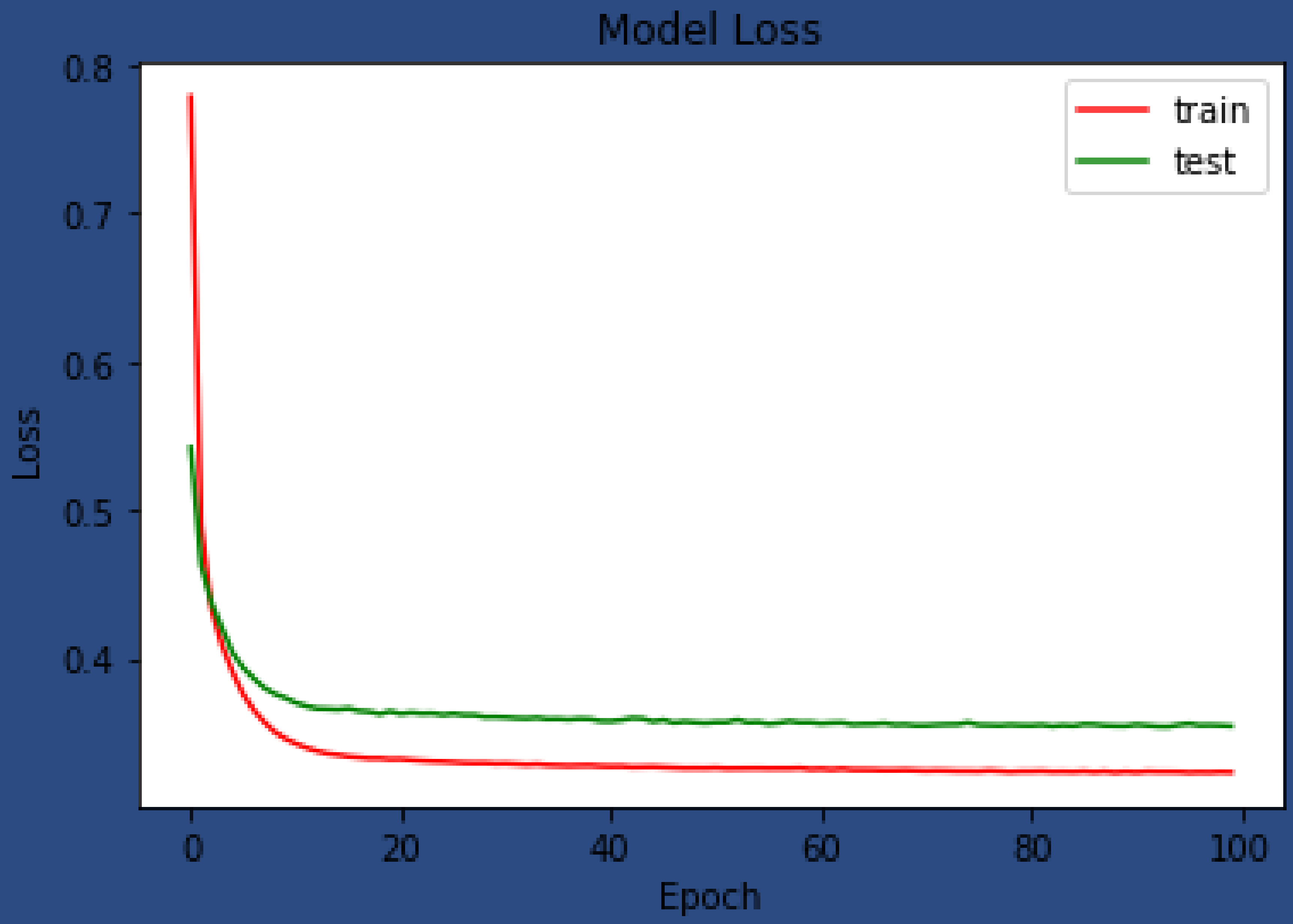
Summarize history for accuracy

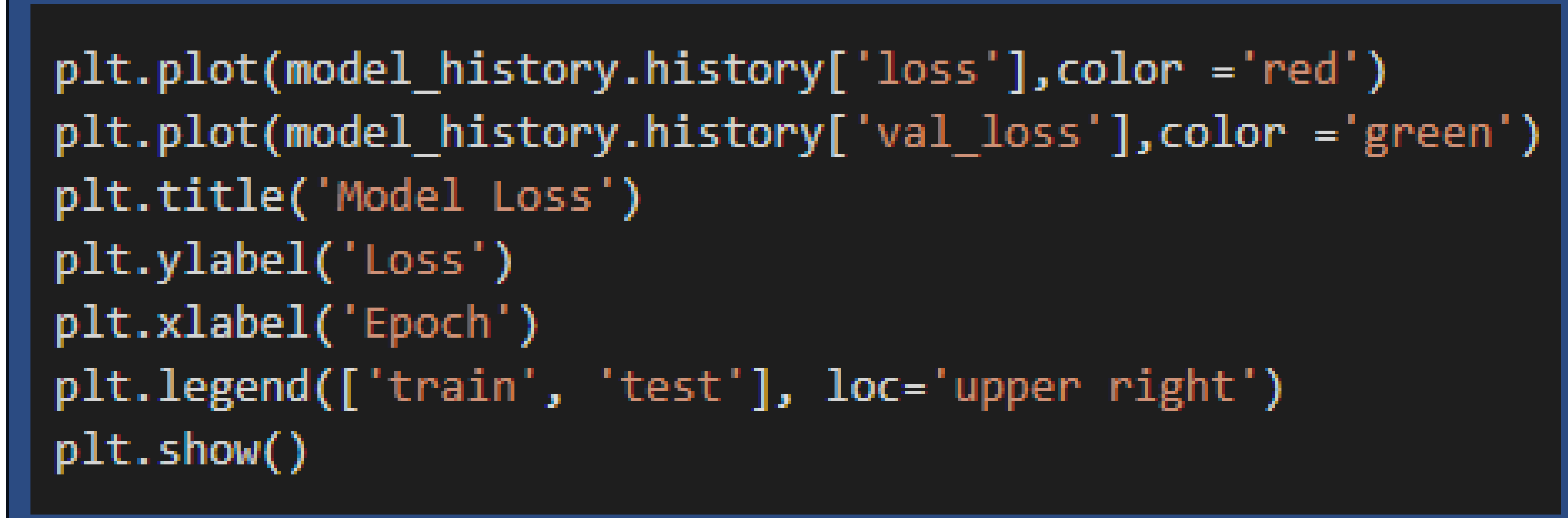




```
plt.plot(model_history.history['accuracy'],color ='red')
plt.plot(model_history.history['val_accuracy'],color ='green')
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

Summarize history for loss

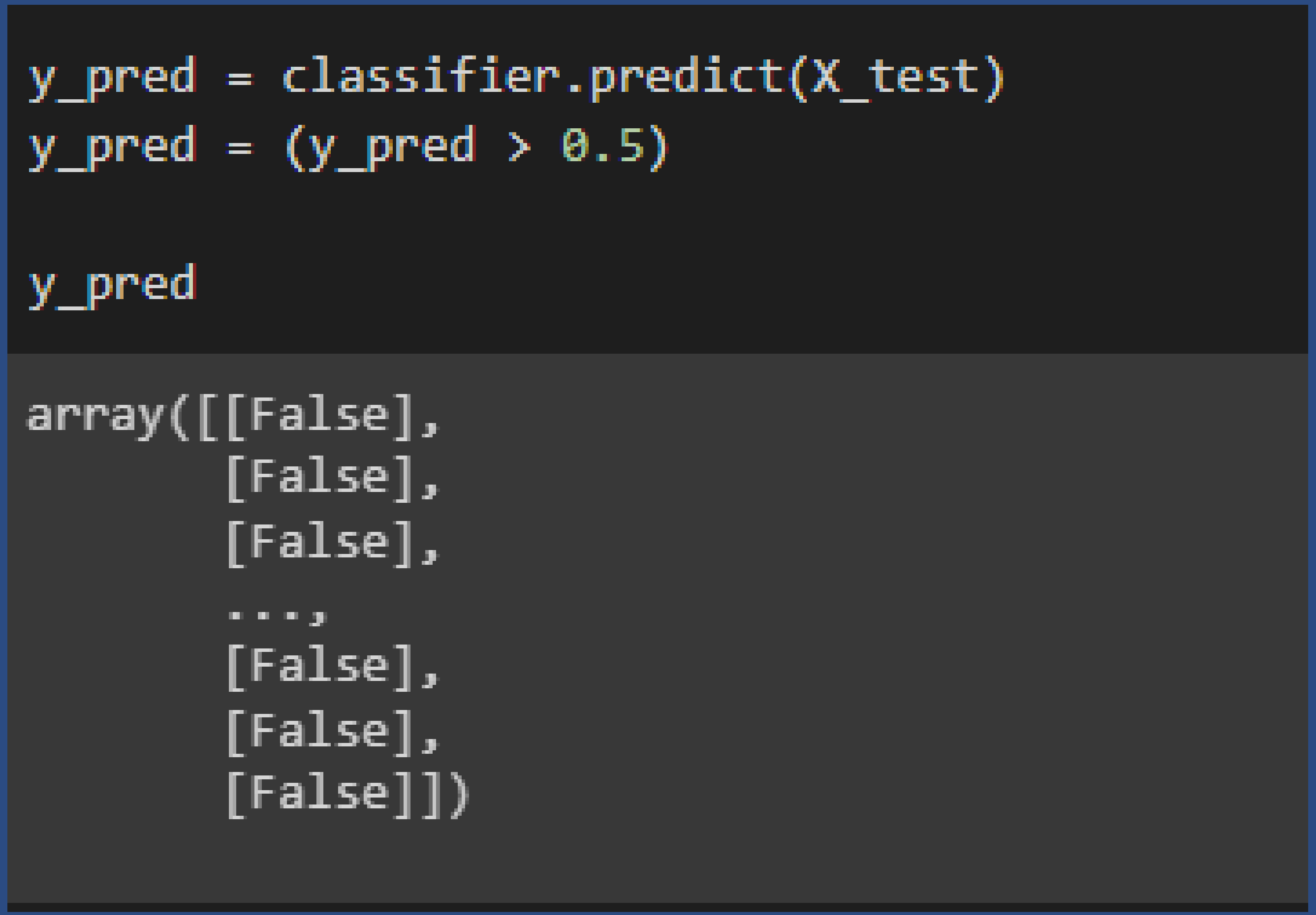




```
plt.plot(model_history.history['loss'],color ='red')
plt.plot(model_history.history['val_loss'],color ='green')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
```

Making the predictions and evaluating the model

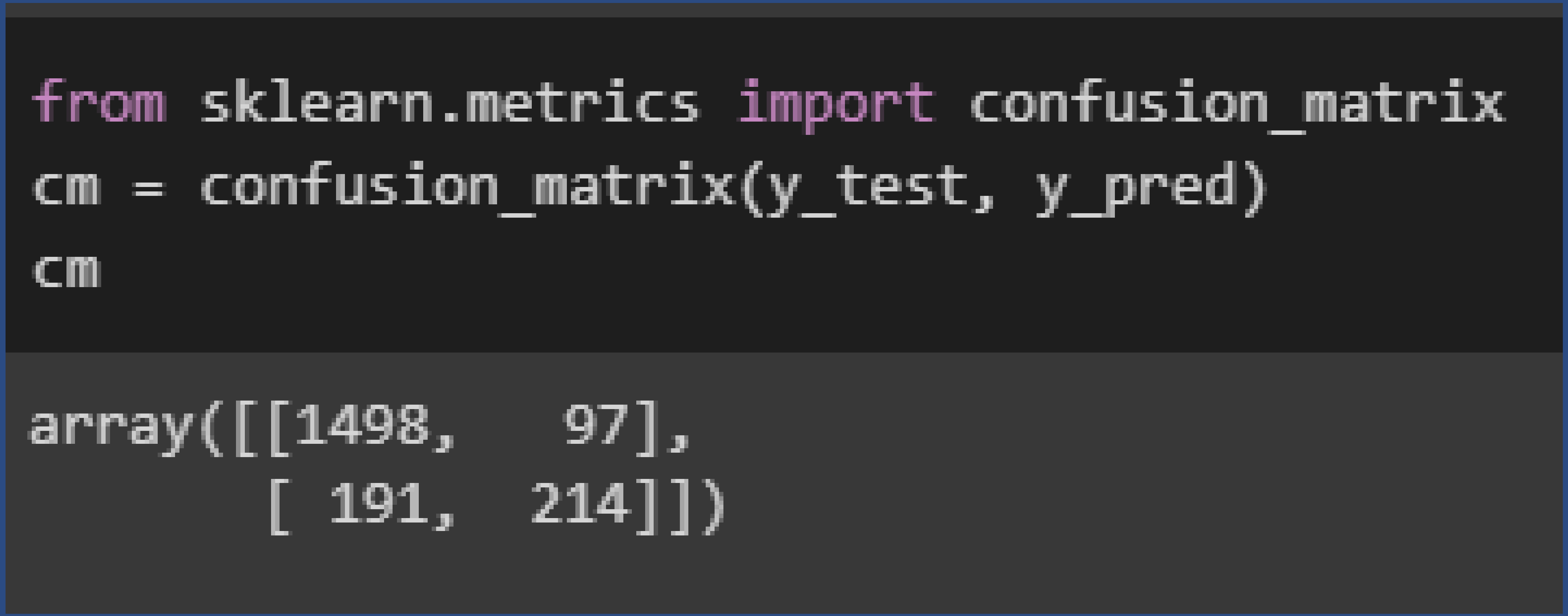
Predicting the Test set results



```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

y_pred
```

```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

Making the Confusion Matrix

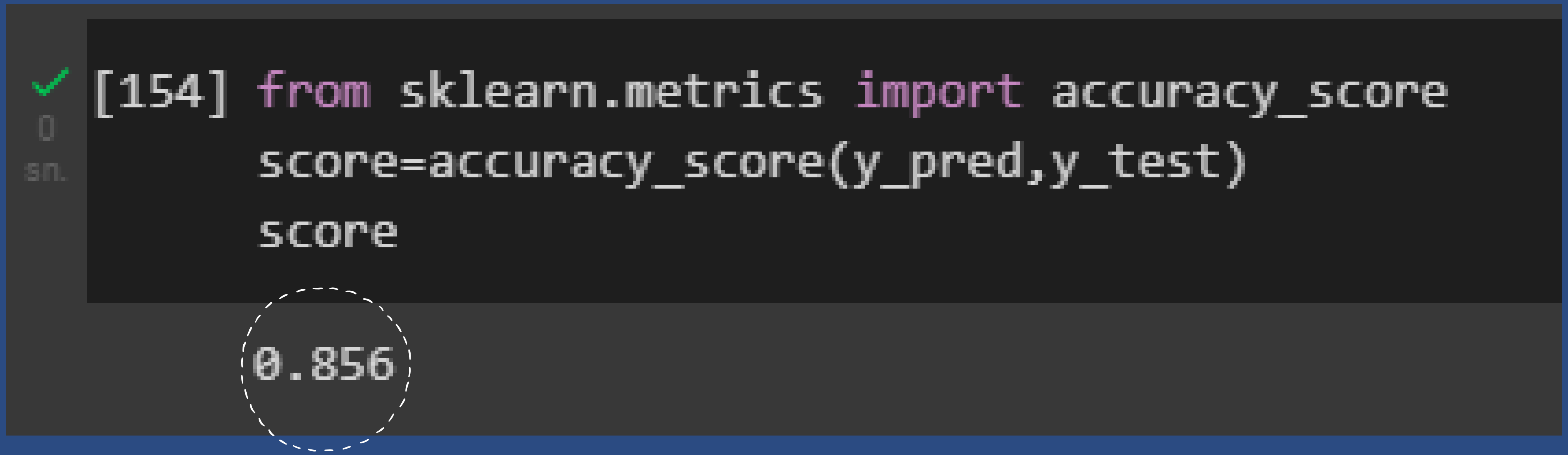


```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1498,   97],
       [ 191,  214]])
```

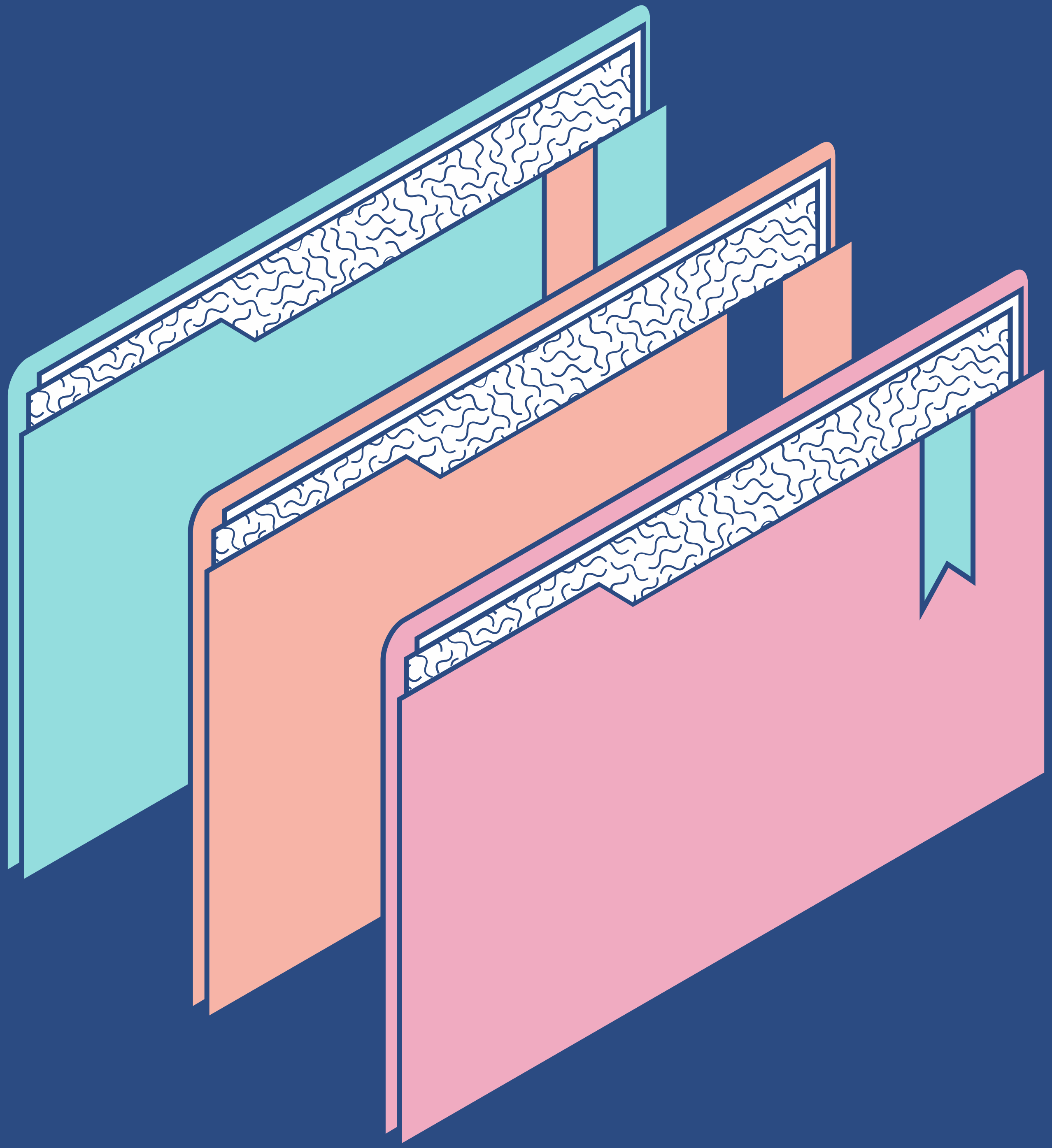
[matrix]

Calculate the Accuracy



```
[154] from sklearn.metrics import accuracy_score
      score=accuracy_score(y_pred,y_test)
      score
```

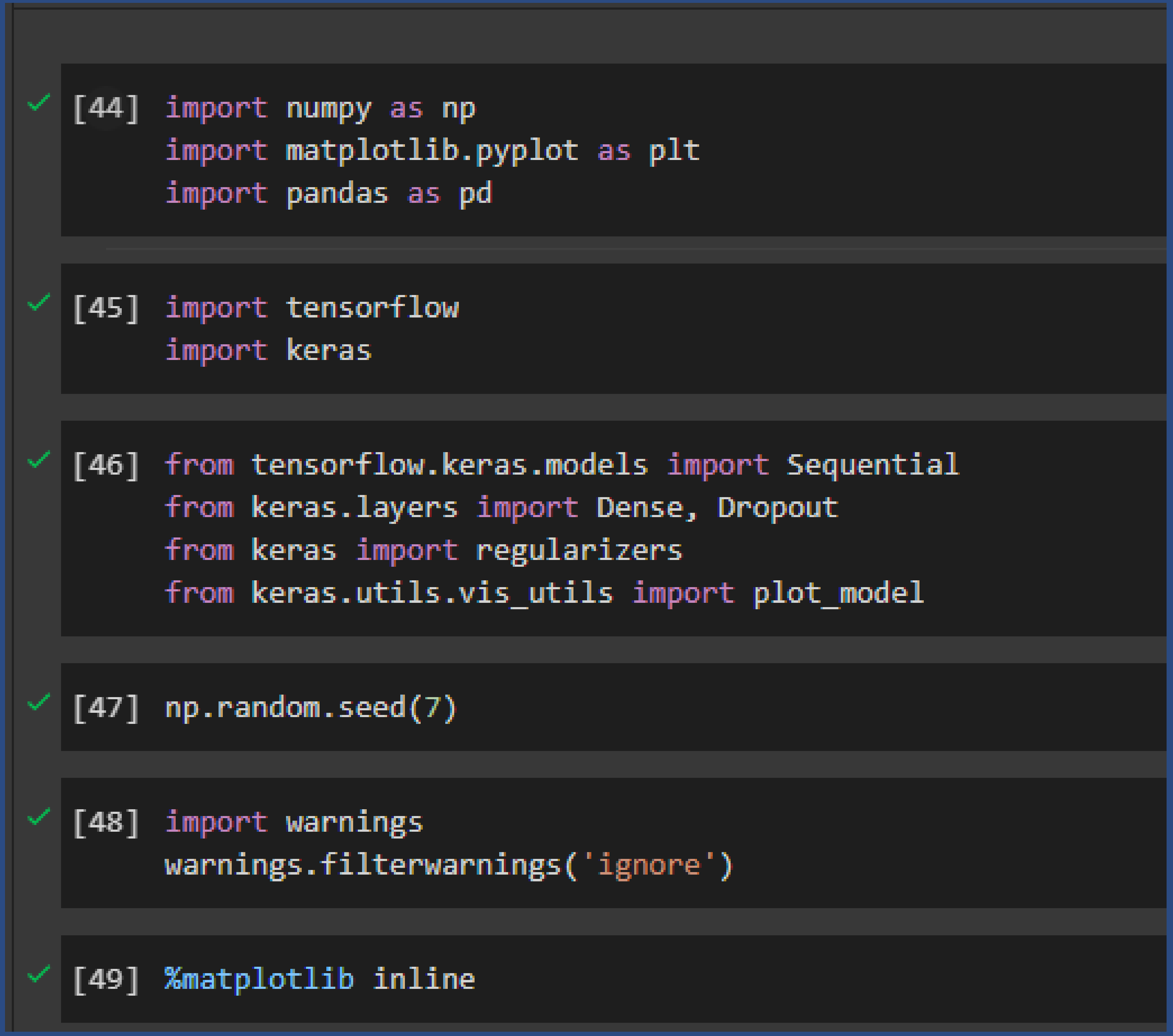
```
0.856
```



2.

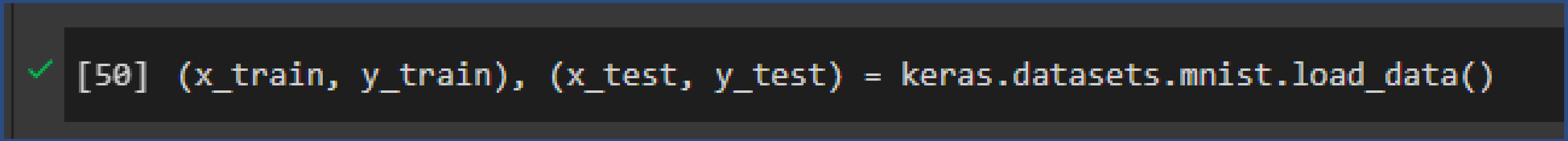
A) USE THE MNIST DATASET İN KERAS AND PERFORM THE CLASSİFİCATİON OF THE DİGİTS USİNG ANN. B) COMPARE THE ACCURACY OF YOUR MODEL WİTHOUT HİDDEN LAYER, WİTH ONE HİDDEN LAYER AND WİTH TWO HİDDEN LAYERS

Importing Necessary Libraries:



```
✓ [44] import numpy as np
       import matplotlib.pyplot as plt
       import pandas as pd

✓ [45] import tensorflow
       import keras

✓ [46] from tensorflow.keras.models import Sequential
       from keras.layers import Dense, Dropout
       from keras import regularizers
       from keras.utils.vis_utils import plot_model

✓ [47] np.random.seed(7)

✓ [48] import warnings
       warnings.filterwarnings('ignore')

✓ [49] %matplotlib inline
```

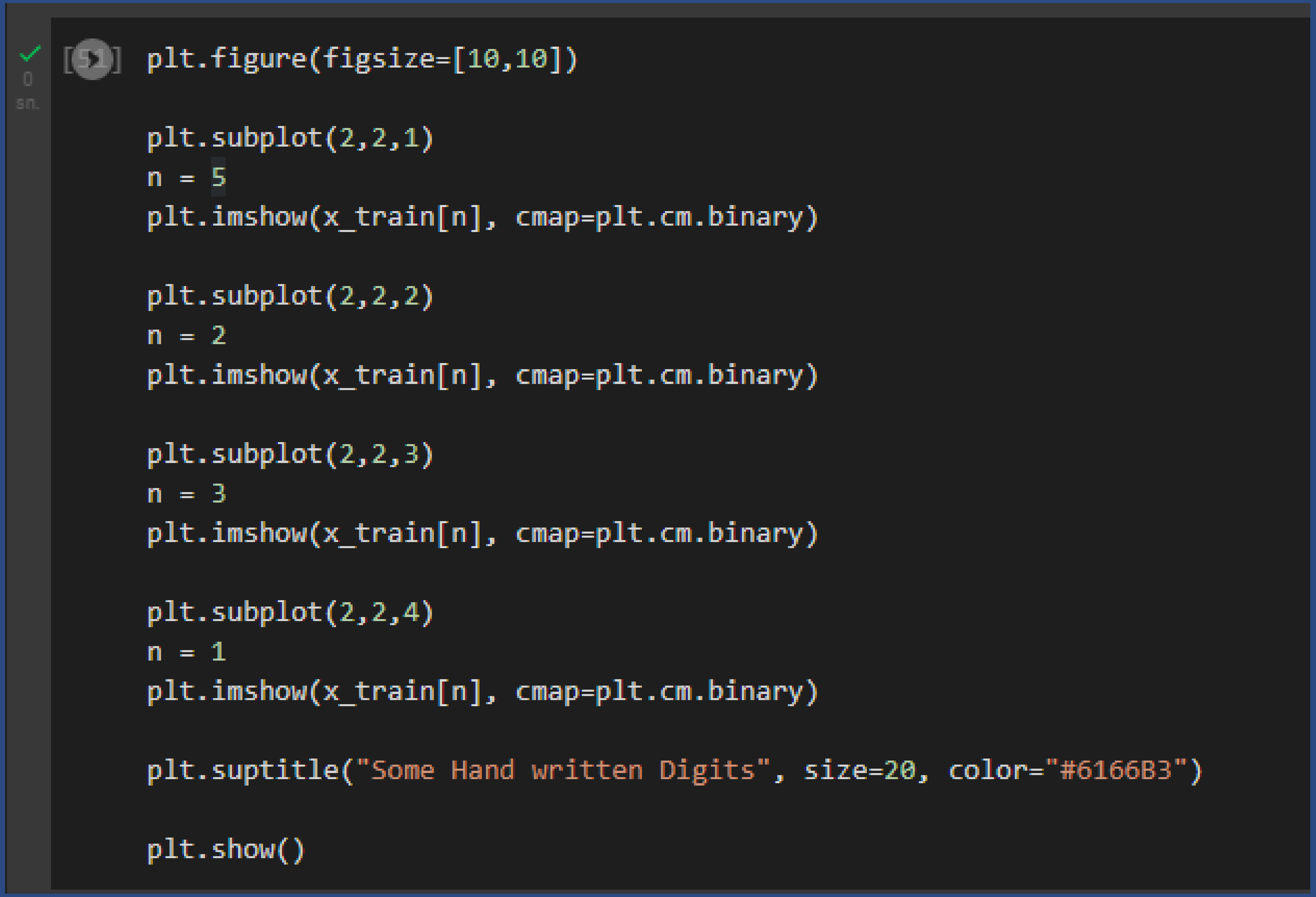
Importing MNIST Data:



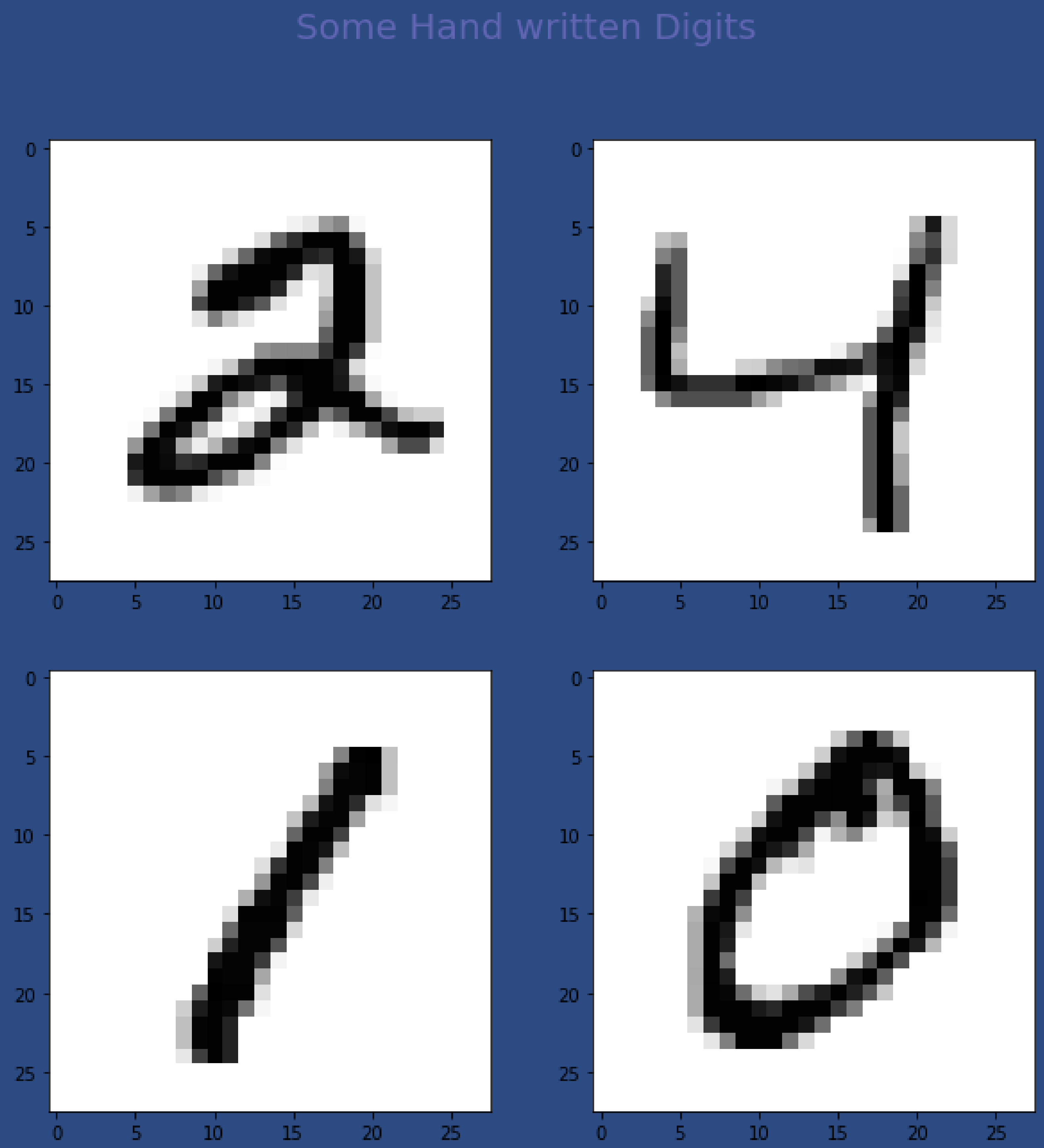
```
✓ [50] (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

Visualizing MNIST Data:

Visualizing (Hidden Input)

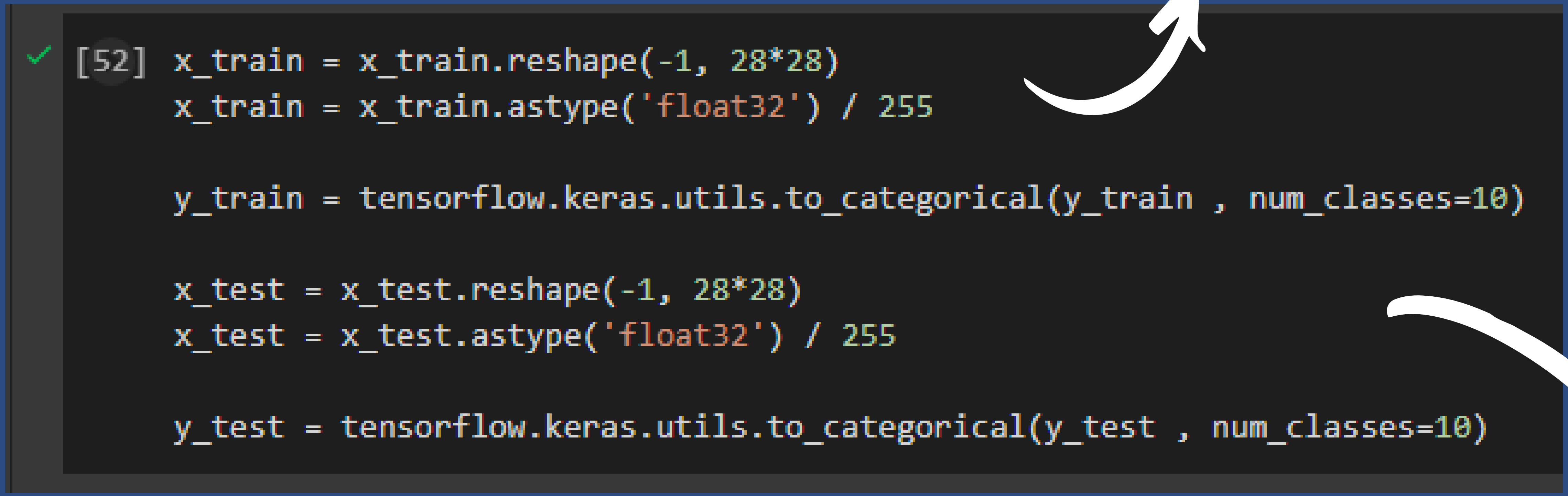


```
plt.figure(figsize=[10,10])

plt.subplot(2,2,1)
n = 5
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.subplot(2,2,2)
n = 2
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.subplot(2,2,3)
n = 3
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.subplot(2,2,4)
n = 1
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.suptitle("Some Hand written Digits", size=20, color="#6166B3")

plt.show()
```



Pre-Processing The Data:

Fixing the dimensions of the train set

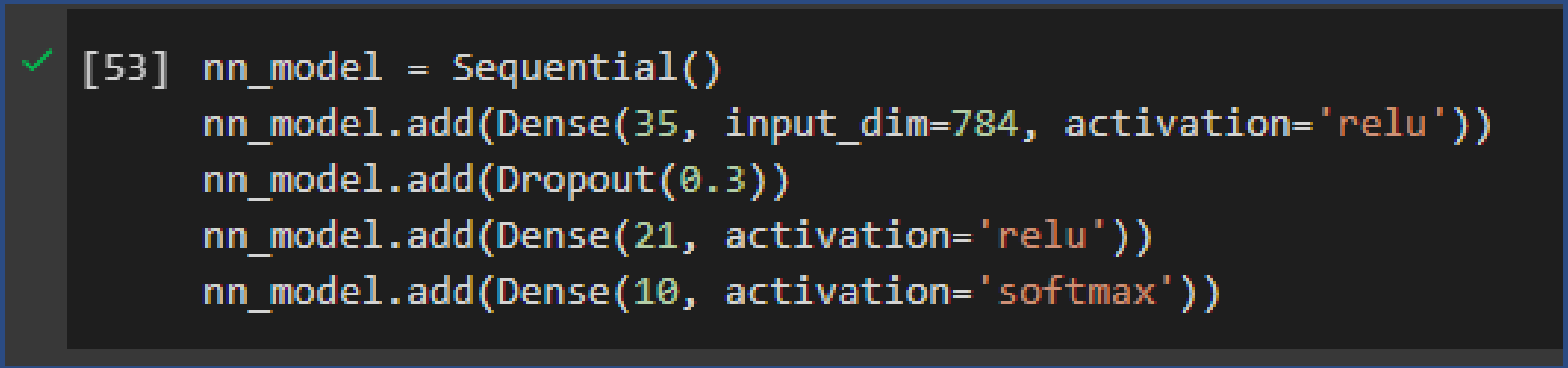


```
[52] x_train = x_train.reshape(-1, 28*28)
     x_train = x_train.astype('float32') / 255

     y_train = tensorflow.keras.utils.to_categorical(y_train , num_classes=10)

     x_test = x_test.reshape(-1, 28*28)
     x_test = x_test.astype('float32') / 255

     y_test = tensorflow.keras.utils.to_categorical(y_test , num_classes=10)
```

Fixing the dimensions of the test set

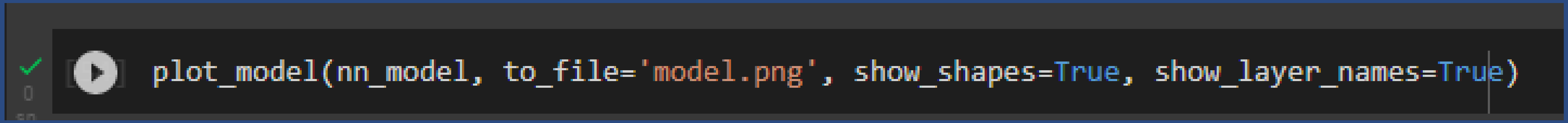
Designing The Neural Network:

Making the model (Hidden Output)

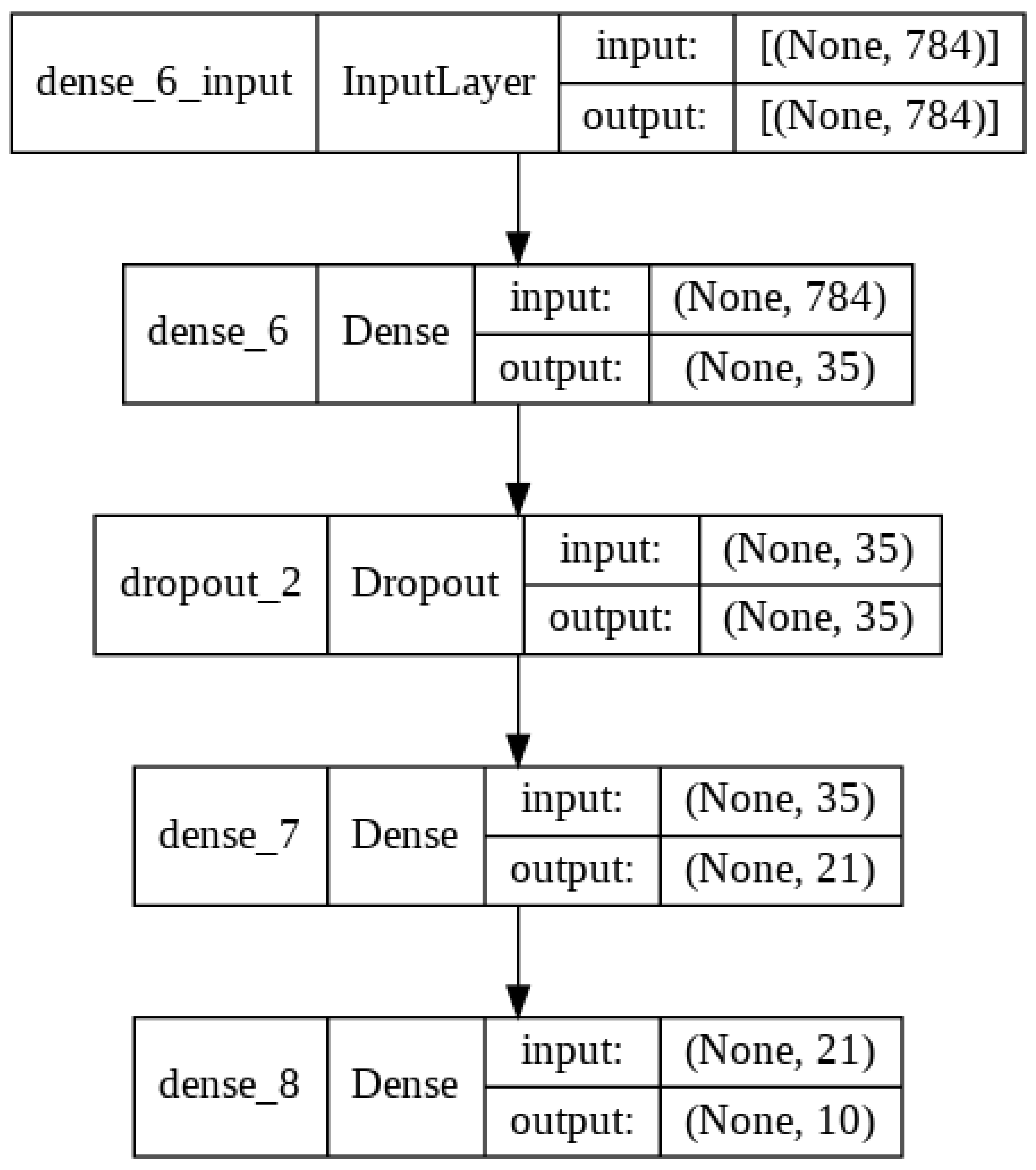


```
✓ [53] nn_model = Sequential()
       nn_model.add(Dense(35, input_dim=784, activation='relu'))
       nn_model.add(Dropout(0.3))
       nn_model.add(Dense(21, activation='relu'))
       nn_model.add(Dense(10, activation='softmax'))
```

Visualizing the model (Hidden Input)



```
plot_model(nn_model, to_file='model.png', show_shapes=True, show_layer_names=True)
```



dense_6_input	InputLayer	input:	[(None, 784)]
		output:	[(None, 784)]

dense_6	Dense	input:	(None, 784)
		output:	(None, 35)

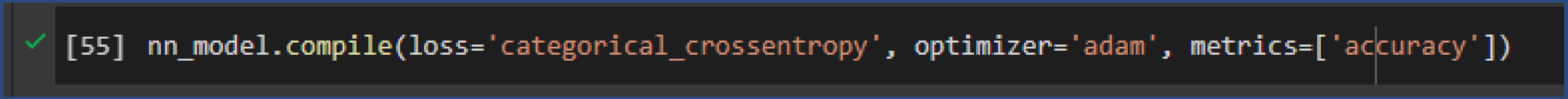
dropout_2	Dropout	input:	(None, 35)
		output:	(None, 35)

dense_7	Dense	input:	(None, 35)
		output:	(None, 21)

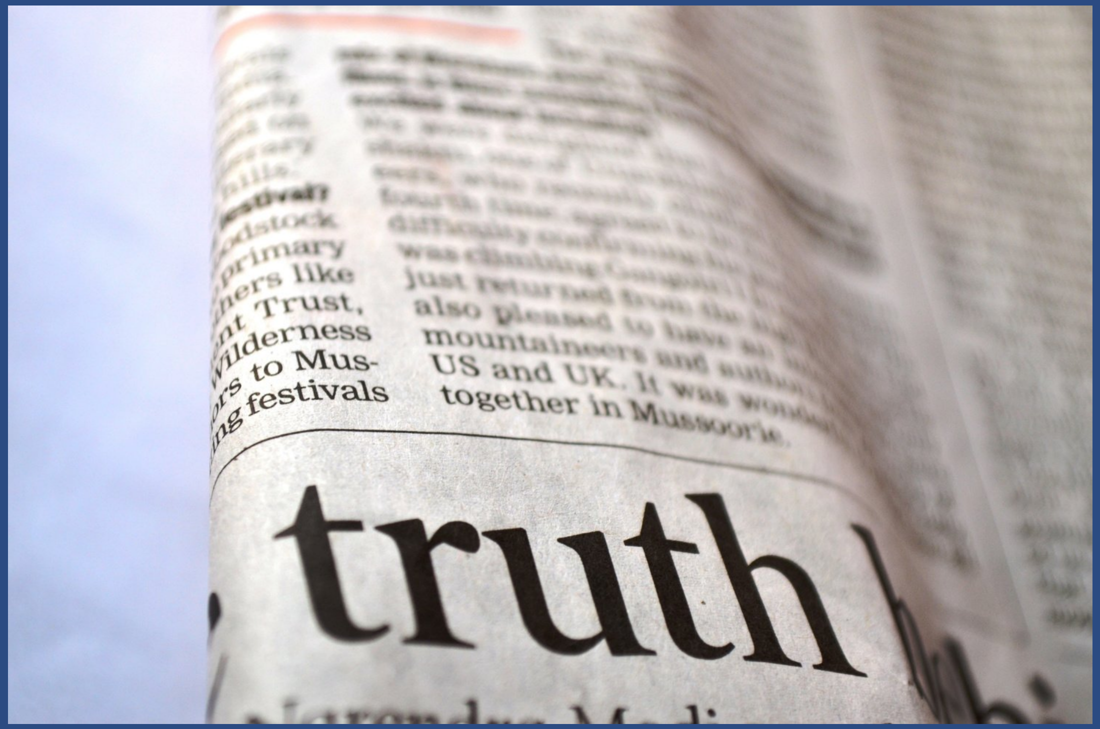
dense_8	Dense	input:	(None, 21)
		output:	(None, 10)

Compiling The Model:

Compiling The Model:

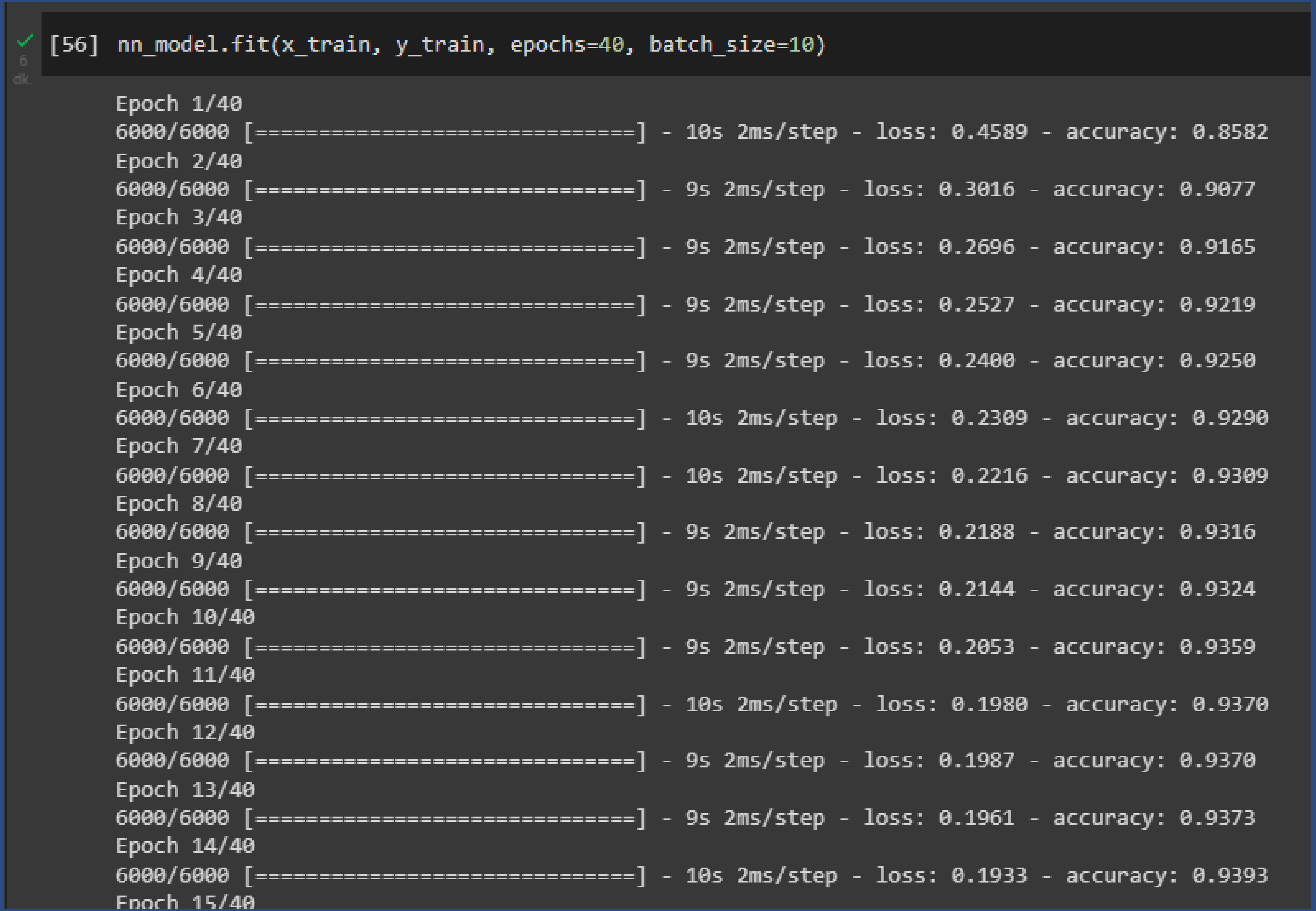


```
[55] nn_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```



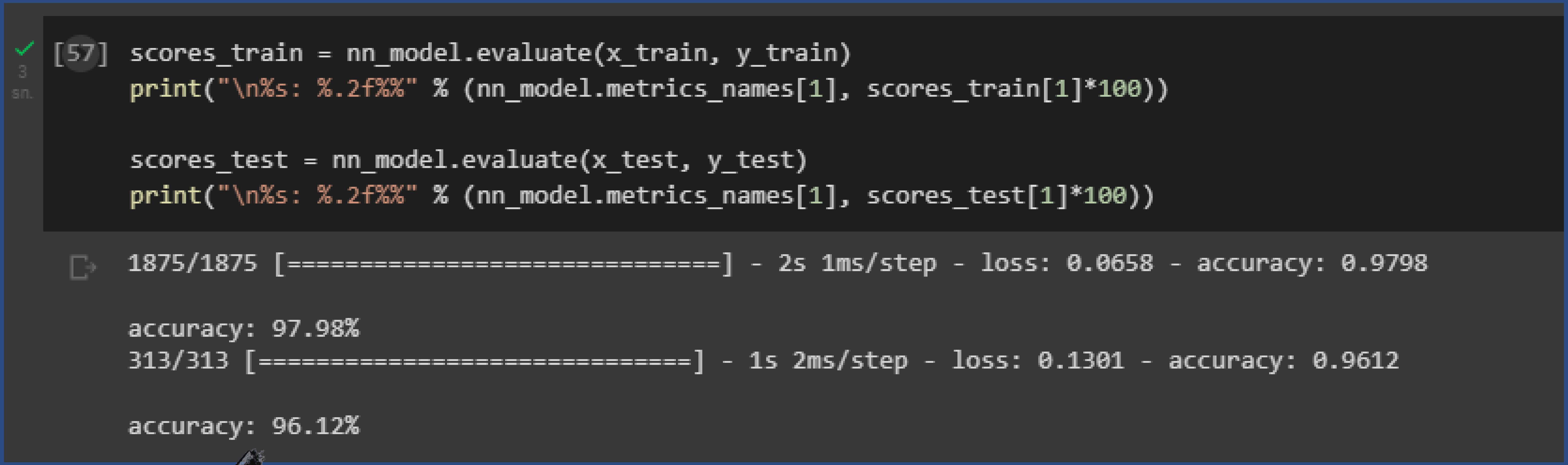
Training The Model:

Fitting the model



```
[56] nn_model.fit(x_train, y_train, epochs=40, batch_size=10)

     Epoch 1/40
     6000/6000 [==============================] - 10s 2ms/step - loss: 0.4589 - accuracy: 0.8582
     Epoch 2/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.3016 - accuracy: 0.9077
     Epoch 3/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.2696 - accuracy: 0.9165
     Epoch 4/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.2527 - accuracy: 0.9219
     Epoch 5/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.2400 - accuracy: 0.9250
     Epoch 6/40
     6000/6000 [==============================] - 10s 2ms/step - loss: 0.2309 - accuracy: 0.9290
     Epoch 7/40
     6000/6000 [==============================] - 10s 2ms/step - loss: 0.2216 - accuracy: 0.9309
     Epoch 8/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.2188 - accuracy: 0.9316
     Epoch 9/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.2144 - accuracy: 0.9324
     Epoch 10/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.2053 - accuracy: 0.9359
     Epoch 11/40
     6000/6000 [==============================] - 10s 2ms/step - loss: 0.1980 - accuracy: 0.9370
     Epoch 12/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.1987 - accuracy: 0.9370
     Epoch 13/40
     6000/6000 [==============================] - 9s 2ms/step - loss: 0.1961 - accuracy: 0.9373
     Epoch 14/40
     6000/6000 [==============================] - 10s 2ms/step - loss: 0.1933 - accuracy: 0.9393
     Epoch 15/40
```

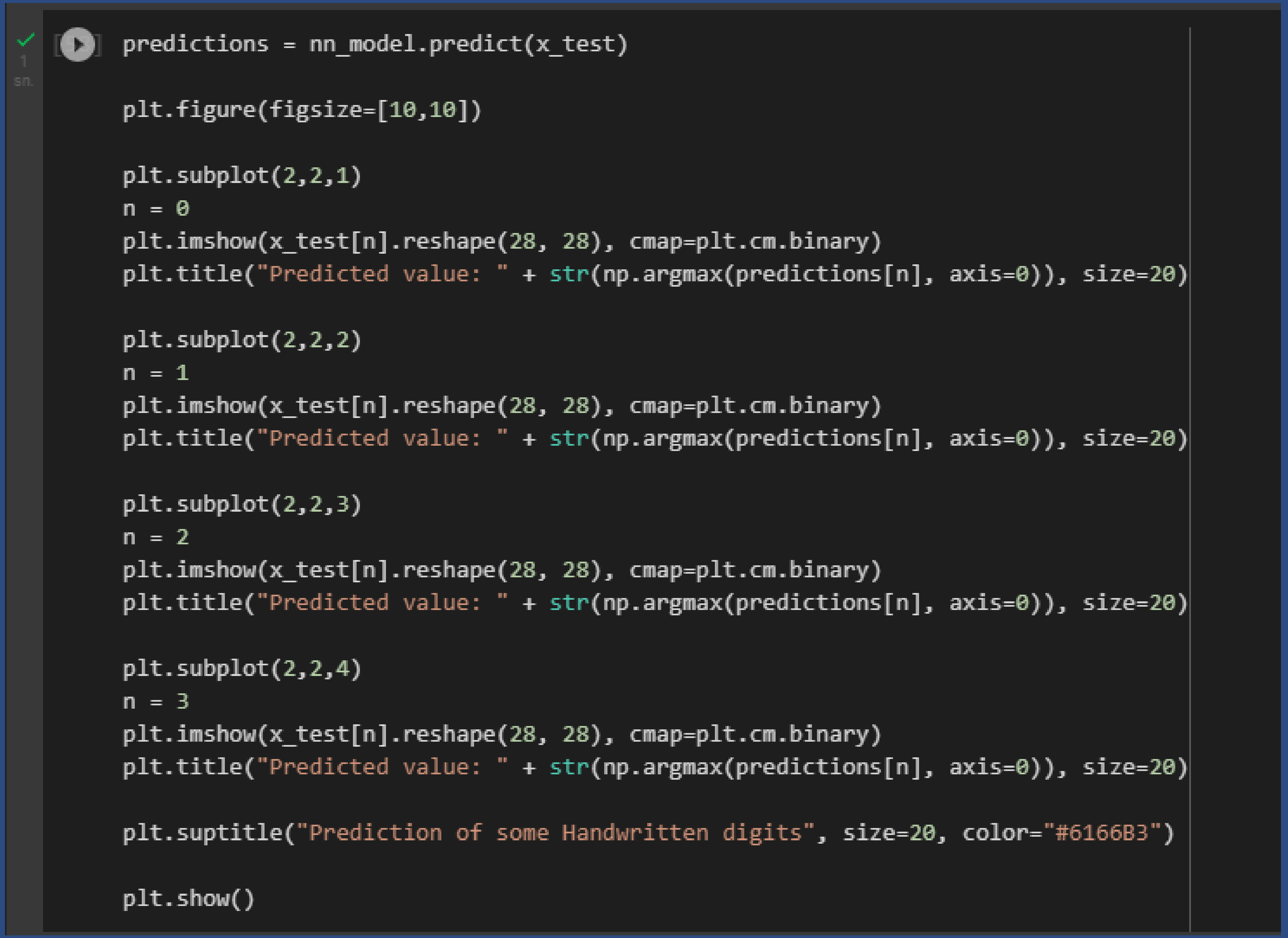
Evaluating The Model:



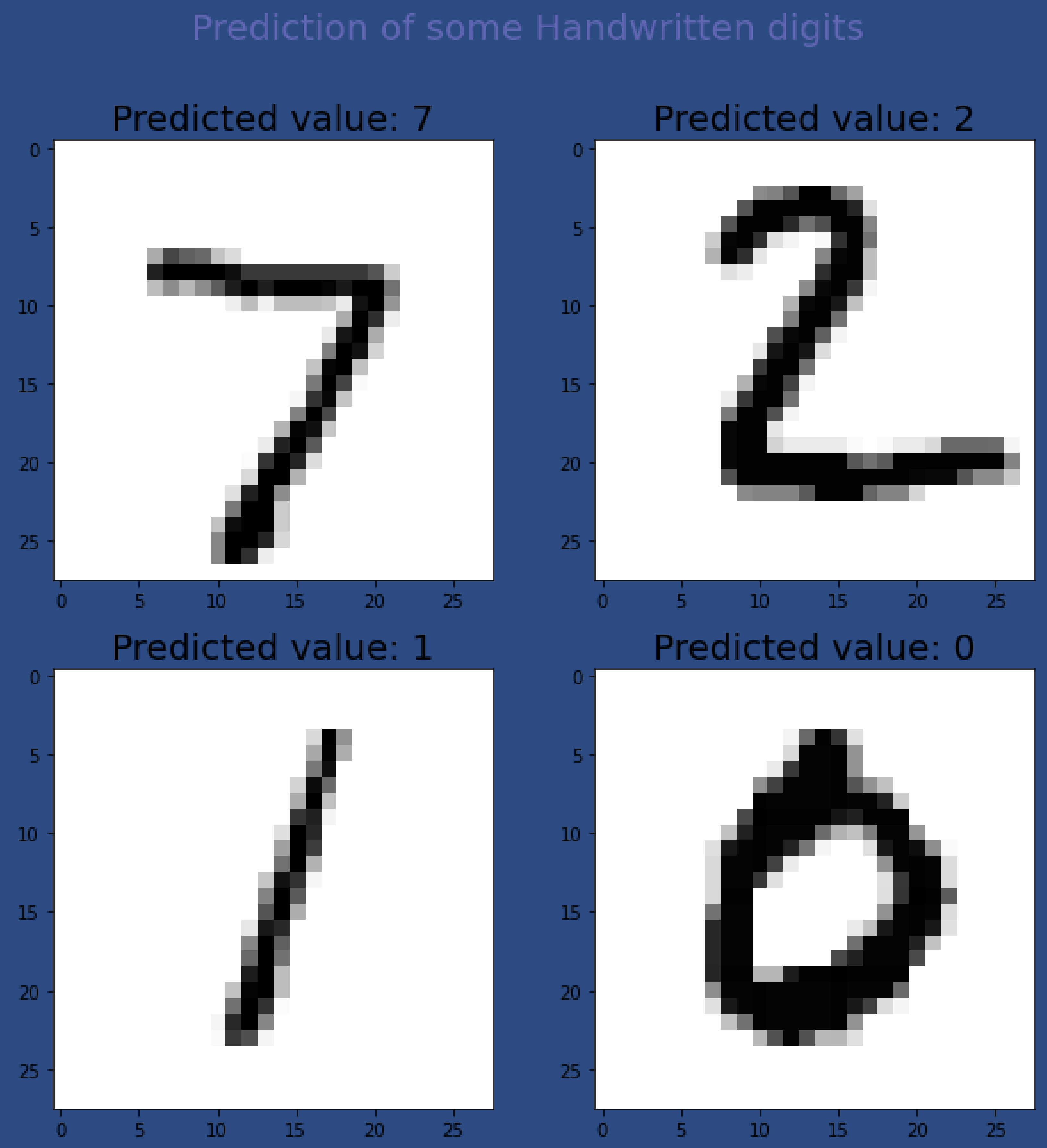
```
scores_train = nn_model.evaluate(x_train, y_train)
print("\n%s: %.2f%%" % (nn_model.metrics_names[1], scores_train[1]*100))

scores_test = nn_model.evaluate(x_test, y_test)
print("\n%s: %.2f%%" % (nn_model.metrics_names[1], scores_test[1]*100))
```

```
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0658 - accuracy: 0.9798

accuracy: 97.98%
313/313 [==============================] - 1s 2ms/step - loss: 0.1301 - accuracy: 0.9612

accuracy: 96.12%
```

Predictions (Hidden Input)



```
predictions = nn_model.predict(x_test)

plt.figure(figsize=[10,10])

plt.subplot(2,2,1)
n = 0
plt.imshow(x_test[n].reshape(28, 28), cmap=plt.cm.binary)
plt.title("Predicted value: " + str(np.argmax(predictions[n], axis=0)), size=20)

plt.subplot(2,2,2)
n = 1
plt.imshow(x_test[n].reshape(28, 28), cmap=plt.cm.binary)
plt.title("Predicted value: " + str(np.argmax(predictions[n], axis=0)), size=20)

plt.subplot(2,2,3)
n = 2
plt.imshow(x_test[n].reshape(28, 28), cmap=plt.cm.binary)
plt.title("Predicted value: " + str(np.argmax(predictions[n], axis=0)), size=20)

plt.subplot(2,2,4)
n = 3
plt.imshow(x_test[n].reshape(28, 28), cmap=plt.cm.binary)
plt.title("Predicted value: " + str(np.argmax(predictions[n], axis=0)), size=20)

plt.suptitle("Prediction of some Handwritten digits", size=20, color="#6166B3")

plt.show()
```



Prediction of some Handwritten digits

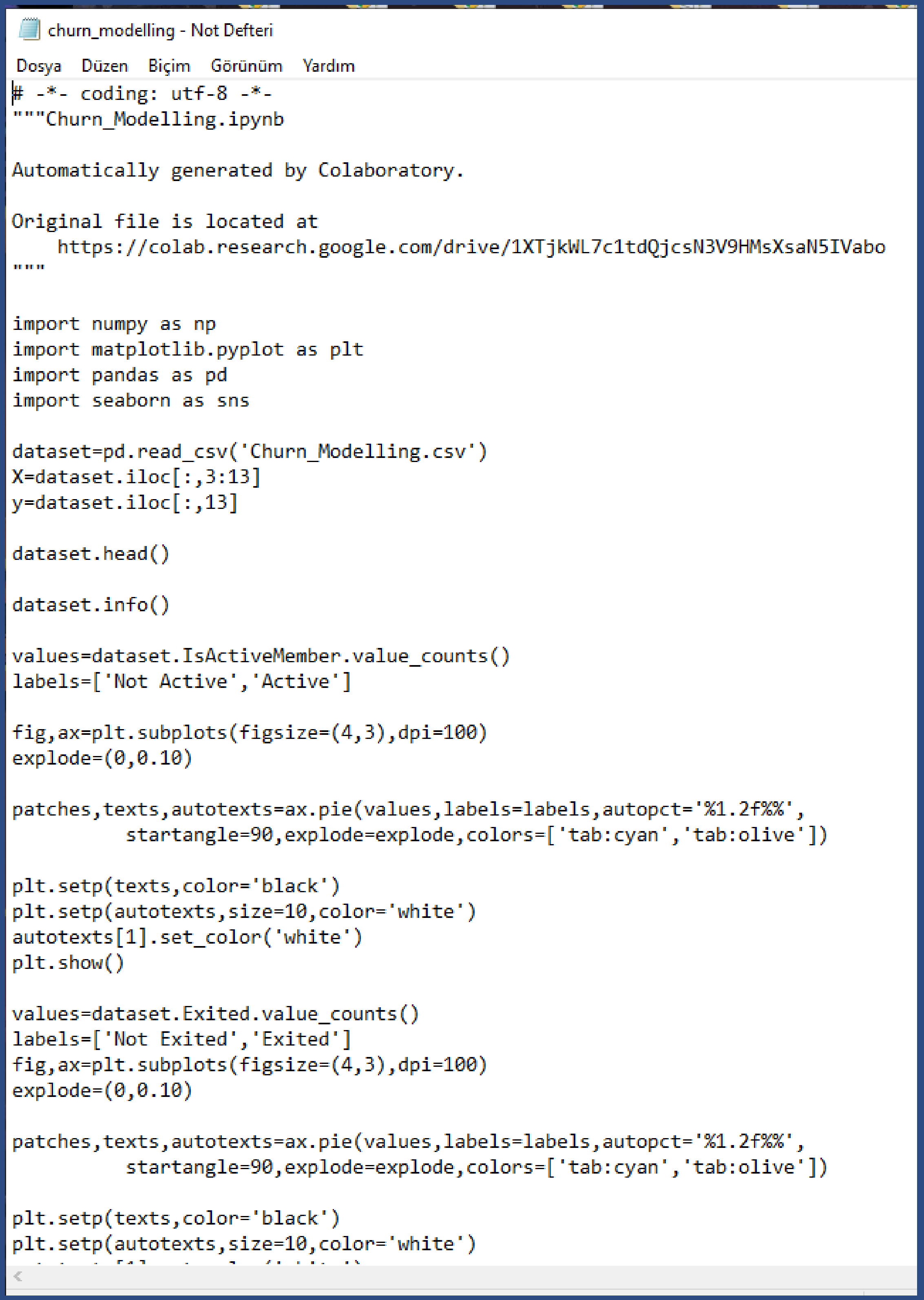
Free Resources

https://www.kaggle.com/

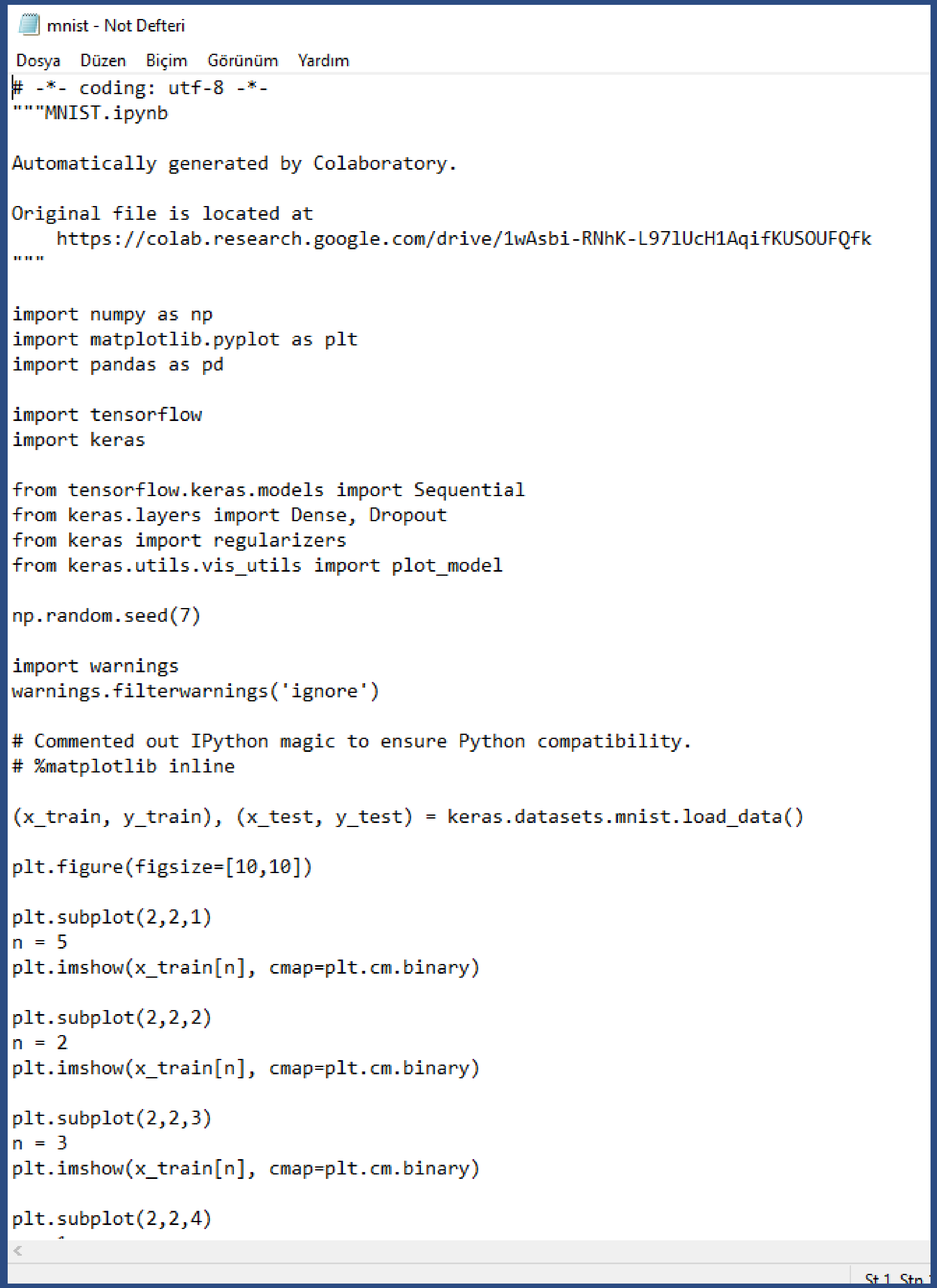
Course materials

ZEYNEP ÖZIŞIL
180722023
COMPUTER ENGİNEERİNG

I ADDED THE .PY EXTENSION FILE OF MY CODES TO THE EXPLANATION SECTION OF THE ASSIGNMENT WINDOW.


churn_modelling - Not Defteri
Dosya Düzen Biçim Görünüm Yardım

```
# -*- coding: utf-8 -*-
"""Churn_Modelling.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1XTjkWL7c1tdQjcsN3V9HMsXsaN5IVabo
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

dataset=pd.read_csv('Churn_Modelling.csv')
X=dataset.iloc[:,3:13]
y=dataset.iloc[:,13]

dataset.head()

dataset.info()

values=dataset.IsActiveMember.value_counts()
labels=['Not Active','Active']

fig,ax=plt.subplots(figsize=(4,3),dpi=100)
explode=(0,0.10)

patches,texts,autotexts=ax.pie(values,labels=labels,autopct='%1.2f%%',
          startangle=90,explode=explode,colors=['tab:cyan','tab:olive'])

plt.setp(texts,color='black')
plt.setp(autotexts,size=10,color='white')
autotexts[1].set_color('white')
plt.show()

values=dataset.Exited.value_counts()
labels=['Not Exited','Exited']
fig,ax=plt.subplots(figsize=(4,3),dpi=100)
explode=(0,0.10)

patches,texts,autotexts=ax.pie(values,labels=labels,autopct='%1.2f%%',
          startangle=90,explode=explode,colors=['tab:cyan','tab:olive'])

plt.setp(texts,color='black')
plt.setp(autotexts,size=10,color='white')
```



```
# -*- coding: utf-8 -*-
"""MNIST.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1wAsbi-RNhK-L97lUcH1AqifKUSOUFQfk
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import tensorflow
import keras

from tensorflow.keras.models import Sequential
from keras.layers import Dense, Dropout
from keras import regularizers
from keras.utils.vis_utils import plot_model

np.random.seed(7)

import warnings
warnings.filterwarnings('ignore')

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

plt.figure(figsize=[10,10])

plt.subplot(2,2,1)
n = 5
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.subplot(2,2,2)
n = 2
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.subplot(2,2,3)
n = 3
plt.imshow(x_train[n], cmap=plt.cm.binary)

plt.subplot(2,2,4)
```