classification

February 19, 2022

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
```

[ ]:
```
np.random.seed(40)
```

0.0.1 Load & check the data:

[ ]:
```
# Loading the MNIST dataset from sklearn
from sklearn.datasets import fetch_openml
mnist_jay = fetch_openml('mnist_784', version=1, as_frame=True)
mnist_jay.keys()
```

[ ]:
```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names',
'target_names', 'DESCR', 'details', 'url'])
```

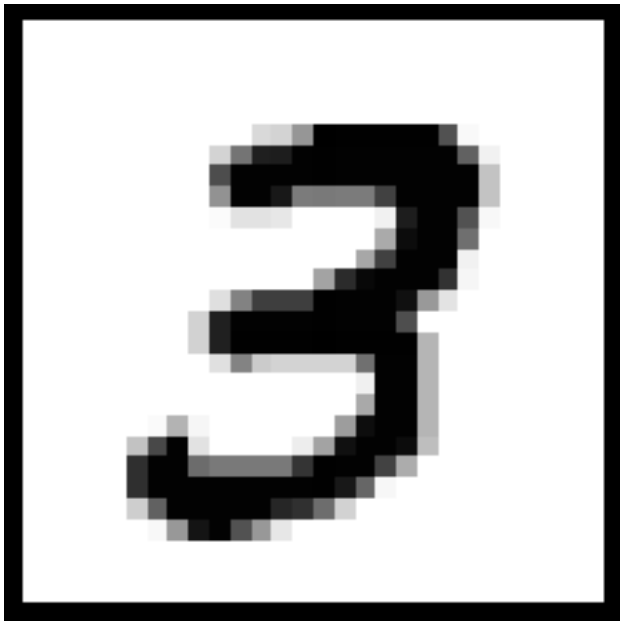
[ ]:
```
#Assign the data and target to a ndarray
X_jay, y_jay = mnist_jay['data'], mnist_jay['target']
X_jay.shape, y_jay.shape
```

[ ]:
```
((70000, 784), (70000,))
```

[ ]:
```
# print the type of X_jay
print(type(X_jay))
print(type(y_jay))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

[ ]:
```
some_digit = X_jay.to_numpy()[7]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap=mpl.cm.binary)
plt.axis("off")
plt.show()
```



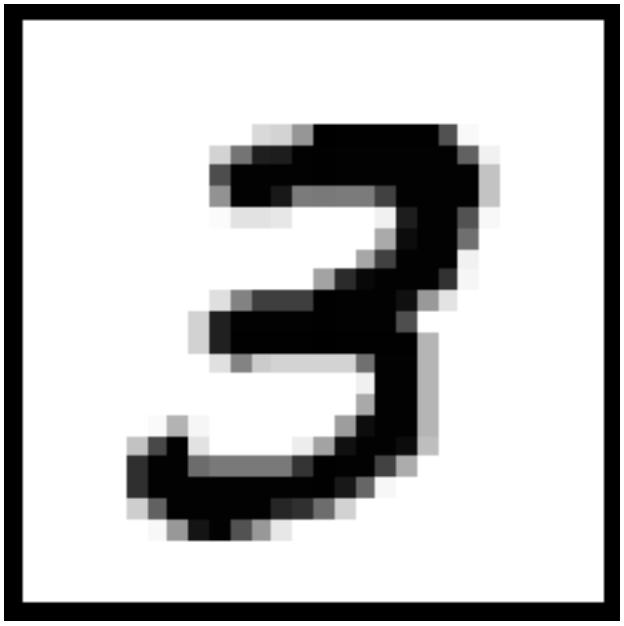
[ ]:
```
y_jay[7]
```

[ ]:
```
'3'
```

[ ]:
```
y_jay = y_jay.astype(np.uint8)
```

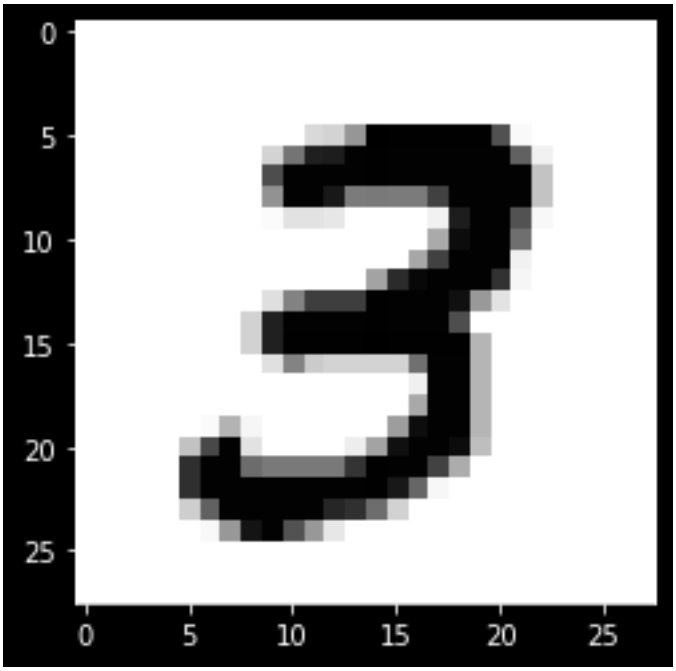
[ ]:
```
some_digit = X_jay.to_numpy()[7]
some_digit_image = some_digit.reshape(28, 28)
```

[ ]:
```
plt.imshow(some_digit_image, cmap=mpl.cm.binary, interpolation="nearest")
plt.axis("off")
plt.show()
```

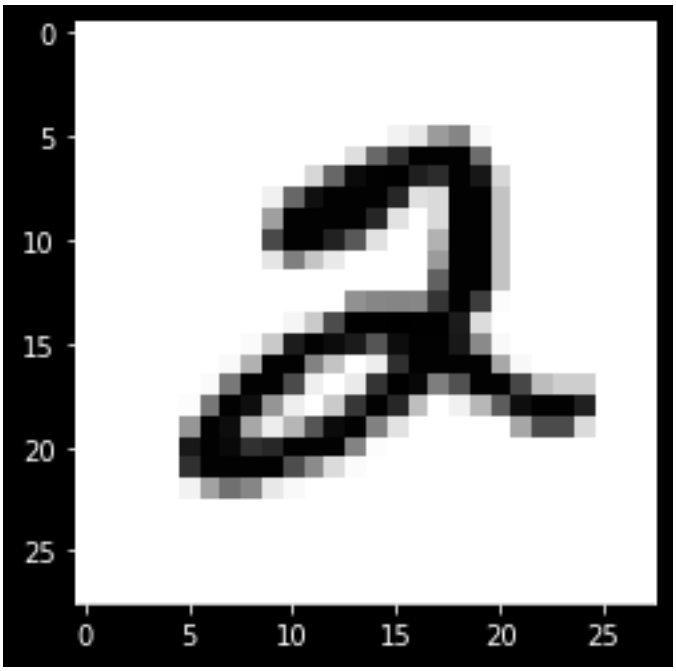


[ ]:
```
some_digit1 = X_jay.to_numpy()[7]
some_digit2 = X_jay.to_numpy()[5]
some_digit3 = X_jay.to_numpy()[0]
```

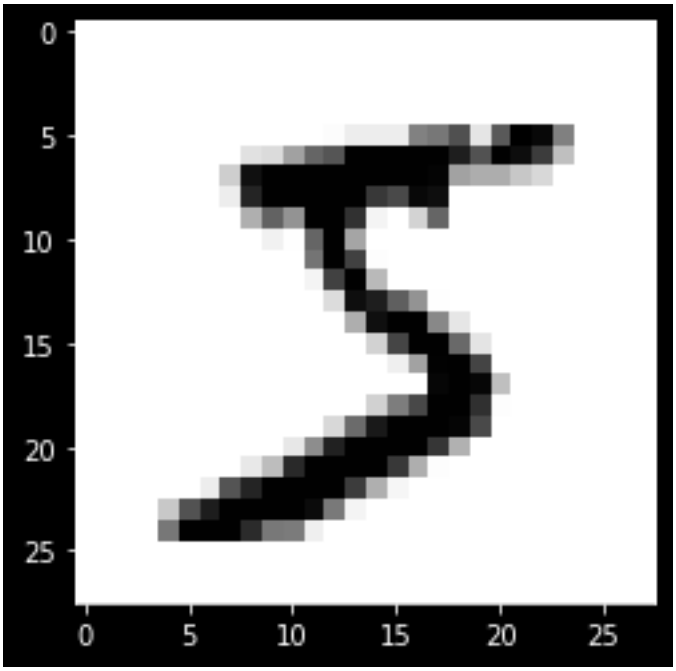
[ ]:
```
# Use imshow method to plot the values of the three variables you defined in␣
 ↪the above point.
plt.imshow(some_digit1.reshape(28, 28), cmap=mpl.cm.binary,␣
 ↪interpolation='nearest')
display(plt.show())
plt.imshow(some_digit2.reshape(28, 28), cmap=mpl.cm.binary,␣
 ↪interpolation='nearest')
display(plt.show())
plt.imshow(some_digit3.reshape(28, 28), cmap=mpl.cm.binary,␣
 ↪interpolation='nearest')
display(plt.show())
```




```
None
```




```
None
```



None

0.0.2 Pre-process the data

[ ]:
```
#       The current target values range from 0 to 9 i.e. 10 classes. Transform␣
 ↪the target variable to 3 classes as follows:
# a.        Any digit between 0 and 3 inclusive should be assigned a target␣
 ↪value of 0
# b.        Any digit between 4 and 7 inclusive should be assigned a target␣
 ↪value of 1
# c.        Any digit between 8 and 11 inclusive should be assigned a target␣
 ↪value of 9
# d.        Use the following code to do this:
y_jay_new = np.where(y_jay < 4, 0, y_jay)
y_jay_new = np.where((y_jay_new > 3) & (y_jay_new < 8) , 1, y_jay_new)
y_jay_new = np.where((y_jay_new > 7) & (y_jay_new < 12), 9, y_jay_new)
print(y_jay_new)
```

```
[1 0 1 … 1 1 1]
```

[ ]:
```
# Print the frequencies of each of the three target classes in y_jay_new
unique, counts = np.unique(y_jay_new, return_counts=True)
print(np.asarray((unique, counts)).T)
```

```
[[    0 28911]
 [    1 27306]
 [    9 13783]]
```

[ ]:
```
# Split your data into train test. Assign the first 60,000 records for training␣
↪and the last 10,000 records for testing.
X_train, X_test = X_jay[:60000], X_jay[60000:]
y_train, y_test = y_jay_new[:60000], y_jay_new[60000:]
```

0.0.3 Build Classification Models

[ ]:
```
# Train a Naive Bayes classifier using the training data. Name the classifier␣
↪NB_clf_firstname.
from sklearn.naive_bayes import GaussianNB
NB_clf_jay = GaussianNB()
NB_clf_jay.fit(X_train, y_train)

# Predict the class labels for the test data using the trained classifier.␣
↪Assign the result to y_pred_firstname.
y_pred_jay = NB_clf_jay.predict(X_test)
print(y_pred_jay)
```

```
[9 0 9 … 9 9 1]
```

[ ]:
```
# Train a SGD classifier using the training data. Name the classifier␣
↪SGD_clf_firstname.
from sklearn.linear_model import SGDClassifier
SGD_clf_jay = SGDClassifier()
SGD_clf_jay.fit(X_train, y_train)

# Predict the class labels for the test data using the trained classifier.␣
↪Assign the result to y_pred_firstname.
y_pred_jay = SGD_clf_jay.predict(X_test)
print(y_pred_jay)
```

```
[1 0 0 … 1 1 1]
```

[ ]:
```
# Use the classifier to predict the three variables you defined in point 7␣
↪above.
print(NB_clf_jay.predict([some_digit1]))
print(NB_clf_jay.predict([some_digit2]))
print(NB_clf_jay.predict([some_digit3]))
```

```
[9]
[9]
[0]

C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

```
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but GaussianNB was fitted with feature names
  warnings.warn(
```

[ ]:
```
# Predict the class labels for the test data using the trained classifier.␣
↪Assign the result to y_pred_firstname1.
y_pred_jay1 = SGD_clf_jay.predict(X_test)
# Use the classifier to predict the three variables you defined in point 7␣
↪above.
print(SGD_clf_jay.predict([some_digit1]))
print(SGD_clf_jay.predict([some_digit2]))
print(SGD_clf_jay.predict([some_digit3]))
```

```
[0]
[0]
[1]
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SGDClassifier was fitted with feature
names
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SGDClassifier was fitted with feature
names
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SGDClassifier was fitted with feature
names
  warnings.warn(
```

[ ]:
```
# Print the accuracy of the classifier on the test data.
print(NB_clf_jay.score(X_test, y_test) * 100)

# Print the accuracy of the classifier on the test data.
print(SGD_clf_jay.score(X_test, y_test) * 100)
```

```
38.2
83.65
```

[ ]:
```
# Train a KNN classifier using the training data. Name the classifier␣
↪KNN_clf_firstname.
from sklearn.neighbors import KNeighborsClassifier
KNN_clf_jay = KNeighborsClassifier()
KNN_clf_jay.fit(X_train, y_train)

# Predict the class labels for the test data using the trained classifier.␣
↪Assign the result to y_pred_firstname.
y_pred_jay = KNN_clf_jay.predict(X_test)
print(y_pred_jay)
```

```
# Predict the class labels for the test data using the trained classifier.␣
 ↪Assign the result to y_pred_firstname2.
y_pred_jay2 = KNN_clf_jay.predict(X_test)
# Use the classifier to predict the three variables you defined in point 7␣
 ↪above.
print(KNN_clf_jay.predict([some_digit1]))
print(KNN_clf_jay.predict([some_digit2]))
print(KNN_clf_jay.predict([some_digit3]))

print(KNN_clf_jay.score(X_test, y_test) * 100)
```

```
[1 0 0 … 1 1 1]
```

```
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
```

```
[0]
[0]
```

```
C:\Users\asus\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
```

```
[1]
97.48
```

[ ]: