logistic_regression

January 25, 2022

1 Assignment of Logistic Regression

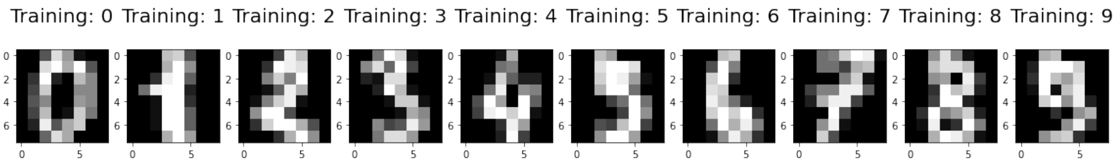
[ ]:
```
# imoprt libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
from sklearn.datasets import load_digits
digits= load_digits()
```

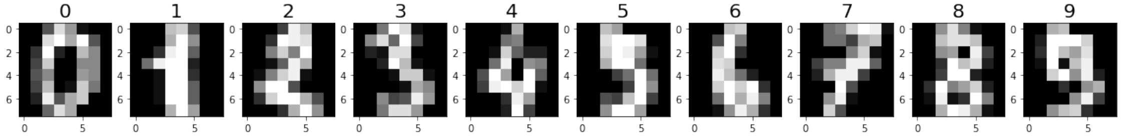
[ ]:
```
# input variables/features (x=data)
digits.data.shape
x=digits.data
# means 1797 pictures size 64=8x8
```

[ ]:
```
# output labels (y=target)
digits.target.shape
y=digits.target
```

[ ]:
```
plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.data[0:10],digits.target[0:
 ↪10])):
    plt.subplot(1,10,index +1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Training: %i\n'  % label, fontsize = 20)  ## error to be removed
```



[ ]:
```
plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.data[0:10],digits.target[0:
 ↪10])):
    plt.subplot(1, 10, index + 1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title(label, fontsize = 20)
```



```
[ ]: # help(plt)
```

```
[ ]: # split the data
     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) # changing above data into x, y
```

```
[ ]: print("Trained input data:=", x_train.shape)
     print("Test input data:=", x_test.shape)
     print("Trained out put data:=", y_train.shape)
     print("Test out put data:=", y_test.shape)
```

```
Trained input data:= (1437, 64)
Test input data:= (360, 64)
Trained out put data:= (1437,)
Test out put data:= (360,)
```

```
[ ]: #train the model
     from sklearn.linear_model import LogisticRegression
     model= LogisticRegression().fit(x_train,y_train)
     model
```

```
C:\Users\Haier\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
[ ]: LogisticRegression()
```

[ ]:
```
model.predict(x_test[0:10])
predictions=model.predict(x_test)
predictions
```

[ ]:
```
array([2, 8, 2, 6, 6, 7, 1, 9, 8, 5, 2, 8, 6, 6, 6, 6, 1, 0, 5, 8, 8, 7,
       8, 4, 7, 5, 4, 9, 2, 9, 4, 7, 6, 8, 9, 4, 3, 1, 0, 1, 8, 6, 7, 7,
       1, 0, 7, 6, 2, 1, 9, 6, 7, 9, 0, 0, 9, 1, 6, 3, 0, 2, 3, 4, 1, 9,
       2, 6, 9, 1, 8, 3, 5, 1, 2, 8, 2, 2, 9, 7, 2, 3, 6, 0, 9, 3, 7, 5,
       1, 2, 9, 9, 3, 1, 4, 7, 4, 8, 5, 8, 5, 5, 2, 5, 9, 0, 7, 1, 4, 7,
       3, 4, 8, 9, 7, 9, 8, 2, 1, 5, 2, 5, 8, 4, 1, 7, 0, 6, 1, 5, 5, 9,
       9, 5, 9, 9, 5, 7, 5, 6, 2, 8, 6, 9, 6, 1, 5, 1, 5, 9, 9, 1, 5, 3,
       6, 1, 8, 9, 8, 7, 6, 7, 6, 5, 6, 0, 8, 8, 9, 8, 6, 1, 0, 4, 1, 6,
       3, 8, 6, 7, 4, 9, 6, 3, 0, 3, 3, 3, 0, 7, 7, 5, 7, 8, 0, 7, 1, 9,
       6, 4, 5, 0, 1, 4, 6, 4, 3, 3, 0, 9, 5, 9, 2, 1, 4, 2, 1, 6, 8, 9,
       2, 4, 9, 3, 7, 6, 2, 3, 3, 1, 6, 9, 3, 6, 3, 3, 2, 0, 7, 6, 1, 1,
       9, 7, 2, 7, 8, 5, 5, 7, 5, 2, 3, 7, 2, 7, 5, 5, 7, 0, 9, 1, 6, 5,
       9, 7, 4, 3, 8, 0, 3, 6, 4, 6, 3, 2, 6, 8, 8, 8, 4, 6, 7, 5, 2, 4,
       5, 3, 2, 4, 6, 9, 4, 5, 4, 3, 4, 6, 2, 9, 0, 1, 7, 2, 0, 9, 6, 0,
       4, 2, 0, 7, 9, 8, 5, 7, 8, 2, 8, 4, 3, 7, 2, 6, 9, 1, 5, 1, 0, 8,
       2, 8, 9, 5, 6, 2, 2, 7, 2, 1, 5, 1, 6, 4, 5, 0, 9, 4, 1, 1, 7, 0,
       8, 9, 0, 5, 4, 3, 8, 8])
```

[ ]:
```
## accuracy test
score=model.score(x_test,y_test)
print('the accuracy score id:', score)
```

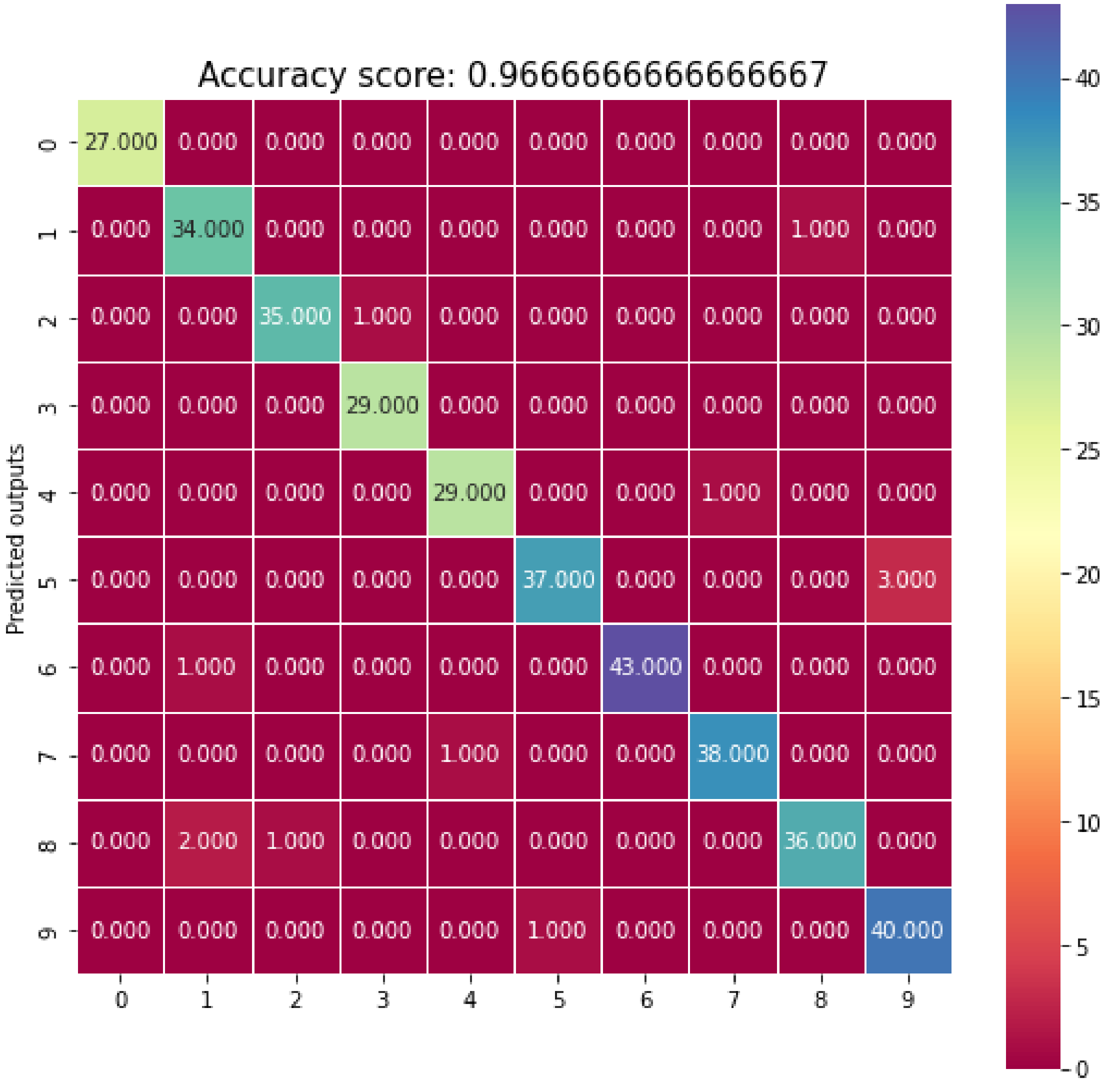
```
the accuracy score id: 0.9666666666666667
```

[ ]:
```
###confusion matrics
from sklearn import metrics
cm =metrics.confusion_matrix(y_test,predictions)
cm
```

[ ]:
```
array([[27,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 34,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  0, 35,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 29,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 29,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 37,  0,  0,  0,  3],
       [ 0,  1,  0,  0,  0,  0, 43,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 38,  0,  0],
       [ 0,  2,  1,  0,  0,  0,  0,  0, 36,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 40]], dtype=int64)
```

[ ]:
```
import seaborn as sns
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidth=.5, square=True,␣
 ↪cmap='Spectral');
```

```
plt.ylabel('Actual outputs');
plt.ylabel('Predicted outputs');
all_sample_title='Accuracy score: {0}'.format(score)
plt.title(all_sample_title, size=15);
```



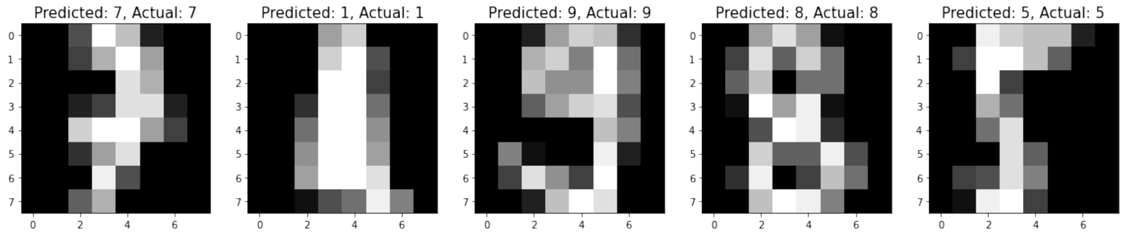
	0	1	2	3	4	5	6	7	8	9
0	27.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	34.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
2	0.000	0.000	35.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	0.000	29.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	0.000	29.000	0.000	0.000	1.000	0.000	0.000
5	0.000	0.000	0.000	0.000	0.000	37.000	0.000	0.000	0.000	3.000
6	0.000	1.000	0.000	0.000	0.000	0.000	43.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	1.000	0.000	0.000	38.000	0.000	0.000
8	0.000	2.000	1.000	0.000	0.000	0.000	0.000	0.000	36.000	0.000
9	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	40.000

[ ]:
```
print(cm)
```

```
[[27  0  0  0  0  0  0  0  0  0]
 [ 0 34  0  0  0  0  0  0  1  0]
 [ 0  0 35  1  0  0  0  0  0  0]
 [ 0  0  0 29  0  0  0  0  0  0]
 [ 0  0  0  0 29  0  0  1  0  0]
 [ 0  0  0  0  0 37  0  0  0  3]
 [ 0  1  0  0  0  0 43  0  0  0]
 [ 0  0  0  0  1  0  0 38  0  0]
 [ 0  2  1  0  0  0  0  0 36  0]
 [ 0  0  0  0  0  1  0  0  0 40]]
```

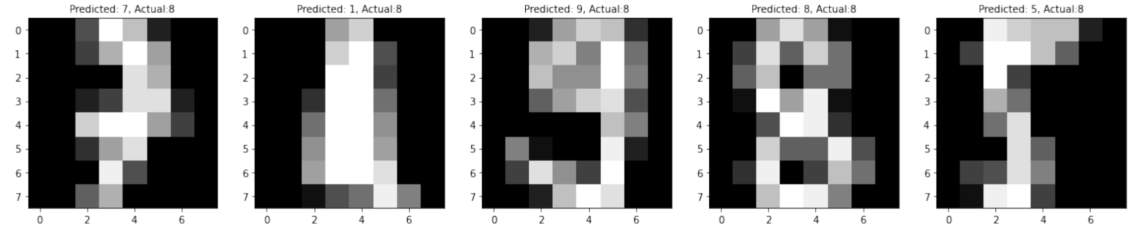
[ ]:
```
# getting missed classified labels
import numpy as np
import matplotlib.pyplot as plt
index= 0
misclassifiedIndexes = []
for label, predict in zip(y_test, predictions):
    if label != predict:
        misclassifiedIndexes.append(index)
        index +=1
```

[ ]:
```
# plotting misclassified labels with known labels
plt.figure(figsize=(20,4))
for plotIndex, badIndex in enumerate(misclassifiedIndexes[5:10]):
    plt.subplot(1, 5, plotIndex + 1)
    plt.imshow(np.reshape(x_test[badIndex], (8,8)), cmap=plt.cm.gray)
    plt.title("Predicted: {}, Actual: {}".format(predictions[badIndex],␣
 ↪y_test[badIndex]),fontsize=15)
```



2 Labelled Successfully

[ ]:
```
# plotting misclassified labels with known labels
plt.figure(figsize=(20,4))
for plotIndex, badIndex in enumerate(misclassifiedIndexes[5:10]):
    plt.subplot(1, 5, plotIndex + 1)
    plt.imshow(np.reshape(x_test[badIndex], (8,8)), cmap=plt.cm.gray)
    plt.title("Predicted: {}, Actual:{}".␣
 ↪format(predictions[badIndex],y_test[label] ,y_test[badIndex]),fontsize=10)
```


Predicted: 7, Actual:8

Predicted: 1, Actual:8

Predicted: 9, Actual:8

Predicted: 8, Actual:8

Predicted: 5, Actual:8