Faculty of Engineering
Cairo University

Cairo University

Faculty of Engineering

Systems and Biomedical Department

Machine Learning Assignments

Submitted by:

Ashar Seif Al-Naser Saleh

Sec: 1 BN: 9

Submitted to:

Eng/Christeen Ramsis

SBE452-AI

Dr. Inas A. Yassine

26 October, 2021

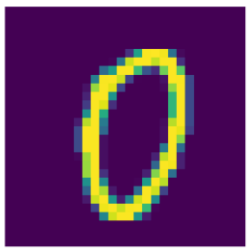
I. Show 5 examples from the used mnist dataset

Figures below shows some digits exist on the data set:

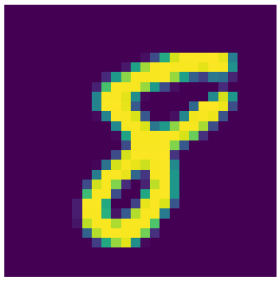
- **Y[0] → 5**

- **Y[150] → 4**

- **Y[777] → 0**



- **Y[1000] → 8**



- **Y[4000] → 7**

II. **The findings of the previous requirements.**

- **Binary Classifier for 3 or not 3 classes**
  [Returns true from the prediction function]

  - **Y[7] →3**

- **SGD Binary Classifier**

```
# Training a Binary Classifier
y_train_3 = (y_train == "3") # True for all 3s, False for all other digits.
y_test_3 = (y_test == "3")

# The SGD classifier has the advantage of being capable of handling very large datasets efficiently
sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
sgd_clf.fit(X_train, y_train_3)

# Get list index for a 3 in Y list
three_index=np.where(Y=="3")[0][0]
#print(three_index)   #index=7

# Choosing a 3 number from the X list and plot it
Random_digit=X[three_index]
Random_digit_image = Random_digit.reshape(28, 28)
plt.imshow(Random_digit_image, cmap=mpl.cm.binary)
plt.axis("off")
plt.savefig("DigitThree.png")
#plt.show()

# Making Sure it is the same at the Y list
#print(Y[7])       # 3

# Predict the 3 number using prediction function
print(sgd_clf.predict([Random_digit]))      # True

```

- **Multiclass Classification**
- **Predicts the 10 classes of the dataset**

```
sgd_clf.fit(X_train, y_train)
sgd_clf.predict(X_test)
```

III. Interpret the output of the confusion matrix

- **Calculating performance measures from confusion matrix**

```
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_3,
cv=3)

#Implement the confusion matrix
conf_matrix=confusion_matrix(y_train_3, y_train_pred)
TN=conf_matrix[0][0]   # True negative
FP=conf_matrix[0][1]   # False positive
FN=conf_matrix[1][0]   # False negative
TP=conf_matrix[1][1]   # True positive

accuracy =  (TP+TN) /(TP+FP+TN+FN)
precision = TP/(TP+FP)
sensitivity = TP/(TP+FN)
specifity = TN/(TN+FP)
```

- **Plotting the confusion matrix**