PRML Assignment Submitted at University of canberra

Logistic Regression for Fashion-MNIST images

1. **Introduction of the Datasets**

Fashion_MNIST is a dataset of images from Zalando's article which has 60,000 examples of training sets and 10,000 examples of test sets. The quality of the images is 28*28 pixels (28 in height and 28 in width) total of 784 pixels in total and has grayscale image. Each value of the pixel indicates the lightness or darkness of that pixel, so it means a higher number means darker. The pixel value is an integer which is between 0 and 255. In this dataset, every row has a separate image, and column 1 has the class label. Besides from column 1 rest of the columns are pixel numbers which has total of 784 and have value(darkness of the pixel) from 1 to 255.

2. **Problems to solve from this dataset**

In these datasets there are around 70, 000 images which need to be separate on the basis of 9 categories. So, the target is to develop a machine learning model using linear regression model to get the maximum accuracy predicting the accurate category. Also there are 9 labels from 0 to 9(T-shirt/top, Trouser, pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot)

3. **Research questions**

The question we will ask is whether the dataset is enough to train and test the model to get the maximum accuracy?

4. **Reasons to choose Logistic regression**

We have used logistic regression modelling because the dataset have 10 categories and this models easily handles multi-class classification tasks. Logistic Regression is an excellent choice for the Fashion-MNIST dataset due to its efficiency, interpretability, and ability to handle multi-class classification tasks. It serves as a strong baseline model, providing reliable and easily understandable results, making it suitable for answering questions related to image classification in this context.

Python constructs

1. **Retrieving the Data :** We have downloaded the dataset from the link (https://www.kaggle.com/datasets/zalando-research/fashionmnist). We save into our laptop and retrieve to the spyder.

 # Importing the packages

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import numpy as np
```

#Retreiving the data

```
Data_MIST_Fashion_Test = r"C:\Users\kkabi\OneDrive\Desktop\DAta science UC\patterntutorial\fashion-mnist_test.csv"
Data_MIST_Fashion_Train = r"C:\Users\kkabi\OneDrive\Desktop\DAta science UC\patterntutorial\fashion-mnist_train.csv"
```

2. **Exploring the datasets:** We have first combined both train and test dataset in one file. There are

```
# Combine test and train datasets
Test_train_combined_df_new = pd.concat([Data_MIST_Fashion_Test_df, Data_MIST_Fashion_Train_df], axis=0)
# Display column names
keys = Test_train_combined_df_new.keys()
print("Keys:", keys)
```

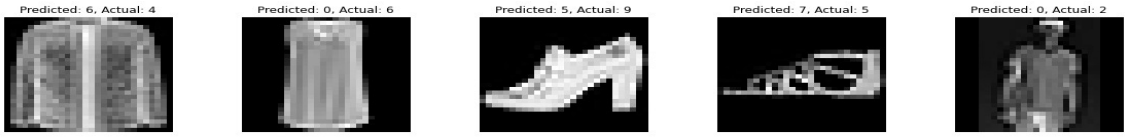
```
Keys: Index(['label', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       ...
       'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779', 'pixel780',
       'pixel781', 'pixel782', 'pixel783', 'pixel784'],
      dtype='object', length=785)
```

```
# Show first 5 labels
labels = Test_train_combined_df_new['label'][:5]
print("Labels:", labels)
```

```
Labels: 0    0
1    1
2    2
3    2
4    3
Name: label, dtype: int64
```

Show the image and data

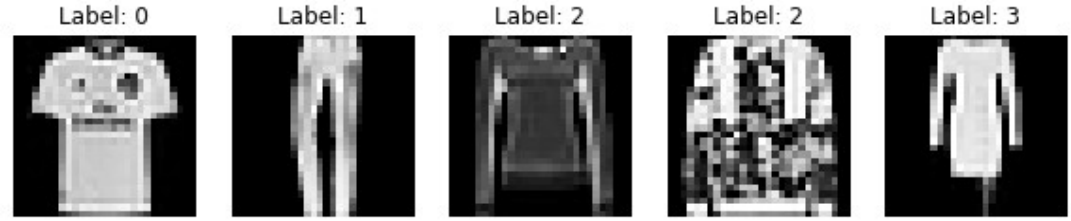
```
# Define the number of images to display
number_images = 5
plt.figure(figsize=(10, 2))
for i in range(number_images):
  # Extract the image data and label
 image_data = Test_train_combined_df_new.iloc[i, 1:785].values
    label = Test_train_combined_df_new.iloc[i]['label']
```



Note: Also there are 9 labels from 0 to 9(0 is T-shirt/top, 1 is Trouser, 3 is pullover, 4 *isDress,5 is Coat, 6 is Sandal, 7 is Shirt, 8 is Sneaker, 9 is Bag,10 is Ankle boot.*

```
 image_data = image_data.reshape(28, 28)
   plt.subplot(1, number_images, i + 1)
   plt.imshow(image_data, cmap='gray')
   plt.title(f"Label: {label}")
   plt.axis('off')
```

plt.show()



Note: Also there are 9 labels from 0 to 9(0 is T-shirt/top, 1 is Trouser, 3 is pullover, 4 isDress,5 is Coat, 6 is Sandal, 7 is Shirt, 8 is Sneaker, 9 is Bag,10 is Ankle boot.

3. Building a Logistic Regression Model

Show the corresponding matrix

```
# Prepare labels (y) and features (X)
y = Test_train_combined_df_new['label']
X = Test_train_combined_df_new.drop('label', axis=1)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,
random_state=0)

# Train Logistic Regression model
lr = LogisticRegression(solver="lbfgs", max_iter=100)
lr.fit(X_train, y_train)
```

```
C:\Users\kkabi\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: 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(
```

Predict on unseen data

```
# Make predictions
y_pred = lr.predict(X_test)
```

```
score_result = lr.score(X_test, y_test)
print("Score:", score_result)
```

```
Score: 0.8495
```

The score is 0.8495 , which is accuracy which seems very low.

4. Analysis of results-classification report

Confusion matrix

```
# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)
```

```
Confusion Matrix:
 [[1168    8   22   65    5    1  131    0   19    2]
 [   2 1381    9   34    5    0    5    0    1    0]
 [  30    6  991   14  203    0  113    0   20    0]
 [  43   21   16 1216   57    0   34    0    5    0]
 [   3    3   99   34 1101    0  106    0    7    0]
 [   2    0    0    0    0 1286    0   79    8   55]
 [ 198    4  201   36  143    0  803    0   22    1]
 [   0    0    0    0    0   28    0 1307    3   54]
 [   4    3    8   13    5   14   20    7 1342    1]
 [   0    0    0    0    0   30    0   43    2 1298]]
```

```
# Classification report
class_report = classification_report(y_test, y_pred)
print("Classification Report:\n", class_report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.82      0.81      1421
           1       0.97      0.96      0.96      1437
           2       0.74      0.72      0.73      1377
           3       0.86      0.87      0.87      1392
           4       0.72      0.81      0.77      1353
           5       0.95      0.90      0.92      1430
           6       0.66      0.57      0.61      1408
           7       0.91      0.94      0.92      1392
           8       0.94      0.95      0.94      1417
           9       0.92      0.95      0.93      1373

    accuracy                           0.85     14000
   macro avg       0.85      0.85      0.85     14000
weighted avg       0.85      0.85      0.85     14000
```

Accuracy of this machine learning model is 0.85 which seems very low.

```
# Display first 5 images with predictions
plt.figure(figsize=(10, 2))
for idx in range(5):
    image_data = X_test.iloc[idx, :].values
 prediction = y_pred[idx]
    plt.subplot(1, 5, idx + 1)
    plt.axis("off")
```

```
    plt.imshow(image_data.reshape(28, 28), cmap=plt.cm.gray_r,
interpolation="nearest")
    plt.title(f"Prediction: {int(prediction)}")
plt.show()
```

Note: Also there are 9 labels from 0 to 9(0 is T-shirt/top, 1 is Trouser, 2 is pullover, 3 isDress, 4 is Coat, 5 is Sandal, 6 is Shirt, 7 is Sneaker, 8 is Bag and 9 is Ankle boot.

Display Misclassified images with predicted Labels

```
# Identify misclassified indexes
misclassifiedIndexes = []
for index, (label, predict) in enumerate(zip(y_test, y_pred)):
    if label != predict:
        misclassifiedIndexes.append(index)

# Display first 5 misclassified images
plt.figure(figsize=(20, 3))
for plotIndex, badIndex in enumerate(misclassifiedIndexes[0:5]):
    plt.subplot(1, 5, plotIndex + 1)
    plt.axis("off")
    plt.imshow(np.array(X_test.iloc[badIndex, :]).reshape(28, 28), cmap=plt.cm.gray,
interpolation='nearest')
    plt.title(f'Predicted: {y_pred[badIndex]}, Actual: {y_test.iloc[badIndex]}',
fontsize=12)
plt.show()
```

5. Concept of regularisation and future

Regularisation is used to prevent the overfitting in logistic regression. It penalizes large coefficients which helps to simplify models to better generalise. Therefore in this datasets there are also high dimensional datasets so regulization will also help in this model to prevent overfitting. This model can be used to compare with better models that can be made also to predict the images that we collected in future.