Image Classification Challenge

This is the documentation of my work for the given task- which is develop an AI Model for classifying Image of fashion items.

To develop this Model, I used the following-

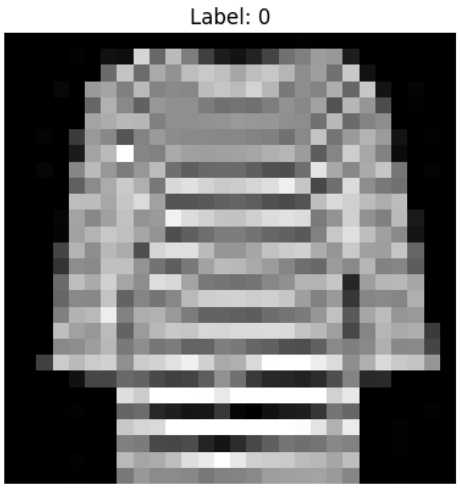
- Python Programming Language
- Tensorflow module
- Mathplotlib
- Scikit learn
- Numpy
- VS Code and Jupyter Notebook

Table Of Contents

Getting an Idea about the Dataset

- The given dataset is a csv file which contains 60,000 rows and 785 columns, in which one column is the label and others are values ranging from 0 to 255.
- Each cell represents a pixel having an alpha value. Here 0 means black and 255 means white.
- The images are 2d but their pixel values are stored in 1D array, hence we will reshape it later in the code.

- Their 10 labels, values ranging from 0 to 9, each value represent a category of the image.
- Here is one sample image generated by the code:-

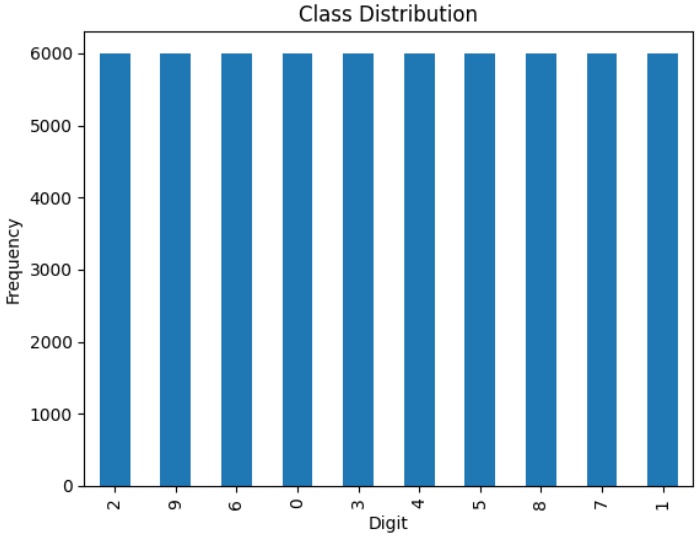


This is achieved by using the function **imshow()** an by reshaping the 1D array of each row into 2D array of 28x28 shape.

- So, it's clear that the data set contains info of 60,000 grayscale images (28x28 pixels) of fashion items.

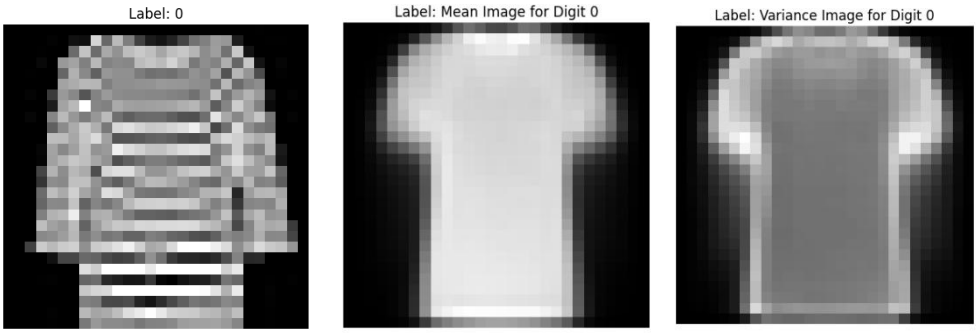
Exploratory Data Analysis

- First we will plot a bar chart where y-axis is the count and x is the category, this is to check if the dataset is balanced or not.



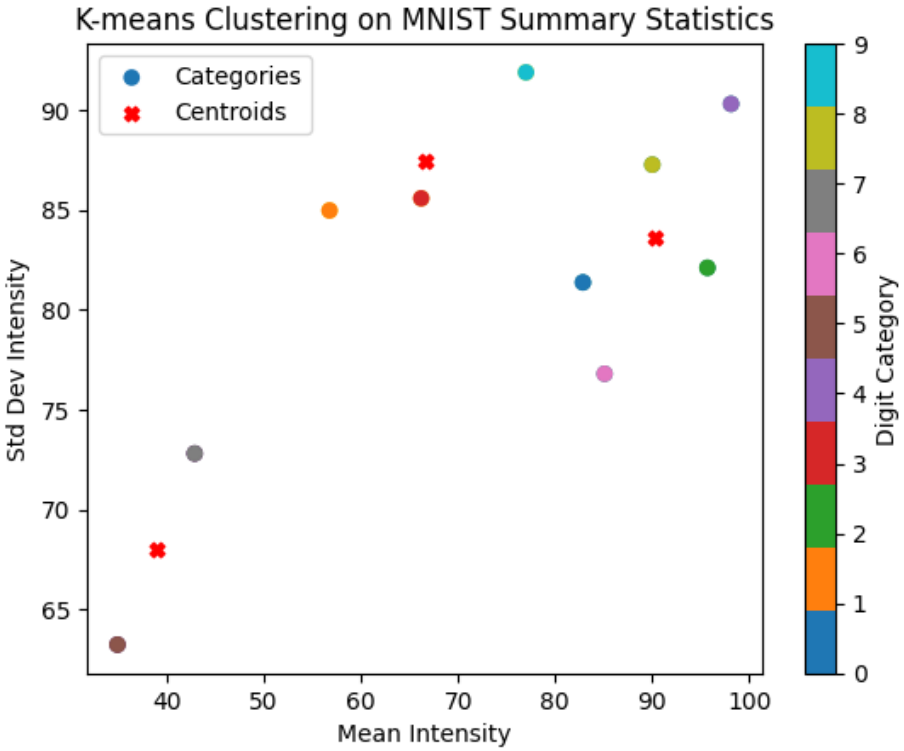
From the above bar chart its clear that the dataset is balanced. This above chart was plot using **series.plot(kind="bar")** .

- Then we will plot random, mean and variance image of each category item, for example-



The 1st is a random image of category 0.
The 2nd is mean image of category 0 giving us an idea of how all images 0 looks like.
The 3rd one is the variance image of category 0 as we can see the inner side is dark which means low variations in values of pixels and the outer side is white which means high variations in values of pixels.

- Then we will compute mean_ intensity and std_intensity of each category and by that this we will plot K-means Clustering, to get the idea of how each category differs from each other.



This gives an idea how each category differs from each other like for example category 0 is a tshirt and category 1 is pant , from image and the visualization its very clear that they both are very different from each other. To achieve the following plot I used **KMeans(), plt.scatter(),plt.colorbar().**

Basic Classification Model (using logistic regression)

- Using multinomial logistic regression I will develop a simple classifier.
- First we will normalize the values of pixels that is to convert them in to numbers b\w 0 and 1, as it improves performance of the model. This is achieved by **StandarScaler.transform(data set of pixels).**
- And then we will also split the dataset into training and testing sets in the ratio of 4:1 .
- After this we create a Logistic Regression model with the argument multiclass=multinomial(as there are more than 2 categories).
- After creation of the model , we train it by giving training data set as arguments and then make predictions on the test data set.

  To achieve all of this I used Logistic Regression class from sklearn.linear_model .
- The accuracy of the model is 84.26% .

Accuracy: 84.26%

	precision	recall	f1-score	support
0	0.78	0.81	0.79	1232
1	0.95	0.96	0.96	1174
2	0.76	0.75	0.75	1200
3	0.84	0.86	0.85	1242
4	0.73	0.76	0.75	1185
5	0.92	0.93	0.92	1141
6	0.65	0.57	0.61	1243
7	0.92	0.93	0.92	1224
8	0.94	0.93	0.93	1149
9	0.94	0.94	0.94	1210
accuracy			0.84	12000
macro avg	0.84	0.84	0.84	12000
weighted avg	0.84	0.84	0.84	12000

Constructing and training a simple neural network

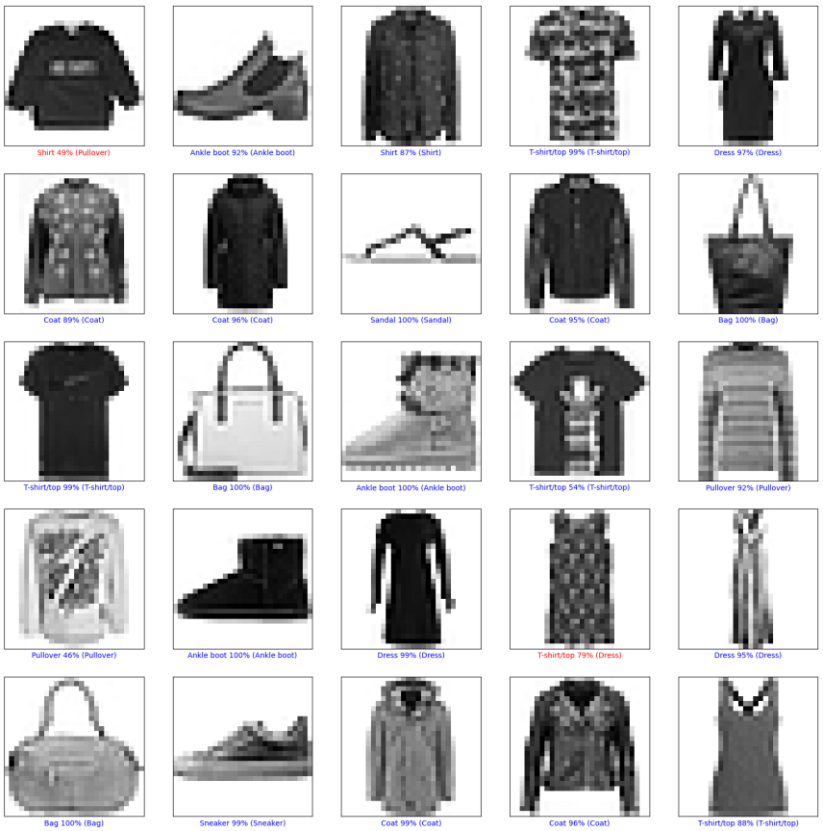
- The above model was based on logistic regression which has an accuracy of 84%, we can get a better accuracy by implementing a simple neural network.
- We create a simple neural network model by writing this code snippet:

  model = Sequential([Flatten(input_shape=(28, 28)), Dense(128, activation='relu'), Dense(10, activation='softmax')])

  1. **Flatten is the input layer it converts 28x28 images into a 1D array of 784.**
  2. **Dense(128,activation='relu') is the hidden layer which is fully connected with 128 neurons and ReLU activation function.**
  3. **Dense(10, activation='softmax') is the output layer which is fully connected with 10 neurons, where each neuron corresponds to one class.**

  To get this we use tensorflow.keras

- After this we compile and train the model using **model.compile() and model.fit() .**
- And then we make prediction and plot a sample output image of predictions.



- Here blue color font means that the model has predicted right and red means it has predicted wrong. Where has the percentage means the confidence the model has on its prediction.
- The overall accuracy of this model is around 91% .