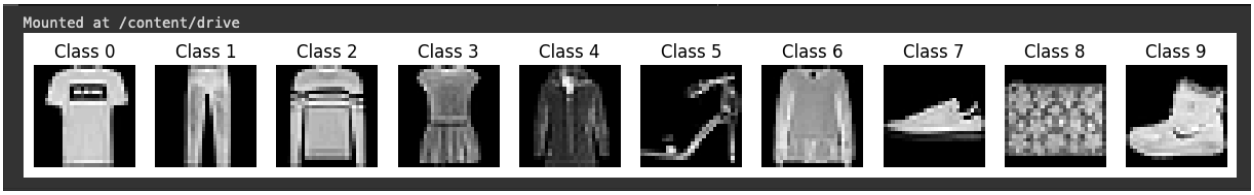
Fashion MNIST
Choosing best algorithm for splitting dates

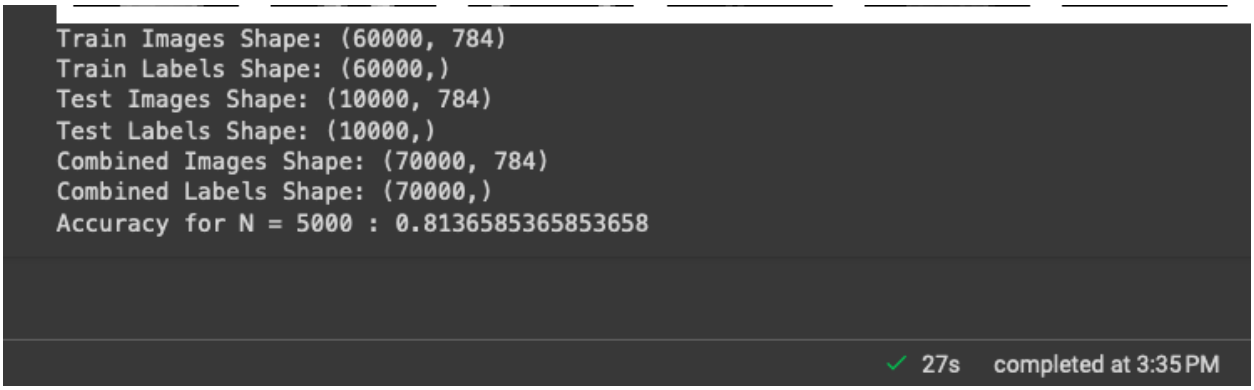
Name: Swathi Baskaran

1. Fashion MNIST dataset by Zalando Research consists of 70,000 grayscale images, each of size 28x28 pixels, representing various fashion items, such as shirts, dresses, shoes, and handbags. These images are divided into 60,000 training samples and 10,000 test samples, with each sample labeled with one of 10 categories. The classes being T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

Here is the screenshot of the first data from each class.



2. I realized we will be dealing with quite a large amount of data. I decided to work on the project with python. I performed a systematic exploration of N values ranging from 1000 to 6000. I first executed the classifier with N=5000, representing a balanced split of 5000 samples for training and 2000 (7000-N) samples for testing to check the run time. This run was executed in 27 secs. This initial run provided valuable insights into the baseline performance and computational overhead.

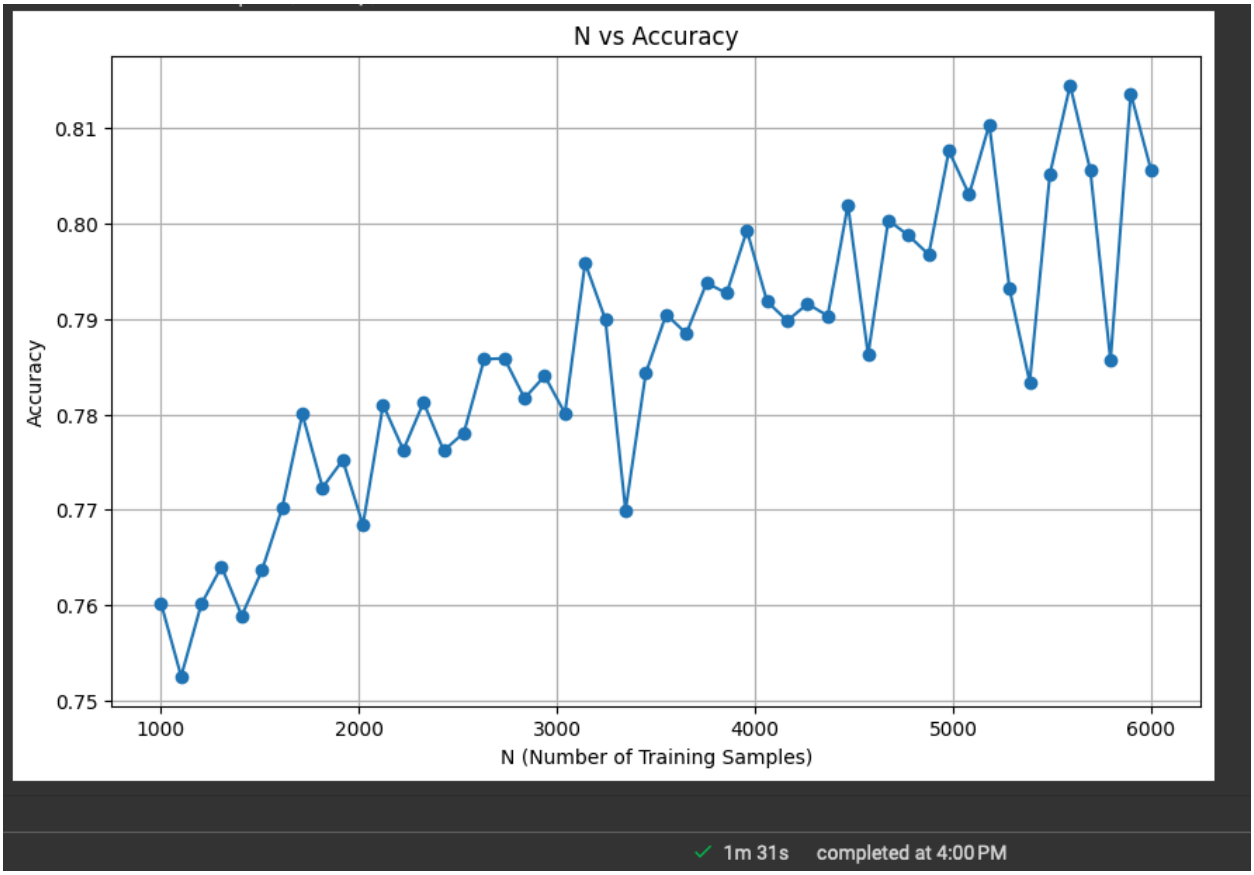


I divided this range into intervals of 1000, that is 6 N values, and ran the classifier for six values within each interval. This execution took about 57 secs to complete. Observations from this step indicated a trend of diminishing accuracy for N values below 3000, suggesting insufficient training data.

```
Train Images Shape: (60000, 784)
Train Labels Shape: (60000,)
Test Images Shape: (10000, 784)
Test Labels Shape: (10000,)
Combined Images Shape: (70000, 784)
Combined Labels Shape: (70000,)
Chosen N values: [1000, 2000, 3000, 4000, 5000, 6000]
Accuracies: [0.7468692603105693, 0.7760717846460619, 0.7842023748064016, 0.796943972835314, 0.805228088159918, 0.768]
Times: [2.4799554347991943, 2.4328348636627197, 1.7310552597045898, 1.0058858394622803, 0.7476711273193359, 0.42720890045166016]

✓ 57s completed at 3:52 PM
```

To validate this observation, I conducted a more comprehensive experiment with 50 random N values within the specified range. This experiment confirmed that N values below 3000 yielded lower accuracy due to the limited amount of training data available. This experiment took 91 secs (1 min 31 secs) to run.



Based on the findings, I narrowed down the search space for the optimal N to the range of 3000 to 6000. The reason being that focusing on this narrower range would allow for a more targeted exploration without compromising accuracy.

To begin the analysis, I primarily wanted to investigate using 1000 different values of N. If I can find an optimal N, I would pick it. This should take about 16mins. To thoroughly explore this range, I decided to run the test for 3000 different N values in the range 3000 - 6000.

Considering the computational cost and feasibility, I estimated that this approach would require approximately one hour of computation time. With our workflow allowing for multiple runs per day, we anticipated completing the experiment within 10 runs that should approximately take 10-11 hours. This two-step approach allowed for an efficient exploration of the parameter space while ensuring adequate coverage for finding the most suitable N. 3000 runs is a good amount and is feasible to find an optimal N value for the splitting of test and train data.

	Time taken for execution	**Expected time for execution**
N = 5000	27 secs	
N = 1000, 2000, 3000, 4000, 5000, 6000 (6 different values)	57 secs	
N = (50 different values in range 1000 - 6000)	91 secs	
N = (1000 different values in range 3000 - 6000)		950 secs = 16 mins
N = (3000 different values in range 3000 - 6000)		2850 secs = 48 mins

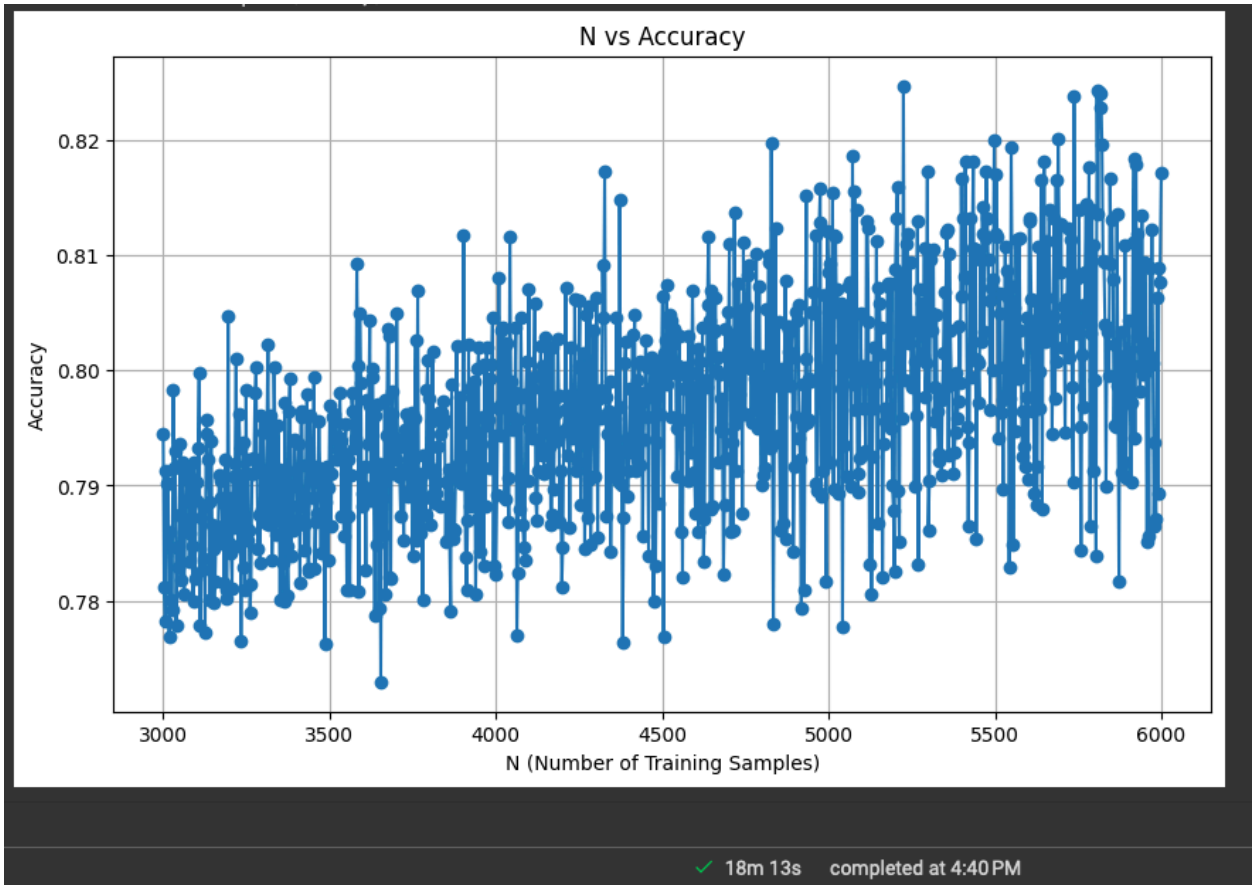
3. **Random Sampling:**

I decided to use random sampling to split train and test data. It is a widely used technique for splitting datasets. Random sampling is a statistical sampling technique in which each element from the population has an equal chance of being selected within the sample. It splits the dataset by generating random individual indices. It operates on a class-wise basis, ensuring that each class is adequately represented in both the training and testing sets. By shuffling the indices of images belonging to each class and then selecting a specified number of indices for training and the remainder for testing, ensures that the selection process is unbiased and does not favor any specific subset of the data.

Here it randomly samples N indices from the range [0, 6999] without replacement. This ensures that each index is selected only once and that the selected indices are unique. This algorithm ensures that N out of the total 7000 images are selected randomly for the training set, while the remaining images are assigned to the test set. This approach fosters diversity in the training and testing data, promoting robust model performance evaluations.

After examining 1000 values of N to understand the relationship between training set size and classification accuracy, it is clear that N values below 3000 yielded lesser accuracies,

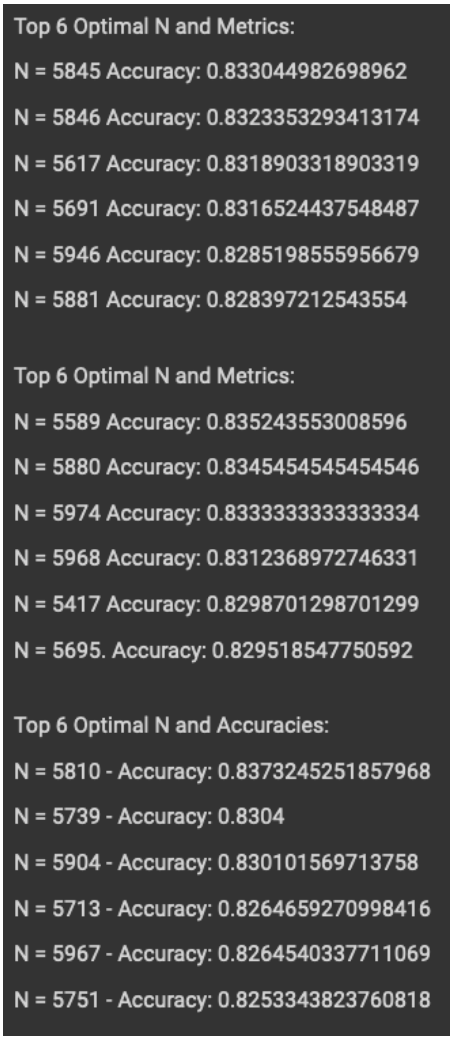
underscoring the importance of sufficient training data for effective model learning. However, N values above 3000 consistently produced accuracies a little over 80%, indicating the potential benefits of larger training sets. So I decided to drop the N values below 3000.

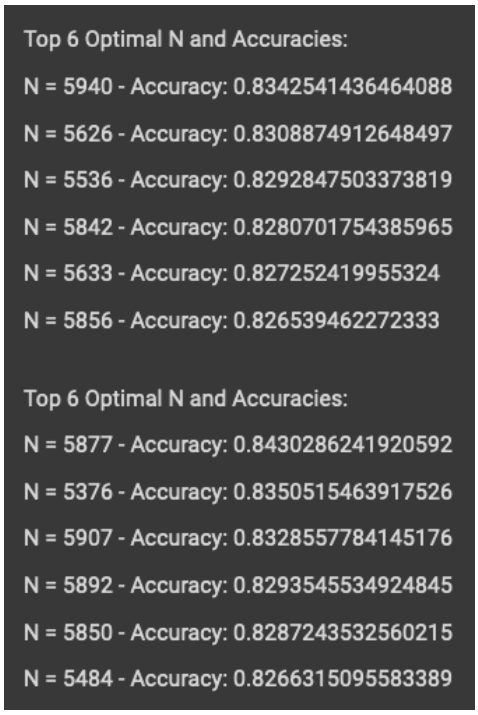


After conducting an extensive analysis of 3000 values of N for our task, it became evident that a substantial number of these values yielded accuracies within the range of 83% to 84%. This finding prompted a thoughtful consideration of the optimal training set size, crucial for ensuring effective model learning and performance.

In light of this, I rounded off these accuracies to 83%, corresponding to an N value of 5810, and adopted an 83:17 ratio for the training and testing sets. This decision was informed by the recognized importance of allocating a larger proportion of data to the training set, enabling the model to grasp meaningful patterns and generalize well to unseen instances. Meanwhile, the smaller testing set size ensured reliable performance estimation, striking a balance between model performance and evaluation.

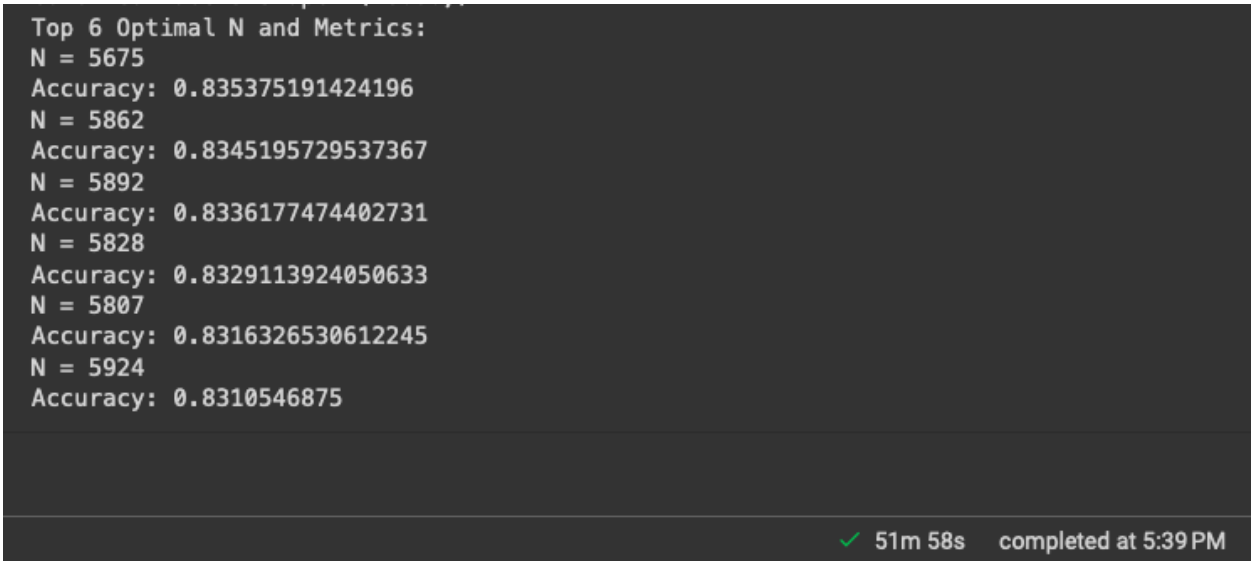
Some outputs of the runs to find optimal run.





Furthermore, the emergence of N=5810 with the higher accuracy during the runs underscored its suitability as the optimal training set size. By embracing a comprehensive approach that integrates empirical evidence, computational considerations, and the principles of random sampling, it was possible to establish a balanced framework for model performance evaluation. This approach not only reaffirmed the fundamental role of training set size in influencing classification scheme performance but also provided valuable insights into achieving a harmonious balance between model performance and computational efficiency.

4. **Kennard-Stone algorithm**



The Kennard-Stone (KS) algorithm, offers a methodical approach to data partitioning, particularly advantageous in machine learning for training and testing set splitting. This algorithm aims to create two subsets with distinct characteristics, emphasizing representative sampling and optimal partitioning. Initially, the entire dataset is considered, and pairwise distances between data points are calculated. Subsequently, seed points are chosen as initial representatives for the two subsets, followed by iterative expansion to include data points closest to the existing subset members. The algorithm ensures each data point is assigned to one of the subsets based on its proximity, leading to a finalized partitioning of the dataset. Notably, the KS algorithm prioritizes representative sampling to avoid biases and optimal partitioning to maximize subset characteristics, making it a valuable tool for tasks requiring diverse and balanced training and testing sets.

Upon comparing random sampling and the Kennard-Stone (KS) algorithm, some significant conclusions are:

Unbiased Data Distribution: Random Sampling: While straightforward, random sampling may result in uneven class distributions between training and testing sets, potentially introducing bias. KS Algorithm: By class-wise sampling and ensuring balanced representation of classes in both sets, the KS algorithm mitigates the risk of bias and promotes more reliable model evaluation.

Model Generalization: Random Sampling: The discrepancy in class distributions between training and testing sets can lead to overfitting or underfitting, impacting model generalization. KS Algorithm: Through representative sampling and preserving class distribution, the KS algorithm facilitates better model generalization to unseen data, enhancing performance in real-world scenarios.

Optimal Training Set Size: Random Sampling: Lacks consideration of the importance or relevance of data points, potentially leading to suboptimal training set sizes. KS Algorithm: Allows for fine-tuning of the training set size, ensuring sufficient representation of each class for effective model training, thereby enhancing performance and generalization.

Performance Evaluation: The Kennard-Stone algorithm yields higher accuracy compared to random sampling, indicating better overall performance. Precision measures the proportion of correctly predicted positive cases out of all cases predicted as positive. Recall measures the proportion of correctly predicted positive cases out of all actual positive cases. The F1 score is the harmonic mean of precision and recall, providing a balance between them. The Kennard-Stone algorithm shows higher precision, higher recall, higher F1Score, potentially leading to more reliable and accurate predictions across all classes. The confusion matrix provides insights into the distribution of true positive, true negative, false positive, and false negative predictions for each class. It's apparent that the KS algorithm's confusion matrix shows fewer misclassifications and a more balanced distribution across classes compared to random sampling.

While random sampling offers simplicity, the Kennard-Stone algorithm provides a systematic and reliable approach to data splitting.

```
Metrics for N=5810 with Random Sampling:
Accuracy: 0.8112118713932399
Precision: 0.8151063983597929
Recall: 0.8112118713932399
F1 Score: 0.8100374218443135
Confusion Matrix:
 [[108   0   2   6   0   0  11   0   1   0]
 [  1 126   4   1   1   0   2   0   0   0]
 [  2   0  89   0  19   0  16   0   0   0]
 [  4   0   2 114   1   0   3   0   0   0]
 [  0   0  18   5  86   0  14   0   0   0]
 [  0   0   0   0   0  81   0  19   1  11]
 [ 29   0  12   2  14   0  45   0   3   0]
 [  0   0   0   0   0   2   0 108   0   6]
 [  1   0   4   2   3   0   4   0 104   1]
 [  0   0   0   0   0   0   0   2   0 123]]
```

```
Metrics for N=5810 with Kennard-Stone algorithm:
Accuracy: 0.8489075630252101
Precision: 0.8520561743238559
Recall: 0.8489075630252101
F1 Score: 0.8481125763661553
Confusion Matrix:
 [[1073    1   24   16    5    0   64    1    6    0]
 [   5 1129   12   35    0    0    8    0    1    0]
 [  12    0  936    5  113    0  121    0    3    0]
 [  58    9   13 1022   51    0   36    0    1    0]
 [   7    1  103   33  920    0  122    0    4    0]
 [   1    0    0    2    0  954    8  142    6   77]
 [ 237    0  161   18   93    0  672    1    8    0]
 [   0    0    0    0    0    4    0 1122    0   64]
 [   5    0   10   10    9    0   15   12 1126    3]
 [   0    0    0    2    0    1    1   38    0 1148]]
```

5. a) L1 Norm (Manhattan Distance): The L1 norm calculates distance by summing the absolute differences along each dimension. It emphasizes differences along each dimension equally, leading to decision boundaries aligned with the axes of the feature space. Due to its robustness to differences in feature scales and alignment with the axes, the L1 norm is suitable for datasets where features have varying scales or linear relationships. Manhattan distance is generally more robust to outliers and noise due to its emphasis on differences along each dimension.

Euclidean Distance: Euclidean distance computes the straight-line distance between two points in a multi-dimensional space. It assumes equal contributions from all dimensions and can capture complex relationships between data points. Euclidean distance may be sensitive to differences in feature scales and outliers, potentially leading to biased results in datasets with varying feature scales. However, Euclidean distance can be sensitive to outliers, especially if features have significantly different scales.

The L1 norm is suitable for datasets with varying feature scales or linear relationships, as it provides a more interpretable model with decision boundaries aligned with the axes of the feature space. In contrast, the Euclidean distance may capture more complex relationships

between data points and is suitable for datasets where all dimensions contribute equally to the similarity measure. The accuracy with Manhattan distance (85.34%) is slightly higher than the accuracy with Euclidean distance (84.66%). This suggests that for the given dataset and problem, the Manhattan distance metric performs slightly better than the Euclidean distance metric.

```
Accuracy with Manhattan Distance: 0.8533613445378151
Accuracy with Euclidean Distance: 0.8465546218487395
```

5. b) Below is the screenshot of the performance metrics of each class.

```
Class: 0
Precision: 0.7675250357653791
Recall: 0.9016806722689076
F1 Score: 0.8292117465224113

Class: 1
Precision: 0.9903508771929824
Recall: 0.9487394957983193
F1 Score: 0.9690987124463519

Class: 2
Precision: 0.7434471803018269
Recall: 0.7865546218487395
F1 Score: 0.7643936300530829

Class: 3
Precision: 0.8941382327209099
Recall: 0.8588235294117647
F1 Score: 0.8761251607372482

Class: 4
Precision: 0.7724601175482787
Recall: 0.773109243697479
F1 Score: 0.7727845443091139

Class: 5
Precision: 0.9947862356621481
Recall: 0.8016806722689076
F1 Score: 0.8878548161935783
```

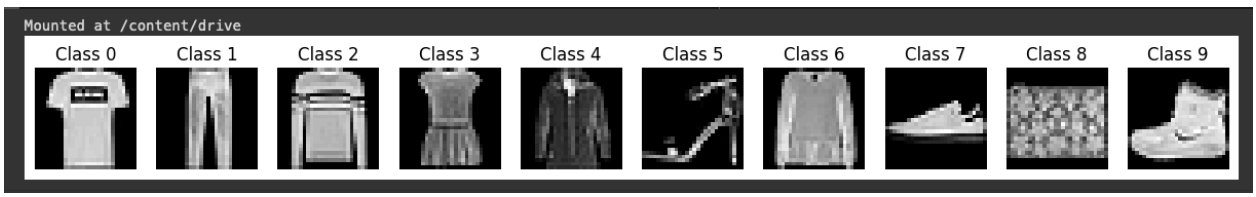
```
Class: 6
Precision: 0.6418338108882522
Recall: 0.5647058823529412
F1 Score: 0.6008046490835941

Class: 7
Precision: 0.8525835866261399
Recall: 0.9428571428571428
F1 Score: 0.8954509177972865

Class: 8
Precision: 0.9748917748917749
Recall: 0.946218487394958
F1 Score: 0.9603411513859276

Class: 9
Precision: 0.8885448916408669
Recall: 0.9647058823529412
F1 Score: 0.9250604351329573

Global Metrics:
Accuracy: 0.8489075630252101
Precision: 0.8520561743238559
Recall: 0.8489075630252101
F1 Score: 0.8481125763661553
```



The classes being T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

Classes 1, 5, 8, and 9 have high precision, recall, and F1-score, indicating excellent performance in classification. Class 1 has the highest precision (0.990) and a very high recall (0.949), indicating that pants are the easiest class to predict/classify among all the other items. Similarly sandals, bag, and ankle boots are easier to identify that the others. The model performs better on these classes that are visually distinct and easier to differentiate.

Classes 0, 3, and 7 also have relatively high precision, recall, and F1-score, showing good performance. These as well are are relatively easier to classify.

Classes 2, 4, and 6 have lower precision, recall, and F1-score compared to other classes, indicating poorer performance in classification. Class 6 has the lowest precision (0.642) and recall (0.565), indicating shirts are the most difficult to predict accurately. These classes have more ambiguity or similarity in appearance.

There are noticeable differences in performance between individual classes. Classes with higher precision, recall, and F1-score (e.g., 1, 5, 8, 9) are likely easier to classify, possibly due to clearer patterns or less variability within those classes. Classes with lower precision, recall, and F1-score (e.g., 2, 4, 6) may be more challenging to classify, possibly due to similarities with other classes or inherent complexity in distinguishing them.

Global Performance:

The overall accuracy of the model is 0.849, which indicates that it correctly predicts the class labels for approximately 85% of the samples in the dataset. The precision, recall, and F1 score are consistent across different classes, with an average precision of 0.852, recall of 0.849, and F1 score of 0.848. Improving the model's ability to distinguish between shirts, t-shirt/top and other similar classes could enhance overall performance.

Link to code :
https://colab.research.google.com/drive/1CY4tVx27ReRV5iu76W3DmIWMakmBLsns?usp=drive_link