An Investigation of Classification Methods for Fashion-MNIST

Kaiyuan Hu: 73838241

Chi Zhang: 25702053

Shengyuan Lu: 93188958

Summary

In this project, we investigated the application of various classification methods, including K-Nearest Neighbors (KNN), Logistic Regression, Feed-Forward Neural Networks (FNN), and Decision Tree, on the Fashion-MNIST dataset. Our results reveal that the FNN classifier performed the best among all four classifiers. Furthermore, we noted that all classifiers have challenges distinguishing visually similar fashion categories.

Data Description

The Fashion-MNIST is a large dataset with fashion images. There are in total 60,000 samples in the training dataset where each class label has equally 6,000 data points, and a total of 10000 images in the testing dataset where each class label has 1,000 data points (Fig. 1.1). This uniform distribution suggests that each feature in the datasets carries an equal influence on label prediction.

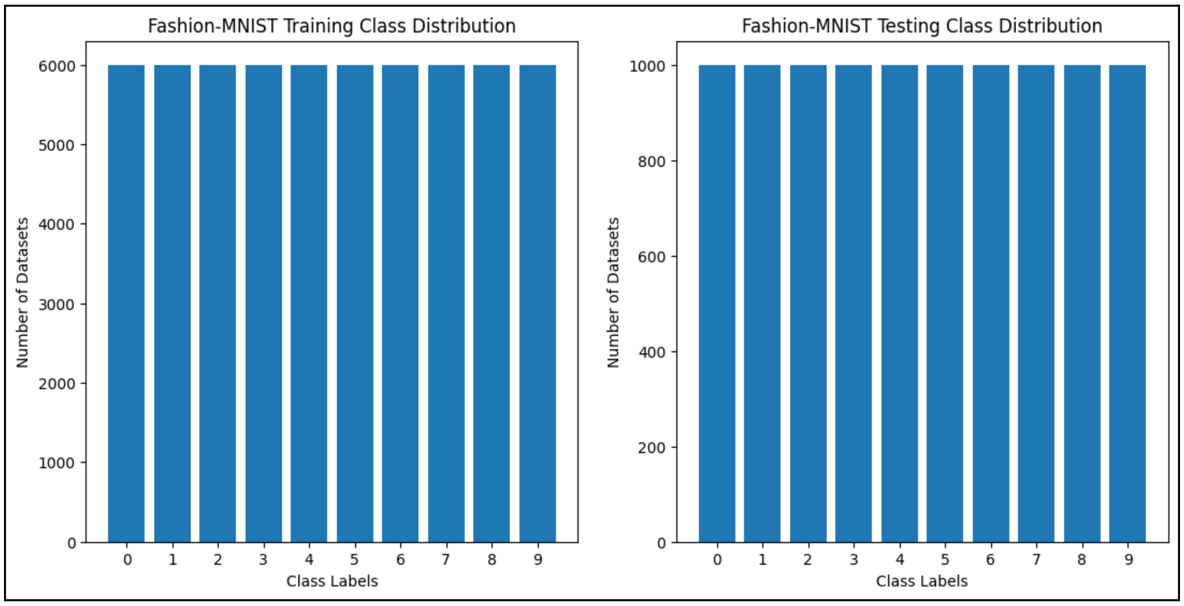


Figure 1.1. class distribution of Fashion-MNIST training and testing Dataset

The dataset has 784 features and each feature corresponds to a pixel in the 28x28 grayscale image that belongs to one of the 10 fashion items. In order to better understand the complexity and visualization perspective of each item, we visualized one sample for each fashion label. As shown by Figure 1.2, the classes in the dataset include T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots.

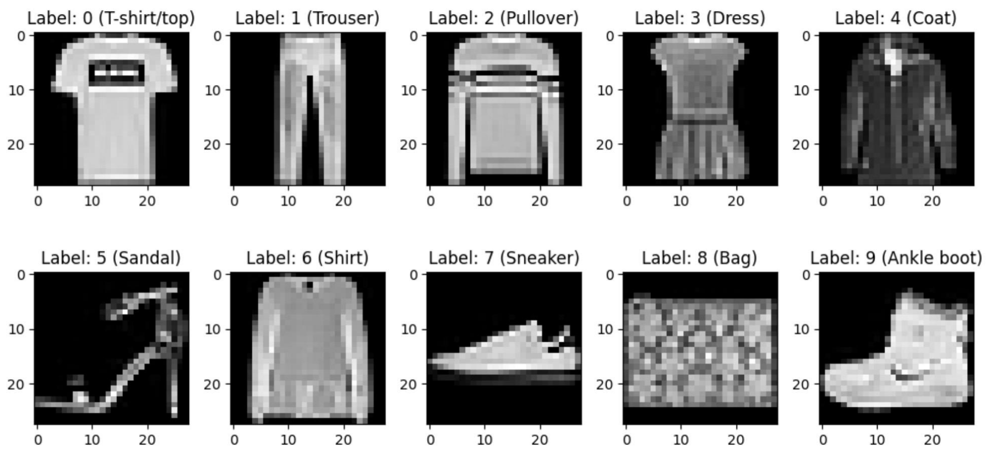


Figure 1.2. Visual demonstration of fashion labels, one sample from each class is shown

Classifiers:

In this project, all classifiers are implemented using scikit-learn to ensure consistency in methodology. Furthermore, to determine the optimal hyperparameters for all classifiers, a hybrid approach involving grid search and manual hyperparameter tuning was used on 75% of the training data then validated by the rest 25%.

K-Nearest Neighbors (KNN):

KNN classifier is a classification algorithm that classifies new data points by estimating the majority labels of its k-nearest training data points. For the KNN classifier, the primary hyperparameter is the n_neighbors, which is the number of neighbors to consider when determining the label for the new datapoint. Before tuning the n_neighbors for KNN classifier, we performed grid search with a selected range of values (Fig. 2.1 - Row 1) to obtain the preliminary setup.

weights	leaf_size	p
uniform, distance	10~50 with a step size of 5	1, 2, 3
uniform	45	1

Figure 2.1. The selected range of values for each hyperparameter

After getting the preliminary setup for other hyperparameters (Fig. 2.1 - Row 2), we tune the primary hyperparameter n_neighbors range from 1~40 with a step size of 2 to avoid even numbers that could result in a tie between data points' distances.

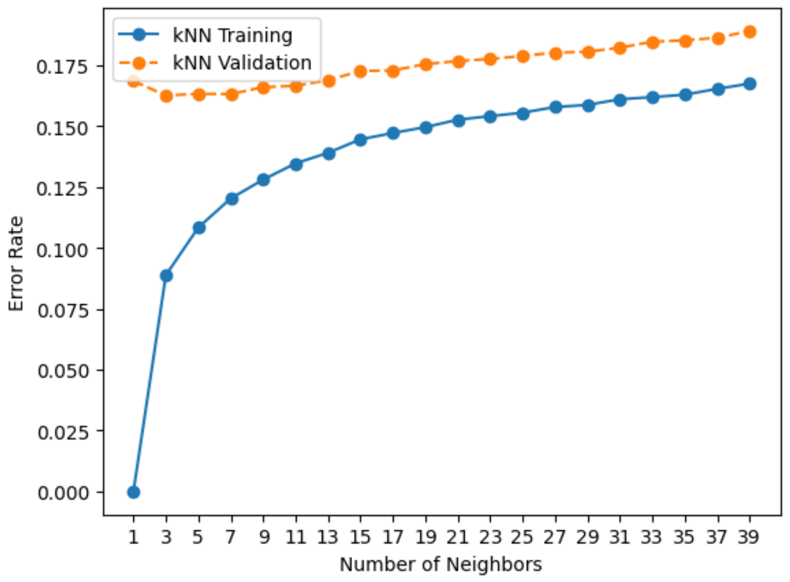


Figure 2.2. Error rate of KNN classifier with different n_neighbors

Based on Figure 2.2, both validation and training error rate increase after the threshold n_neighbors=3, indicating the model is moving toward underfitting. Hence, n_neighbors=3 will be selected for the classifier.

Logistic Regression:

Logistic Regression is a classification algorithm that classifies new data points by estimating the weighted probabilities of falling into a particular class. For the logistic classifier, one primary hyperparameter is C, the inverse of regularization strength. Smaller C means stronger regularization. Before tuning C for the logistic classifier, we performed grid search with a selected range of hyperparameters (Fig. 2.3 - Row 1) to obtain the preliminary setup.

penalty	solver	max_iter
l1, l2	lbfgs, newton-cg, sag, saga	100, 1000, 10000
l2	sag	100

Figure 2.3. The selected range of values for each hyperparameter of the logistic classifier

After getting the preliminary setup for other hyperparameters (Fig. 2.3 - Row 2), we tune the primary hyperparameter C with this set of values c: 0.1, 0.5, 1, 10, 50, 100, 1000. Figure 2.4 shows the performance of the logistic model with different C=1/c:

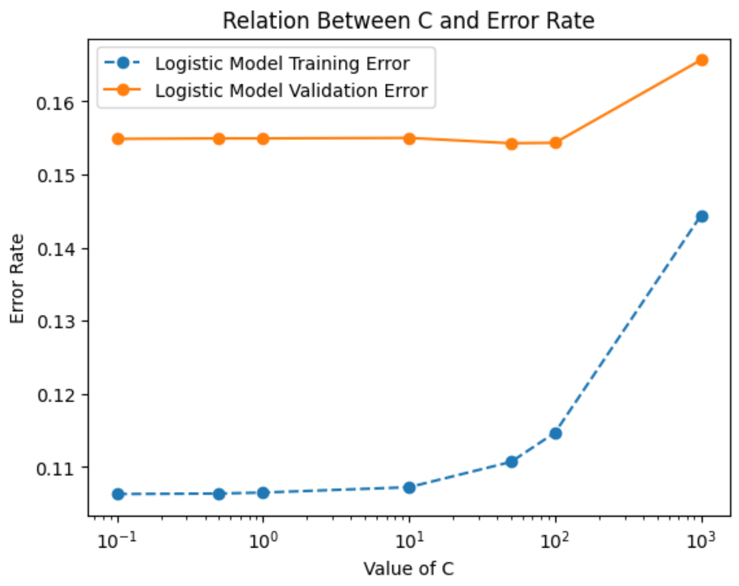


Figure 2.4. Error rate of logistic classifier with different values of C

Based on Figure 2.4, both validation and training error rate increase after the threshold c=50, indicating the model is moving toward underfitting. Hence, C=1/50 will be selected for the classifier since both training and validation errors are low.

<u>Feedforward Neural Network (FNN):</u>

A feedforward neural network is a non-linear mapping from a feature vector to an output (label). For FNN classifiers, one primary hyperparameter is learning_rate_init. This is the initial learning rate used in training and controls the step-sizes when updating the weights. Before tuning learning_rate_init for the FNN classifier, we performed grid search with a selected range of hyperparameters (Fig. 2.5 - Row 1) to obtain the preliminary setup.

hidden_layer_sizes	activation	solver
(50,), (50, 50)	relu, tanh	sgd, adam
(50,)	relu	sgd

Figure 2.5. The selected range of values for each hyperparameter of the FNN classifier

After getting the preliminary setup for other hyperparameters (Fig. 2.5 - Row 2), we tune the primary hyperparameter learning_rate_init with this set of values: 0.0001, 0.001, 0.01, 0.1. Figure 2.6 shows the performance of the FNN model with different learning_rate_init:

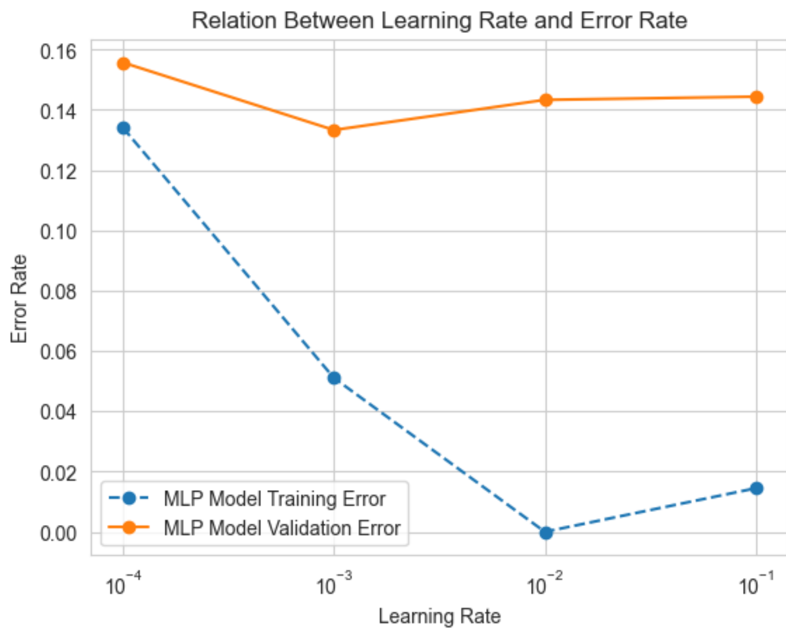


Figure 2.6. Error rate of FNN (MLP) classifier with different values of learning_rate_init

Based on Figure 2.6, the validation error rate has reached a minimum when learning_rate_init = 0.001. Therefore, learning_rate_init=0.001 will be selected for the classifier since the validation error is at the lowest level.

<u>Decision Tree</u>:

A decision tree is a set of propositions on the features connected by paths. It is equivalent to a Disjunctive Normal Form (DNF) representation where each path corresponds to AND statements for propositions along the path and the overall tree is OR of the different possible paths in the tree. For the decision tree classifier, one primary hyperparameter is max_depth. This sets the limit on the complexity of the decision tree. Before tuning max_depth for the decision tree classifier, we performed grid search with a selected range of hyperparameters (Fig. 2.7 - Row 1) to obtain the preliminary setup.

min_samples_split	min_samples_leaf	max_features
2, 5, 10	1, 2, 4	sqrt, log2
10	4	sqrt

Figure 2.7. The selected range of values for each hyperparameter of the decision tree classifier

After getting the preliminary setup for other hyperparameters (Fig. 2.7 - Row 2), we tune the primary hyperparameter max_depth with integers from 1 to 20. Figure 2.8 shows the performance of the decision tree model with different max_depth:

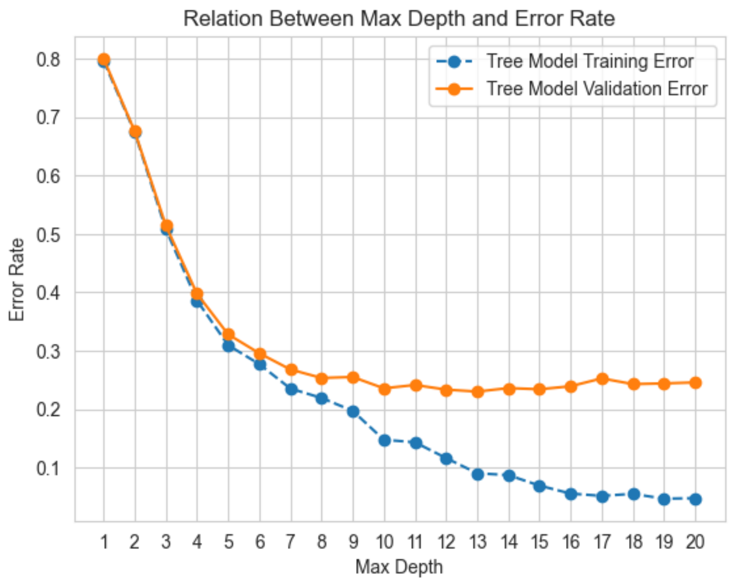


Figure 2.8. Error rate of decision tree classifier with different values of max_depth

Based on Figure 2.8, the classifier has the lowest testing errors when max_depth=10, and starts to gradually increase after this threshold while training errors continue to decrease. This indicates that the model is starting to overfit. Therefore, max_depth=10 will be selected for the classifier.

Experimental Setup:

There are three main steps to set up the experiment. Firstly, data partitioning was performed by dividing the Fashion-MNIST training dataset into two sets: a training set (75%) and a validation set (25%). We also hold aside a testing set (with 10000 samples) provided by Fashion-MNIST to conduct a final evaluation of all classifiers.

Secondly, tuned the preliminary hyperparameters for each classifier by selecting a range of values, and used grid search (GridSearchCV) to perform an exhaustive search to yield the combinations with the highest mean cross-validation scores. After obtaining the preliminary parameters, we further refined each classifier's primary hyperparameter (Refer to 'Classifier' section for details).

Lastly, after getting all the best possible hyperparameters for all classifiers, we evaluated their final performances on the test dataset by looking at different metrics: classification accuracy, precision, recall, and f1-score, which will be discussed in detail in the next section.

Experimental Results:

Following the accuracy evaluation on the testing dataset using the fine-tuned hyperparameters for each model, we found that the FNN classifier obtained the highest accuracy score of 0.8758, while the Decision Tree classifier has the lowest accuracy score of 0.7759 (Fig. 3.1). The results suggest that the FNN classifier has the ability to form complex and non-linear decision boundaries to classify each fashion category. On the other hand, the decision tree can only form axis-parallel decision boundaries, leading to more simplistic and segmented decision boundaries, and therefore limits its ability to capture the complex relationships in the data.

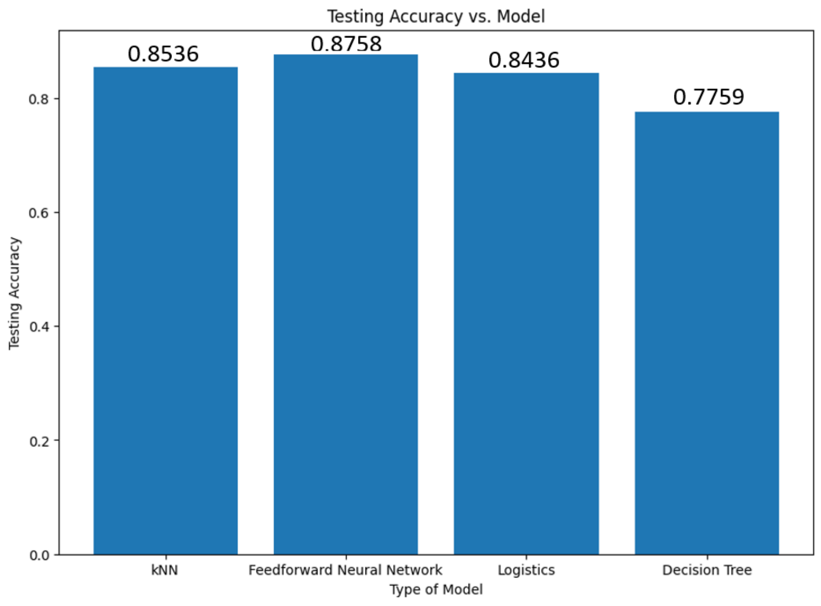
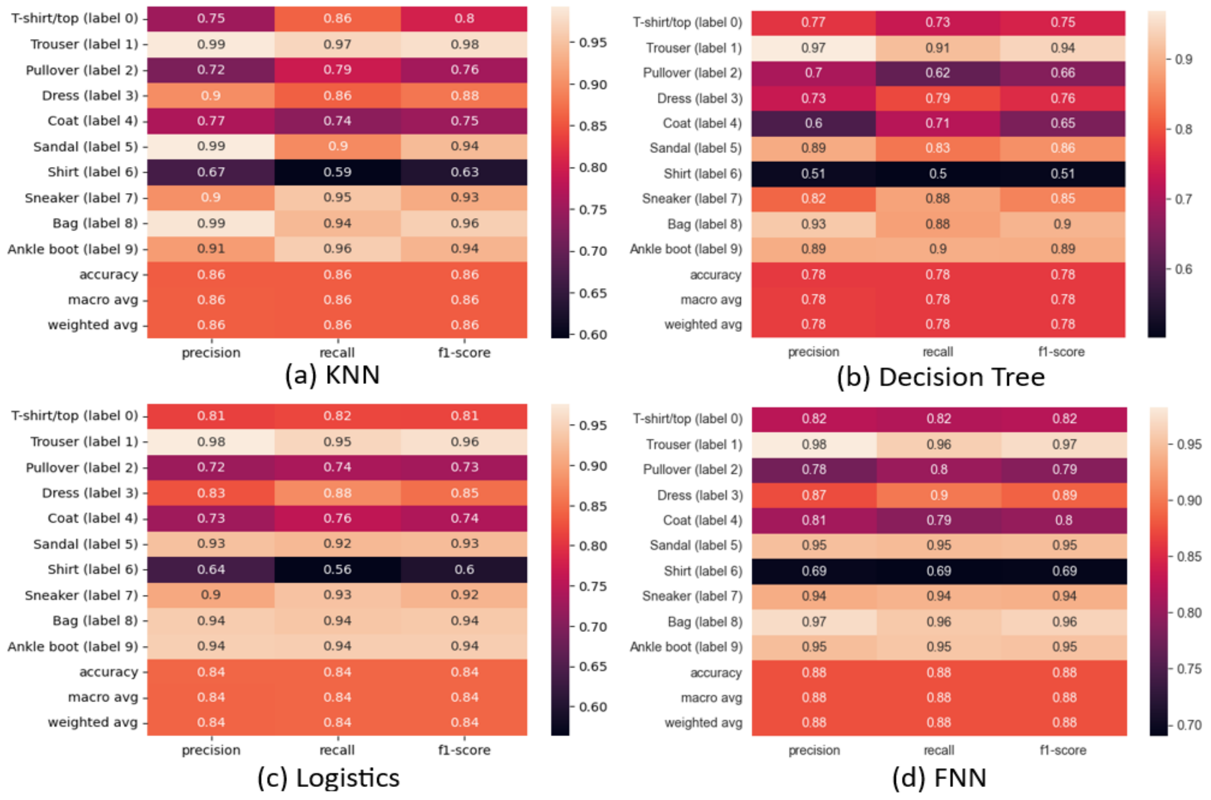


Figure 3.1. Bar chart of accuracy on the testing dataset for each model

In addition to the accuracy score of the model, we also analyzed the precision, recall, and F1-scores for each model. These metrics provide a more comprehensive evaluation of the models' performance for each fashion category.

Precision measures the likelihood of correctly classifying the positive prediction, while recall measures the proportion of true positives that are correctly classified. Referring to Figure 3.2, Shirt (label 6) has the lowest precision and recall values. This indicates that for all our classifiers that predict the new data points as Shirt, the best classifier FNN is correct only 69% of the time. Additionally, among all the data points that are actually Shirts, the best classifier can only identify 69% of them as Shirt. Ultimately, looking at the F1-score, which balances both precision and recall values, illustrates that all classifiers have low accuracies in predicting Shirts, followed by Coat (label 4) and Pullover (label 2).

Overall, the FNN classifier has the best performance among all four classifiers we have experimented with the highest accuracy score of 0.8758 and macro average F1-score of 0.88. These metrics highlight FNN's proficient handling of complex, non-linear relationships between features.



(a) KNN

	precision	recall	f1-score
T-shirt/top (label 0)	0.75	0.86	0.8
Trouser (label 1)	0.99	0.97	0.98
Pullover (label 2)	0.72	0.79	0.76
Dress (label 3)	0.9	0.86	0.88
Coat (label 4)	0.77	0.74	0.75
Sandal (label 5)	0.99	0.9	0.94
Shirt (label 6)	0.67	0.59	0.63
Sneaker (label 7)	0.9	0.95	0.93
Bag (label 8)	0.99	0.94	0.96
Ankle boot (label 9)	0.91	0.96	0.94
accuracy	0.86	0.86	0.86
macro avg	0.86	0.86	0.86
weighted avg	0.86	0.86	0.86

(b) Decision Tree

	precision	recall	f1-score
T-shirt/top (label 0)	0.77	0.73	0.75
Trouser (label 1)	0.97	0.91	0.94
Pullover (label 2)	0.7	0.62	0.66
Dress (label 3)	0.73	0.79	0.76
Coat (label 4)	0.6	0.71	0.65
Sandal (label 5)	0.89	0.83	0.86
Shirt (label 6)	0.51	0.5	0.51
Sneaker (label 7)	0.82	0.88	0.85
Bag (label 8)	0.93	0.88	0.9
Ankle boot (label 9)	0.89	0.9	0.89
accuracy	0.78	0.78	0.78
macro avg	0.78	0.78	0.78
weighted avg	0.78	0.78	0.78

(c) Logistics

	precision	recall	f1-score
T-shirt/top (label 0)	0.81	0.82	0.81
Trouser (label 1)	0.98	0.95	0.96
Pullover (label 2)	0.72	0.74	0.73
Dress (label 3)	0.83	0.88	0.85
Coat (label 4)	0.73	0.76	0.74
Sandal (label 5)	0.93	0.92	0.93
Shirt (label 6)	0.64	0.56	0.6
Sneaker (label 7)	0.9	0.93	0.92
Bag (label 8)	0.94	0.94	0.94
Ankle boot (label 9)	0.94	0.94	0.94
accuracy	0.84	0.84	0.84
macro avg	0.84	0.84	0.84
weighted avg	0.84	0.84	0.84

(d) FNN

	precision	recall	f1-score
T-shirt/top (label 0)	0.82	0.82	0.82
Trouser (label 1)	0.98	0.96	0.97
Pullover (label 2)	0.78	0.8	0.79
Dress (label 3)	0.87	0.9	0.89
Coat (label 4)	0.81	0.79	0.8
Sandal (label 5)	0.95	0.95	0.95
Shirt (label 6)	0.69	0.69	0.69
Sneaker (label 7)	0.94	0.94	0.94
Bag (label 8)	0.97	0.96	0.96
Ankle boot (label 9)	0.95	0.95	0.95
accuracy	0.88	0.88	0.88
macro avg	0.88	0.88	0.88
weighted avg	0.88	0.88	0.88

Figure 3.2. Heatmap of precision/recall/f1-scores for each fashion category

Insight:

One primary takeaway from this project was understanding the challenges of classifying visually similar categories. We found that all classifiers struggled to accurately predict "Shirts". One hypothesis on why all classifiers have low accuracy in predicting shirts is that the features, which are pixel values of images, for Shirts are similar to certain other fashion categories in the dataset. For instance, Shirts and T-shirts/tops can exhibit the same visual patterns, which makes classifiers often confuse Shirts with T-shirts. Similarly, other categories such as Dress (label 3), Coat (label 4), and Pullover (label 2) also exhibit lower accuracy (<0.9 F1-score) as they also have similar visual patterns.

Contributions:

Chi Zhang: visualization of the dataset; provided example codes for members to work on other classifiers; tuned and evaluated KNN classifier; wrote multiple sections of the report (Summary, Data Description, Classifiers - KNN part, Experimental Setup, Experimental Results, Insight, and Appendix); final evaluation of the report and correct mistakes.

Shengyuan Lu: I tuned the hyperparameters, evaluated the classifiers, and created learning curves for the decision tree and feedforward neural network classifiers. To ensure accurate comparison, I closely collaborated with Chi and Kaiyuan to ensure that all classifiers had identical sections. Wrote the learning curve analysis on the report and checked the report for grammar errors.

Kaiyuan Hu: I focused on the logistic classifier, actively involved in fine-tuning the hyperparameters, conducting thorough evaluations of the classifiers, and generating informative learning curves. I also wrote the Logistic Classifier part, Feedforward Neural Network part, and Decision Tree part of "Classifiers" in the report.

Appendix

Appendix 1 - Dataset Visualization

Given the high-dimensionality nature (784 features) of the dataset, direct visualization is impractical. Therefore, we applied Principal Component Analysis (PCA) to visualize how the samples are scattered, which is a technique that can be used to reduce the dimensionality of data to a lower dimensional space by clustering variables that are likely to be correlated (Shlens 1). The scatter plot provides an overview of the relationship between different data points and their corresponding fashion categories (Fig. 4.1).

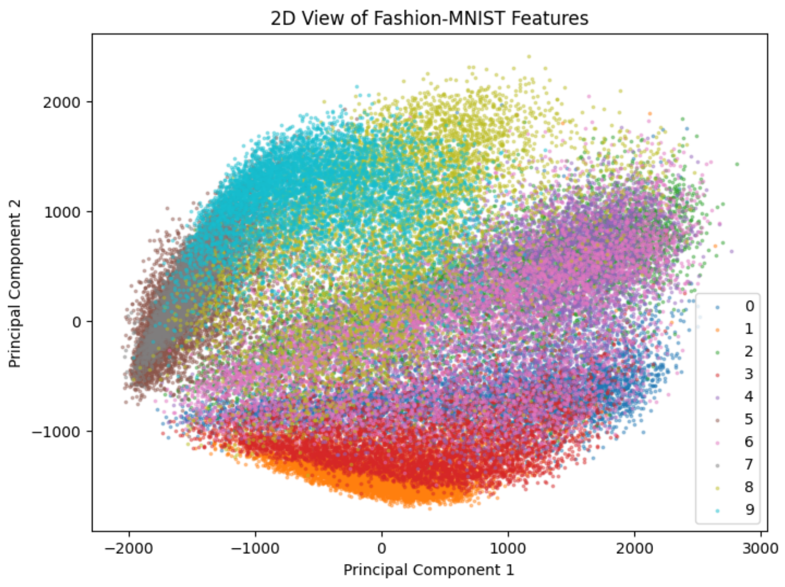


Figure 4.1. Scatter plot of training dataset in two dimensions through PCA

Appendix 2 - Learning Curves Analysis

Based on the learning curves in Figure 4.2, we can observe that the FNN classifier has high variation because it still has a high decrease rate in testing error after training on 20,000 data, followed by the KNN classifier. On the other hand, the testing error of the logistic classifier begins to flatten out, which suggests it has a high bias as it can only learn (piecewise-)linear decision boundaries.

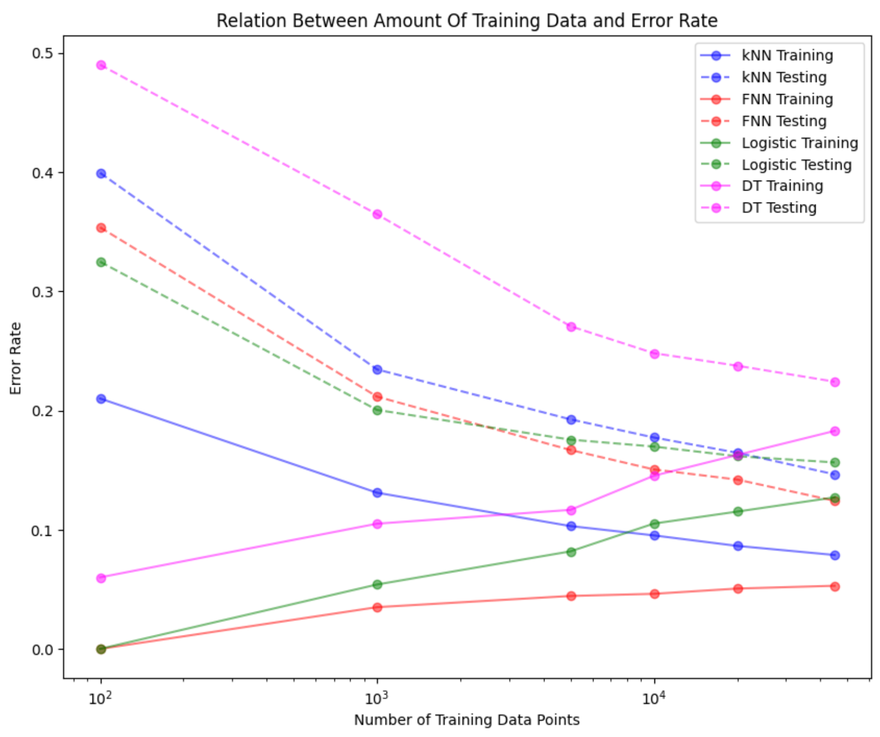


Figure 4.2. Learning Curves of All Classifiers

Appendix 3 - Project GitHub Repository

Link: https://github.com/shengyuan-lu/CS-178-Project

- Include one jupyter notebook for each classifier
- Include Fashion-MINIST visualizations

Works Cited

Shlens, Jonathon. "A Tutorial on Principal Component Analysis." *ArXiv*, 2014, /abs/1404.1100. Accessed 4 June. 2023.