Machine Learning Methods for Modeling and Classification of fashion MNIST

A Preprint

David Blumenstiel
Department of Data Science and Information Systems
CUNY School of Professional Studies
NY, NY 1001
blumenstieldavid@gmail.com

Bonnie Cooper
Department of Biological Sciences
SUNY College of Optometry
NY, NY 10036
bcooper@sunyopt.edu

Robert Welk
Department of Data Science and Information Systems
CUNY School of Professional Studies
NY, NY 1001
robert.j.welk@gmail.com

Leo Yi
Department of Data Science and Information Systems
CUNY School of Professional Studies
NY, NY 1001
leo.yi35@spsmail.cuny.edu

December 9, 2021

Abstract

Fashion MNIST is a clothing classification dataset that is popular for deep learning and computer vision applications. In this report, we use Fashion MNIST to apply multiple machine learning methods reviewed in Data622: Machine Learning and Big Data, an elective course for the Masters of Science in Data Science at CUNY School of Professional Studies. We explore approaches to reduce the dimensionality of the data by engineering new descriptive features and performing Principal Components Analysis. We follow this up with machine learning methods for classification such as k-Nearest Neighbor, Support Vector Machine, Convolutional Neural Network and more. We find that dimensionality reduction can be a powerful tool for speeding up the training of a model while still delivering comparable accuracy on classification tasks.

***K*eywords** fashion MNIST · machine learning · classification

1 Introduction

Fashion MNIST is a clothing classification dataset that builds in complexity in comparison to the classic MNIST dataset. MNIST is a dataset of handwritten digits that has been a go-to dataset for benchmarking various image processing and machine learning algorithms (LeCun et al. 1998). Classification algorithms applied to MNIST revolutionized the field of image processing in the 90s (Krizhevsky, Hinton, and others 2009). However, contemporary machine learning methods can achieve 97% accuracy. Convolutional neural networks score as high as 99.7% accuracy. As a result, MNIST is now considered too easy and, with ~48000 MNIST related publications (Noever and Noever 2021), MNIST has also been used by the machine learning community exhaustively. Fashion MNIST was developed as an alternative.

Figure 1: Exemplar Fashion-MNIST images. All image files are grayscale 28x28 images

Fashion MNIST can serve as a direct drop-in replacement for MNIST. Fashion MNIST and MNIST are both labeled data that share the same dataset size: 60,000 training images and 10,000 test images. Additionally, Fashion MNIST and MNIST images share the same dimensions and structure: 10 distinct categories of grayscale images with 28x28 pixel size. Figure 1 depicts a sampling of several dozen example Fashion-MNIST images. Due to the complexity and variety of the images, Fashion MNIST is a more challenging dataset for machine learning algorithms (Xiao, Rasul, and Vollgraf 2017).

In this report we utilize Fashion MNIST to implement a variety of machine learning approaches covered over the course of our studies in Data622: Machine Learning and Big Data as part of the Masters of Data Science program at CUNY School of Professional Studies. Our goal is to use dimensionality reduction to optimize the classification of Fashion MNIST images using a variety of machine learning algorithms. Here, we describe our approaches and compare classification model performance.

With an image size of 28 x 28, Fashion MNIST images, in their raw form, are high dimensional data with a total of 784 features. Additionally, there is a relatively high degree of correlation between pixel values of a given image. That is to say that, unless an pixel is at an object border, a light pixel in a region of the image is very likely to be situated next to light neighboring pixels, likewise for dark pixels. We can assess this by observing the mean and standard deviation of the pixel values across the dataset.

Figure 2 visualizes the mean (left) and standard deviations (right) of the pixel values for the `train` dataset. For both plots, higher intensity values are rendered as dark whereas low values are light. For many of the pixels in the image, the mean values are intermediate (gray) whereas the corresponding standard deviations are relatively high (dark). These pixels make up most of the variance in the dataset. However, we can see that there are two regions of pixels towards the image center where the mean pixel value is high (dark) while the pixel standard deviation is low (light). For these regions, the pixels have consistently high pixel values. Towards the periphery of the image there are pixels with both low mean values (light) and low standard deviation (light). These pixels have consistently low values. The pixels with either consistently low or high values are of low information content and would not contribute much to models for image classification if used as features. Therefore, we describe two methods for reducing the dimensionality of the Fashion-MNIST dataset. First, we describe a Feature Engineering approach where we build summary features that quantify distinctions between categorical images. Second, we describe a dimensionality reduction approach that

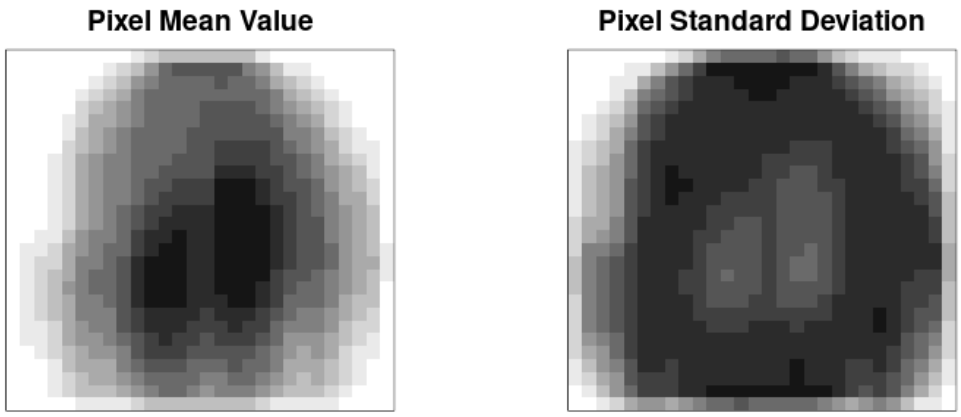

Figure 2: Pixel values across dataset. Left panel: mean pixel values. Right panel: pixel value standard deviation

uses Principal Component Analysis. Furthermore, this report goes on to describe multiple different models developed from the resulting reduced sets and evaluate the performances.

2 Modeling Fashion-MNIST with Engineered Features

2.1 Building Feature Representations based on Categorical Differences

One of the methods that can be used for dimensionality reduction is feature engineering. Instead of processing 28^2 or 784 individual variables, we will use feature engineering to process the data into fewer variables that attempt to summarize information within the original dataset. If resources were unlimited, we could use both the original pixel variables along with the features below.

Figure 3 is similar to Figure 2, however Figure 3 shows the mean pixels intensity for each Fashion-MNIST categorical label. Here we are attempting to generalize the shapes and pixels used by each of the different fashion items. Looking at the shoes, we can see that there are full horizontal rows that are completely white, which is also common to bags. As we look at the differences and similarities of each of the classes, the shape of different sections allows us to classify the labels in our minds, which we'll try to replicate with feature engineering that follows. If we can create new variables to try and capture the differences in shape, we can see if those variables will be useful in contributing to the accuracy of the models we train.

Below is a description of the engineered feature set:

- % of pixels that are different shades of grey – 0, 50, 87, 167, 209, 255
  - white: 0
  - grey1: 1-46
  - grey2: 47-93
  - grey3: 94-168
  - grey4: 169-205
  - black: 206-255
- % of non-white pixels in each of the 28 rows
- % of non-white pixels in each of the 28 columns
- % of rows that are completely white

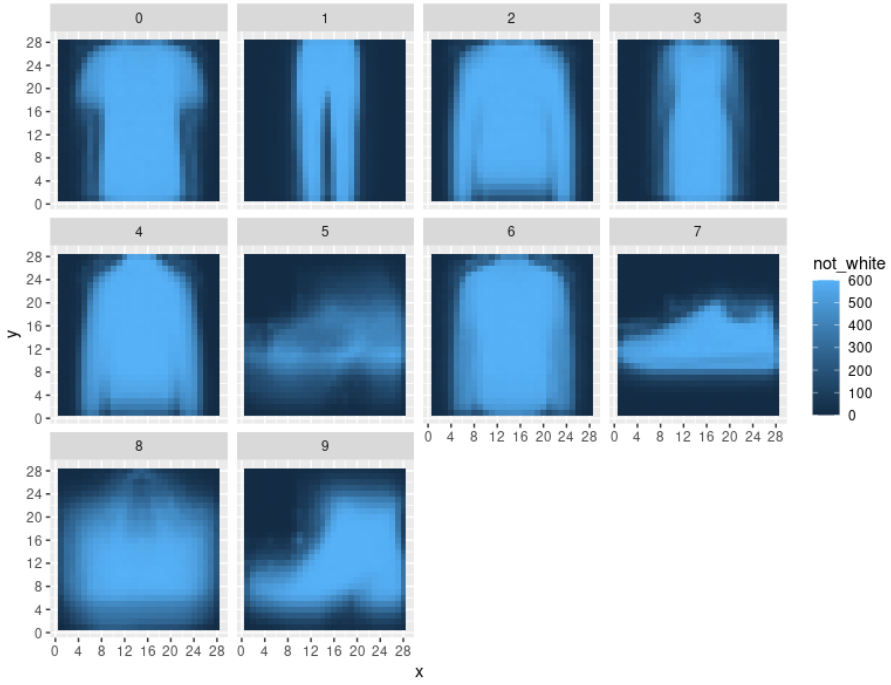

Figure 3: Categorical Heat Map: Each panel shows mean pixel intensity for each Fashion-MNIST category

- % of columns that are completely white
- 10 custom blocks that are subsections of the grids, showing percent non-white as well as percent light grey (1 - 122).

The features listed were determined by visually inspecting Figure 3. Moving forward in this report, this feature set will be referred to as the Feature Engineered dataset. For detailed code that describes the feature generation, please refer to the report code documentation.

2.2 Modeling Fashion-MNIST with the Featured Engineered Dataset

Once the Feature Engineered dataset was generated, we performed 10 fold cross validation to train the following models:

- Random forest
- Support vector machine using a radial kernel
- k-Nearest Neightbor
- Multinomial logistic regression
- Naive Bayes

These models were trained exclusively on our Feature Engineered dataset. Additionally, in order to deal with the hardware limitations of personal computers, the training data will only use a fraction of the original 60,000 training records. The training set used was a stratified sample which accounts for 16.7% or 10,010 observations of the original training set.

Table 1 gives a summary of a few select metrics that describe the training and evaluating of the different model approaches. In terms of training duration, the fastest model was Naive-Bayes at 0.2 minutes. However, the radial Support Vector Machine (SVM) model trained for the longest at 19 minutes. Model duration is important to consider, especially since all modeling procedures using the Engineered Feature dataset we used a subset of ~10,000 training instances. Despite the long training, the radial SVM model performed the best by scoring the highest accuracy on the test data. However, the Random Forest model achieved a comparable

Table 1: Model Training Summary

Model Type	Training Duration (min)	Training Accuracy	Test Accuracy
SVM (Radial)	19.0	0.854	0.856
Multinomial Log Reg	9.3	0.861	0.824
Random Forest	5.5	0.799	0.855
k-NN	2.6	0.8292	0.795
Naive Bayes	0.2	0.715	0.713

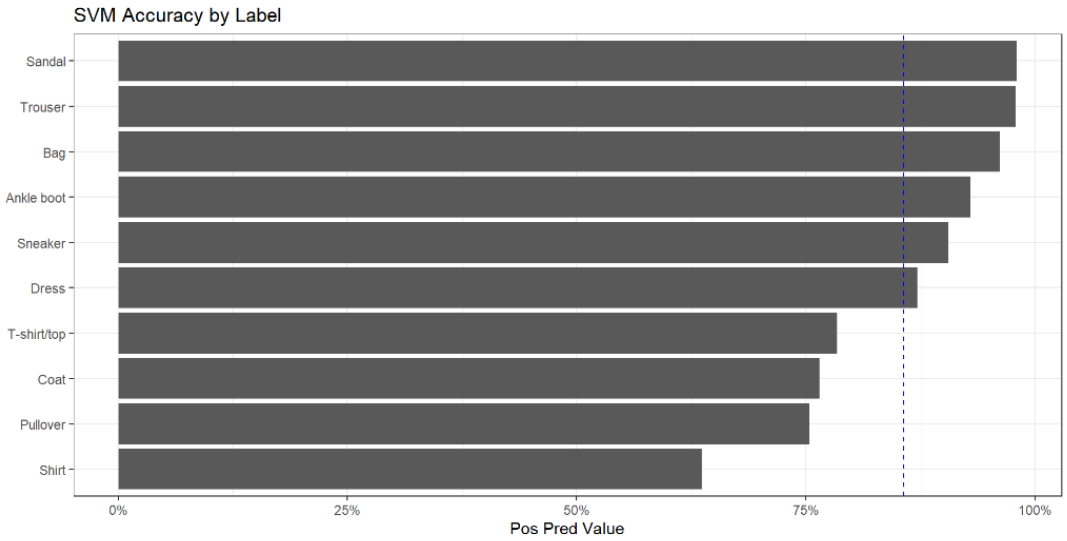

Figure 4: SVM Accuracy by Label. the accuracy for each Fashion-MNIST label category is given as a row in this vertical bar chart. The blue line shows the overall accuracy for classification of the test data.

test accuracy. Therefore, if long training durations are of concern, the Ransom Forest model is a reasonable alternative.

The performance accuracy given in Table 1 is an important measure of a model's performance. However, it is not a very thorough measure of a model's ability to predict the proper labels. Therefore, we also looked at accuracy for the SVM model broken down by category. Figure 4 shows the accuracy for each Fashion-MNIST categorical label. As expected, the errors for the SVM model trained using only variables summarizing subsections or shapes of the original training data led to specific items of clothing to be confused for others. For instance, similarly shaped items such as 'Shirt,' 'Pullover,' and 'Coat' show the lowest accuracy for classifications. The similar shapes of the items suggest that the derived variables that were used did not adequately pick up the variance among the items. Furthermore, the confusion matrix for the SVM model (see report code documentation) show common misclassifications of pullovers as coats and shirts as t-shirts/tops or pullovers. Additionally, for each of the three types of footwear, the errors predicted by the model are most likely one of other two footwear types.

3 Modeling Fashion-MNIST with Principal Component Analysis

3.1 Principal Component Analysis of Fashion-MNIST

We have shown above that there is redundancy in the fashion MNIST dataset. Here we will use PCA to reduce the number of features while retaining as much of the variance possible. PCA does this by finding a new set of axes that fit to the variance of the data. At heart, PCA is simply an eigendecomposition of the data which returns a set of eigenvectors and eigenvalues. Eigenvectors and eigenvalues describe the transformations necessary to go from the original axes to a new feature space.

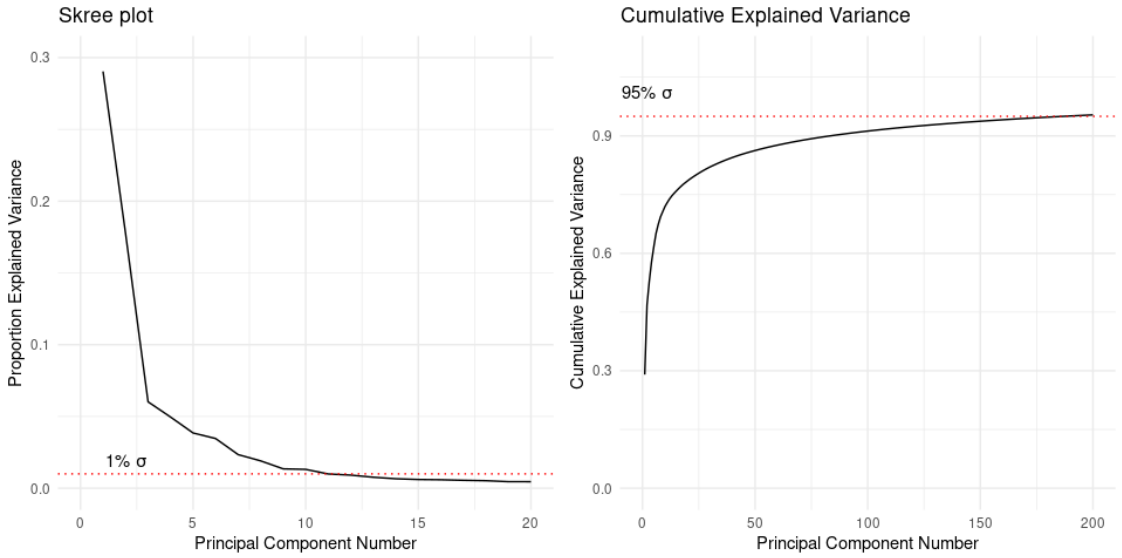

Figure 5: PCA Results. The Skree plot (Left) shows the proportion on explained variance for each the first 20 principal components. The Cumulative Explained Variance plot (Right) shows the total variance explained by successively addding up to 200 principal components

We can use the results of PCA to perform a type of information compression on the original data by subsetting the amount of PCA components we use to describe the original data. For this analysis, we will use a criterion of 95% variance explained. From the 784 components that PCA yields, we will subset the minimum components needed such that the sum of the proportions of explained variance is greater than or equal to 95%. Such a manipulation is favorable because it will reduce redundancy in the data, the chances of overfitting, and the time necessary to train models.

From the skree plot in Figure 5 (left panel), we can see a very sharp drop off in the proportion of explained variance. Principal Components greater than 12 account for less than 1% of the dataset's variance. The first 12 components only account for a cumulative variance of 0.74, therefore, as the right panel in Figure 5 shows, it takes the combined contribution of many more components (187 components) to explain 95% of the variance of each pixel for the original images.

PCA returns a component for every dimension of the data in descending order of the amount of variance accounted for. Figure 6 shows the the first 4 components, component PC392 (middle), and the last component PC784. As already depicted in the Skree and Cumulative Explained Variance plots of Figure 5, the first four components explain 0.58% variance from PC1: 29% $\rightarrow$ PC4: ~5% variance. We can see clearly from the visualization of the components that the first several PCs are clearly discriminating between clothing classifications. For instance, PC1 distinguishes between T-shirt/top & Pullover (dark/high values) and shoe categories (light/low values). On the other hand, PC2 appears to distinguish Trousers from shoe categories. The representation becomes less clear as the explained variance decreases. For instance, PC392 and PC784 only explain 0.01% and 0.001% variance respectively and it is not clear from the visualization just what information these components represent.

PC1 and PC2 account for roughly half (~47%) of the train dataset's variance. Figure 6 visualizes the projections of the train data onto the features space of these two components. Figure 6 shows representations of the `train` images in the feature space of the first 2 dimensions. Each clothing item category is represented by a different color. For clarity, a text label (in black) which corresponds to the categorical mean values of PC1 & PC2 has been added. We can see that there is noticeable separation across the categories. Additionally, we see some clustering of category means that meets expectations. For example, the shoe categories (Sandal, Sneaker and Ankle Boot) group together towards the lower left hand corner of the figure. Clothing items that could all be described as tops with sleeves (Pullover, Coat, Shirt & T-shirt/top) group together in the middle of the distribution. Trousers, on the other hand, have a noticeable distance from tops with sleeves but are contiguous with the dress category which shares a roughly vertical rectangular profile.

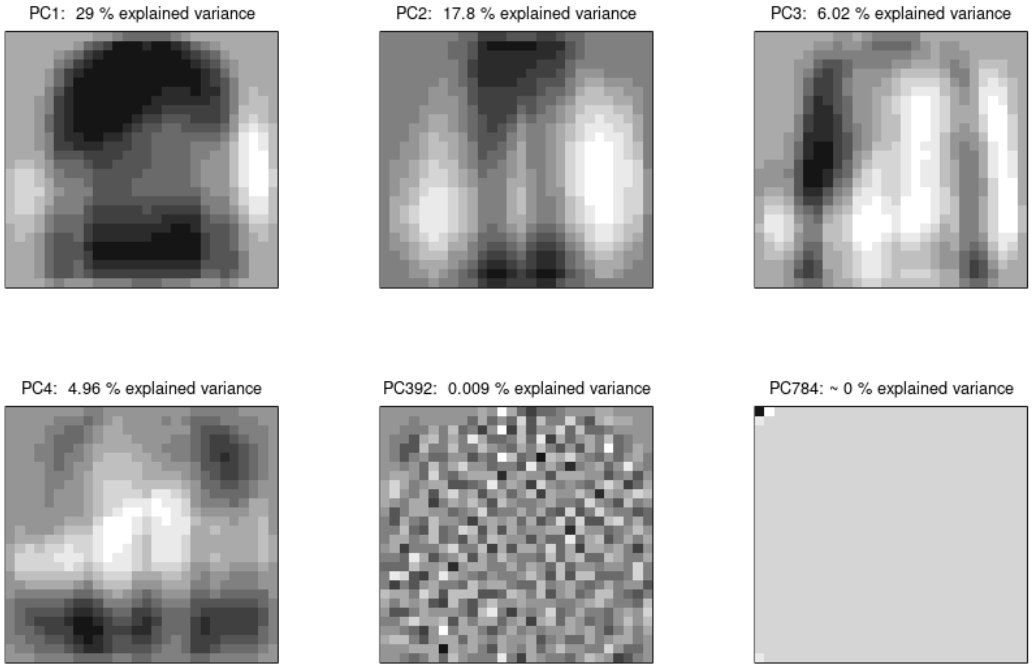

Figure 6: Visualizing PCA Components. The top row (from left to right) and bottom left panel visualize principal components 1 through 4. The middle panel of the bottom row shows the 392th component. The bottom right panels shows principal component 784

Figure 8 shows the representation of the data onto the first 187 components which were shown earlier to account for 95% of the variance in the image data. The result are information compressed versions of the original images. Truncating the data to 187 components does result in information loss, however, as we can see from the visualizations, the images retain much of the detail from the original images while using a feature space 23.85% the size of the original.

PCA is a dimensionality reduction method that can be applied to high dimensional datasets. PCA reduces the number of features while preserving as much variance from the data as possible. Here we used PCA to reduce the number of features of the Fashion MNIST dataset from 784 to 187. We showed through a series of visualizations that transforming the images to the reduced feature space does so with a noticeable loss to image quality, however the gist of the images are still present.

3.2 Modeling Fashion-MNIST with and without the use of PCA Compression: k-Nearest Neighbors

In the previous section, we showed that PCA can be used on Fashion-MNIST to reduce the dimensionality of the data to less than $\frac{1}{4}th$ the original size. Here we implement k-Nearest Neighbor, a non-parametric method, to classify Fashion-MNIST images using both the original data which uses all 784 columns of pixel values as features and the PCA compressed data which reduces the feature space the 187 columns. As in other analyses in the report, these models will also be 'trained' on a subset of 10000 samples, to reduce training times.

K-Nearest Neighbors (kNN) is a non-parametric technique that can be used for image classification. It's a very simple technique, wherin new samples are classified as the most typical class among the 'k' nearest labeled examples it was 'trained' on where 'k' is an integer value. That being said, kNN doesn't actually train, rather, it stores the data and refers back to it when making classifications. For our analysis, the `caret` package was used to create this model, and although it may appear like it's fitting a model, what it's actually doing here is storing all the training data, then trying out different 'k' values, one of which it will choose to use when 'predict' is called. Even if it's not semantically well implemented, it's still very useful for finding a good 'k' value.

Figure 7: Magnitudes of PCA 1 and 2 for all Fashion-MNIST images from the 'train' dataset. Datapoints are shaded according the clothing category. A black text label is rendered at the mean PC1 and PC2 values for each category.

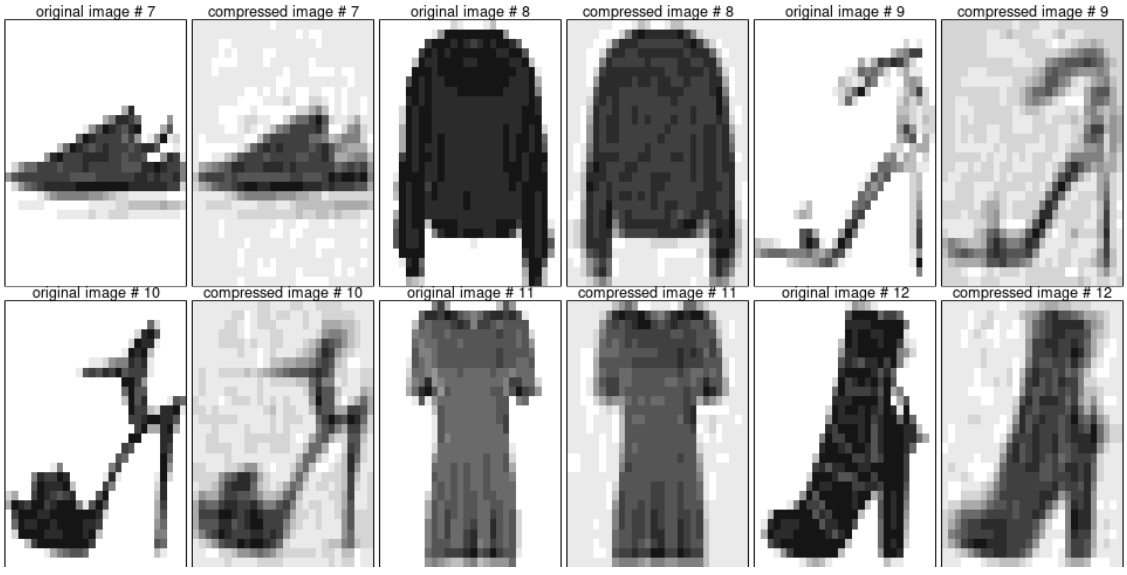

Figure 8: Reconstructed Images using 187 Principal Components. Original Fashion-MNIST images are rendered next to the corresponding compressed images which are reconstructed from the first 187 components

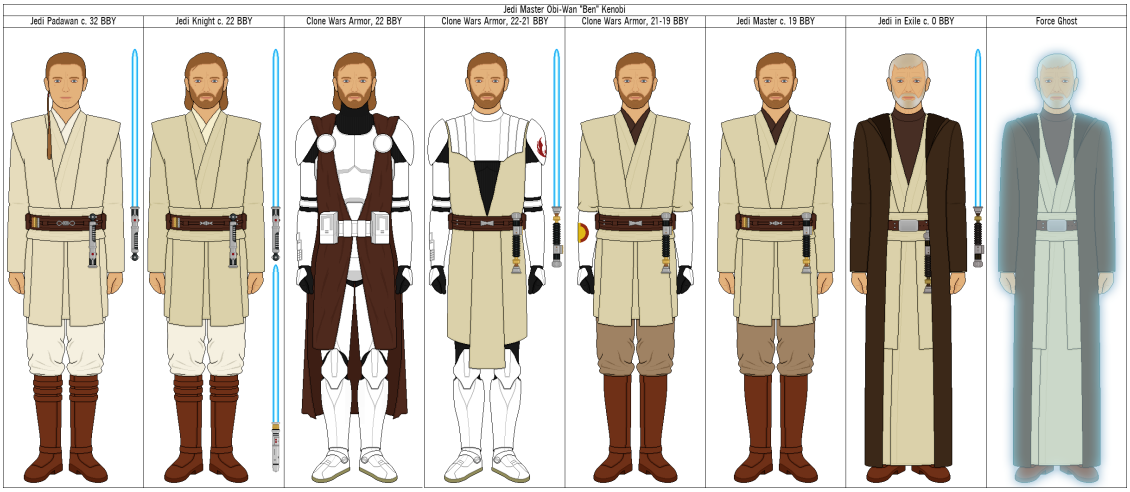

Figure 9: kNN: hyperparameter tuning curve to find the best k value for the original Fashion-MNIST data

We begin by fitting a kNN model to the original Fashion-MNIST dataset. Although this analysis starts with all 784 features, we whittle down some of the data by choosing only variables (pixels) which vary significantly between images. For example, if a pixel in the corner of the images has the same value throughout most images, this pixel will dropped from the data and not be used in fitting model. For full details and code, please refer to the report code documentation. Figure 9 shows the cross-validation accuracy across several values of 'k' while training the model with the `train` subset of the data. This plot gives the hyperparameter tuning curve for the model. We see that k=5 performed as the best value with the training data.

With k=5, we evaluated the model performance for classifying the test data. The overall accuracy for this model was 81.33% (95% CI: 80.57 - 82.09%). The confusion matrix (see report code documentation) informed us that the model has an easier time detecting some labels than others. For example, it correctly classifies most of the sneakers (label 7), bags (8), ankle-boots (9), and trousers (1). However, it has a much harder time classifying pullovers (2), coats (4), sandals (5), and shirts (6). Shirts in particular fared poorly, with only about half correctly classified; t-shirts/tops (0), pullovers (2), and coats (4) were also often missclassified as shirts, which makes sense (they are very similar).

Next we will implement kNN with the PCA compressed data, and see if it has any effect on performance when compared to the previous model. Figure 10 shows the hyperparameter tuning curve used to find the best value of 'k.' When kNN is fit on a subset of 10,000 samples from the PCA reduced feature space, a k value of 7 appears to perform best during cross-validation. The cross-validation accuracies across k values of 1-11 were similar; see Figure 10.

The changes in accuracy across different values of k (Figure 10) aren't terribly drastic. K values of 5-7 performed best across all tested values when fit with the training subset of data. Next we evaluate performance on the test set. The overall accuracy of this model was 82.36% (95% CI: 81.61 - 83.11%). It performed very similarly to the previous model which used a non-PCA compressed dataset. We can see similar patterns in the confusion matrix (see report code documentation): most of the sneakers (label 7), bags (8), ankle-boots (9), and trousers (1) where correctly classified. Also, the model had a harder time classifying pullovers (2), coats (4), sandals (5), and shirts (6). We also see similar missclasifications to the results from the original feature space, where shirt classification is poo, and t-shirts/tops (0), pullovers (2), and coats (4) were often missclassified as shirts.

The PCA compressed kNN model performs slightly better than the kNN model trained with the original feature space (82.35% vs 81.23% overall accuracy; a significant difference at $\alpha = 0.05$). It's unclear why this happens. One could theorize that perhaps the removing some of the precision in the data may make for more robust models that resist over-fitting. It could also be that the data is more separable in a smaller feature space, which kNN would benefit from significantly.

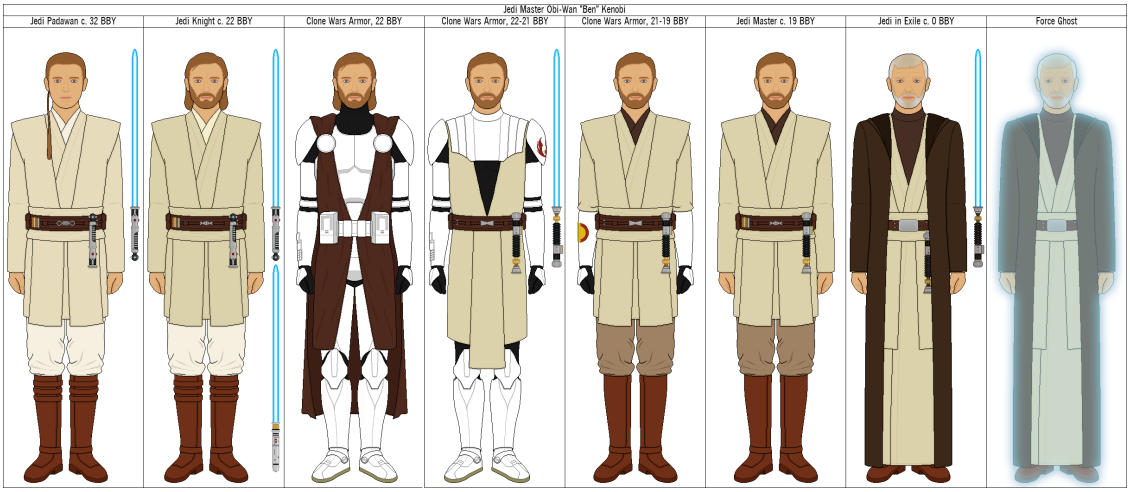

Figure 10: kNN: hyperparameter tuning curve to find the best k value for the PCA compressed Fashion-MNIST feature space

3.3 Modeling Fashion-MNIST using PCA Compression: Additional Machine Learning Models

Using the PCA compressed feature space, three additional model approaches were trained, and quality of predictions was evaluated. For each model, hyperparamters were tuned using 10-fold-cross-validation on the training set to find the combination that maximized Accuracy. Due to the large number of observations in the Fashion MNIST training set, only the first 10,000 observation were used due to insufficient computing power. Here we provide a summary of the implementation for Stochastic Gradient Boosting, Random Forest and a Neural Network model and evaluate each models performance towards classification of PCA compressed Fashion-MNIST. For details and code, please refer to the report code documentation.

1. Stochastic Gradient Boosting Gradient boosting is an ensemble machine learning technique that has application in both regression and classification, and is typically used in decision trees. It is premised on building a sequence of weak models where at each iteration a new tree fits the errors from the previous. For this application, a stochastic gradient boosting (SGB) model will be used. In SGB, a random subsample (without replacement) of variables is used at each iteration so that correlation between successive trees is reduced. Successive trees leverage this randomness to update errors of the base.

There are four hyperparameters that are to be tuned for this model, and a brief description of each is provided as well as the values that were supplied to the grid for cross validation purposes of the model tuning effort:

1. *Number of trees (n.trees):* This refers to the total number of trees to be used in the ensemble. Too many trees can result in overfitting while too few trees can result in underfitting due to the corrective, iterative nature of the algorithm where successive tree's correct the errors at the previous step. Values from 500 to 2500 by 500 were tested during cross validation.
2. *Interaction depth (interaction.depth)*: Controls the depth of the individual trees and subset of random variables at each iteration step. Higher depth trees allows capturing of predictor interactions but could also result in over-fitting. Values from 1 to 7 by 2 were tested.
3. *Learning rate (shrinkage)*: This parameter sets the rate at which a tree learns from the base learner. Smaller rates learn less quickly thus reducing the chances of overfitting but will require a larger number of trees and computation power. Values 0.01, and 0.1 were tested.
4. *Minimum number of observations in terminal nodes (n.minobsinnode)*: Like interaction depth, this also controls the complexity of each tree. Values of 1, 2, and 5 were tested.

- The final values used for the model were n.trees = 2000, interaction.depth = 5, shrinkage = 0.01 and n.minobsinnode = 1

2. Random Forest Random forest is an ensemble decision tree method where a series of decision trees independent from each other are provided a random subset of the predictor space. The results of each

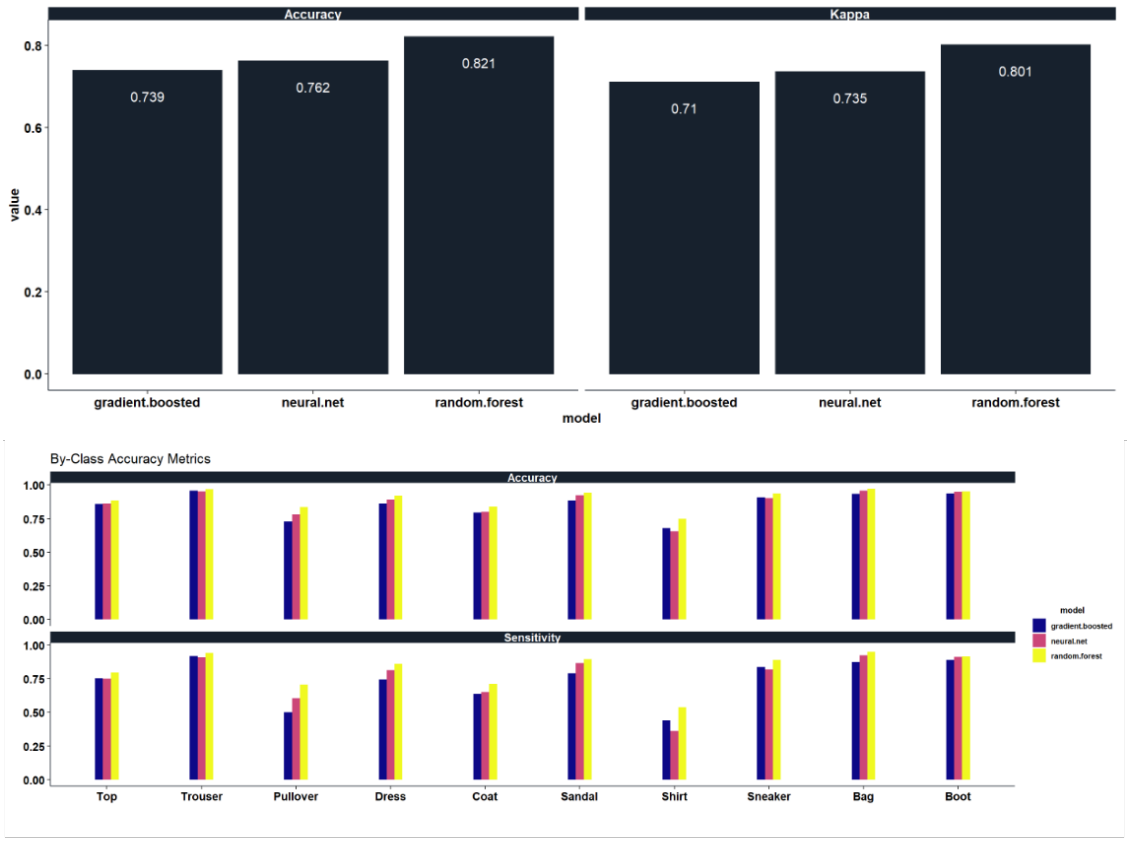

Figure 11: Model Performance Metrics. Top Panels: overall model Accuracy (left) and Cohen's Kappa Statistic (right). Lower Panels: Model Accuracy and Sensitivity for each Fashion-MNIST categorical label

bootstrapped iteration are then given an aggregated response. There is only one hyperparameter for the `rf` implementation of the random forest algorithm that was used.

1. *Number of variables (mtry):* This defines the manner in which variable bootstrapping is conducted by controlling the maximum number of variables that are subset at each interation of the ensemble. Integer values from 1 to 6 inclusive were tested during cross validation.

The final value used for the model was mtry = 3.

3. Neural Network A neural network based on averaging of random seeds, `avNNet` was trained. The following hyperparamters were tuned using cross-validation.

1. *Number of hidden layers (size):* This defines the complexity of the network
2. *Decay:* penalty term for large coefficients; used to regularaize model

The final values used for the model were size = 12, decay = 0.01

We can gain a high-level overview of the model based on the metrics Accuracy and Cohen's Kappa. Accuracy describes how often the classifier was correct, and is defined as the sum of true positive and true negatives divided by the total number of observations. This was determined to be an appropriate metric to evaluate overall model performance since the test set classes are balanced. The Cohen's Kappa statistic is a measure of how well the model prediction matched the labeled test set, while controlling for the accuracy that would have been produced by a random classifier. As a stand alone metric, kappa can be difficult to interpret, but is useful for between model comparisons. The top panel of Figure 11 shows the Accuracy (left) and Kappa (right) performance metrics for the three models. We see that the Random Forest model performed the best.

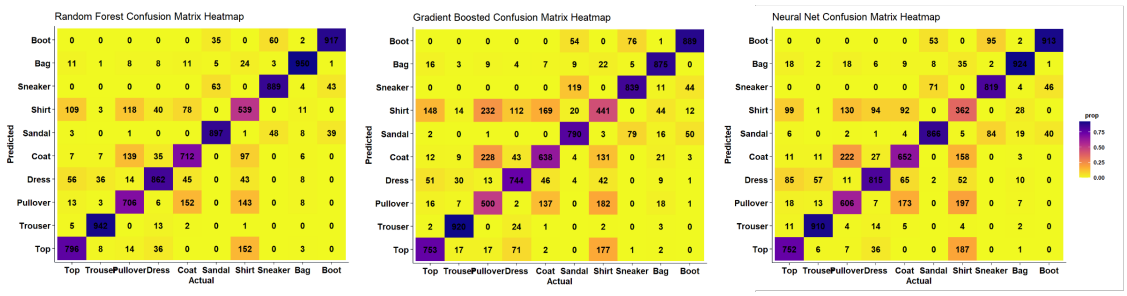

Figure 12: kNN: hyperparameter tuning curve to find the best k value for the PCA compressed Fashion-MNIST feature space

The bottom panel of Figure 11 considers the performance of each of the three models when the Fashion-MNIST classification label is taken into account. We used the classification accuracy and sensitivity (the true positive rate) as metrics to evaluate performance. In Figure 11, we can see that for each category for both Accuracy and Sensitivity, the Random Forest model outperforms both the Stochastic Gradient Descent and the Neural Network models.

Accuracy and Sensitivity are calculated from the results of the confusion matrices that result from evaluating each model's classification results. Figure 12 shows the confusion matrices for the three models; it is very interesting to see that the three models show very similar patterns in classification. For example, all three models struggle to accurately classify the 'Shirt' category. 'Pullovers' and 'Coats' are very commonly misclassified as eachother.

4 Summary & Conclusions

In this report we described two different approaches to implementing dimensionality reduction of the Fashion-MNIST dataset. Our first approach outlined a strategy to engineer features that draw from insights gained from visually inspecting patterns in the data. We referred to this approach as our Feature Engineered Dataset which reduced the dimensionality from 784 features (pixels) to ~80 features. We evaluated the performance of multiple models(Random Forest, SVM, kNN, Multinomial Logisitc Regression and Naive Bayes) and found that a Radial SVM model performed the best. However, Radial SVM had the longest training time (~19minutes) and considering that SVM scales quadratically with the size of a given dataset, this runtime might be prohibitive for some users. Alternatively, the Random Forest achieved comparable accuracy on classifying the test data and trained in a fraction of SVM's training duration, so there is that to be considered.

For our second dimensionality reduction approach, we applied Principal Component Analysis to the Fashion-MNIST dataset. PCA reduces the number of features while preserving as much variance from the data as possible. Here we used PCA to reduce the number of features of the Fashion MNIST dataset from 784 to 187 while still representing 95% of the original data's variance. We showed through a series of visualizations that transforming the images to the reduced feature space does so with an noticeable loss to image quality, however the gist of the images is still present.

We then compared the classification performance of k-Nearest Neighbors models when using the original Fashion-MNIST data and the PCA reduced data. The PCA compressed kNN model performs slightly better than the kNN model trained with the original feature space (82.35% vs 81.23% overall accuracy; a significant difference at $\alpha = 0.05$). We hypothesize that this could possible be because the reduced dimensionality makes the data observations more highly seperable. To pursue this further, we then evaluated the performance of several other machine learning algorithms (Stochastic Gradient Descent, Random Forest and a Neural Network model) for classification of the PCA reduced Fashion-MNIST dataset. Judging by the confusion matrices (Figure 12) we see that these models, although very different algorithms, make similar classification errors across the clothing categories. However, the performance of the Random Forest model stood out as the best classifier.

In conclusion, we find the use of dimensionality reduction would be beneficial in down-stream modeling analysis when working with high-dimensional datasets. As a detraction, additional steps would be necessary to transform the data between full and compressed feature spaces to facilitate model interpretation. However, we feel that the results demonstrated in this report show that dimensionality reduction techniques can be

done in a way that is beneficial for practical reasons (e.g. training duration, computational demands) and has low impact on the accuracy of classification.

5 Report Directions

In your report, be sure to:

- describe the problem you are trying to solve.
- describe your datasets and what you did to prepare the data for analysis.
- methodologies you used for analyzing the data
- why you did what you did
- make your conclusions from your analysis. Please be sure to address the business impact (it could be of any domain) of your solution.

Krizhevsky, Alex, Geoffrey Hinton, and others. 2009. "Learning Multiple Layers of Features from Tiny Images."

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE* 86 (11): 2278–2324.

Noever, David, and Samantha E Miller Noever. 2021. "Overhead Mnist: A Benchmark Satellite Dataset." *arXiv Preprint arXiv:2102.04266*.

Xiao, Han, Kashif Rasul, and Roland Vollgraf. 2017. "Fashion-Mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms." *arXiv Preprint arXiv:1708.07747*.