Fashion-MNIST Dataset Analysis
STAT 542 Final Project Report

Team member name(NetID) :
Yizhe He (yizhehe2),
Zixuan Shao (zixuans5),
Chengyan Ji (cji10)

Project description and summary

Fashion.MNIST is a dataset of Zalando's article images that contains a training set of 60000 samples and a testing set of 10000 samples. Each example is 28*28 grayscale image, associated with a label from 10 classes. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels, and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

Main goal of our analysis:

1. Build several multi-class classification model and compare their performance
2. Construct an ensemble model that corporate the individual model

We would like to use statistical learning modeling techniques to identify what class each observation belongs to based on their 784 pixel values. To be specific, we will use multi-class classifiers such as LDA, Random Forest, Boosting and KNN along with ensemble models techniques. We will assess each model's performance based on its accuracy.

Conclusion:
SVM and ensemble model1 provide the same accuracy. Due to computational efficiency, we conclude Support Vector Machine as our best fitted model, followed by Random Forest.

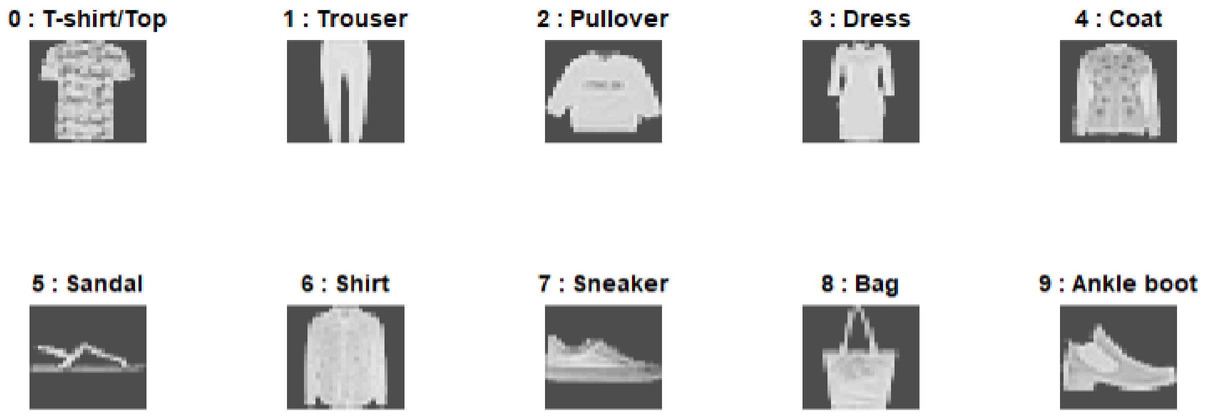


Figure 1. Labels and their corresponding type;

Literature review

We have looked for several different methodologies from the web, including deep-learning models and statistical learning models. Accuracy is measured by MSE for all cases, and the best is using Convolutional Neural Network, with accuracy of 94%. Most models, including Random Forest, Gradient Boosting, Discriminant Analysis and Logistic Regression would achieve accuracy higher than 85%.

1. Random forest algorithm for base-stem genetic algorithm (GA-RF): Apart from traditional random forest, this enhanced algorithm aims to choose the most desirable number of

trees used in random forest. It verifies that when there are too many trees, the accuracy will not increase as fast as we need. The algorithm is based on the claim from Gislason: when a point has reached error convergence and the total precision set based on the square root of the input number is close to the most accurate result, the random forest model is no longer sensitive to the segmentation frequency m. In order to choose the best n (number of trees) and m (segmentation frequency), GA-RF uses a genetic algorithm to reduce computation time. GA-RF uses random forest model to classify samples, and uses a permutation method in genetic algorithms to determine the boundary value of feature screening as the basis for the final determination of feature variables.
Source: Liu. Zheng. Dong. etc. “Classiﬁcation of Fashion Article Images Based on Improved Random Forest and VGG-IE Algorithm”. July 31. 2019. https://www.worldscientific.com/doi/epdf/10.1142/S0218001420510040

2. Logistic Regression: Logistic regression predicted probabilities in the range of ‘0’ and ‘1’. It is a linear model subject to non-linear transformations. Therefore, a linear classifier for the neural network is used. The author does not insert any hidden layers. Instead, the author uses a linear classifier: the input is the linear transformer, i.e. the beta’s; and logistic function is used as the activation function. So the beta’s are fed to the neural network, and the output y’s are transformed through logistic function. If the response is multi-class, like our data set, the regression the final output will then be the probabilities of the response falling into all of the categories. So in this article, the x’s, i.e. the pixels are fed to the neurons, and then the output is fed into the activation function which is the logistic function. The final output is the probability, which we can use to determine the classification with a chosen threshold. Then the loss is calculated. In this case, the logarithmic softmax function is used for calculating the loss. And then gradient descent is used to train the linear classifier.
Source: R, Harsh. “Logistic Regression on Fashion: MNIST using PyTorch”. June 4, 2020. https://blog.jovian.ai/logistic-regression-on-fashion-mnist-e3473ca496f0

Summary Statistics, Data processing and Unsupervised learning

Since this is pretty large data, it is important to start to look at some summaries of the data before doing any analysis. First, we create a frequency table for each class of label in both training and testing data.

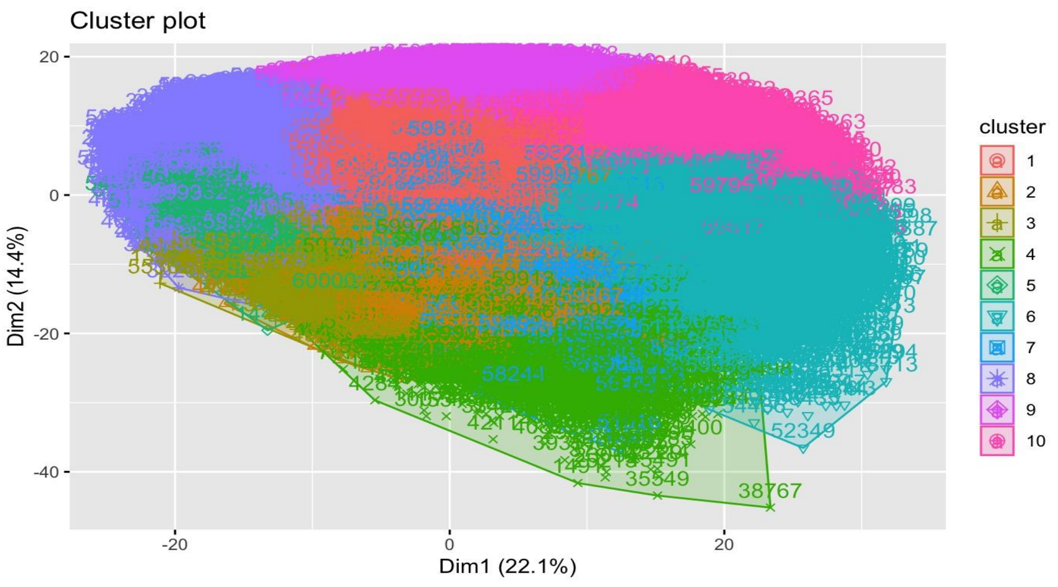
Label	0	1	2	3	4	5	6	7	8	9
Train	6000	6000	6000	6000	6000	6000	6000	6000	6000	6000
Test	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Frequency table

There are no missing values in both the training and testing data set. For further analysis, we scale the training set and use the same scale to scale the testing set. As it shows in the Frequency table, The data is balanced for each class label and therefore will not lead to issues for prediction in future analysis. In this section, we used two types of unsupervised learning methods, specifically, two clustering algorithms in order to help us better understand the overall structure of the data set.

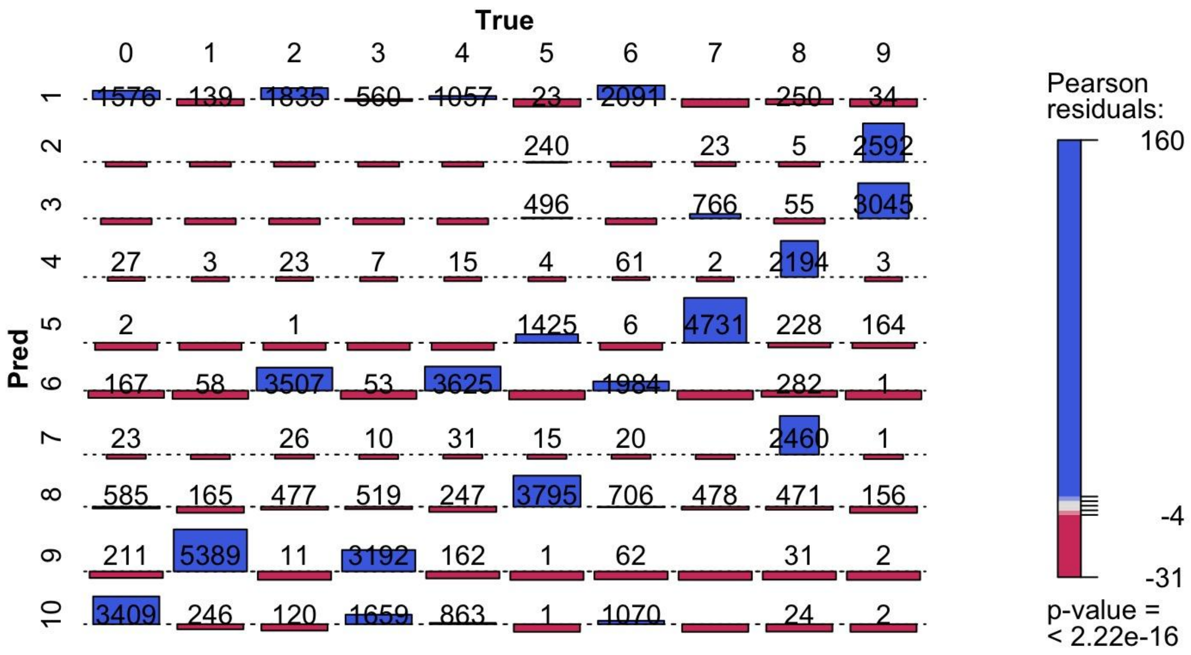
K-means clustering

The first clustering method we performed is K-means clustering. One of the advantages of this algorithm is computation efficient. Considering that there is a huge amount of predictors in this data, K-means probably is the easiest way for clustering. This algorithm needs us to prespecify how many clusters we want to use, since we have the prior knowledge that there are 10 classes of label, we decided to use 10 clusters to match the number of labels.



Cluster plot for 10-means clustering

The plot shows a visualization in 2-dimensional plane of how the data were clustered, we can clearly see that there are many clusters that are gathered together or very close to other clusters. This implies that there are some similar observations but they are not in the same cluster. We also create a frequency table of the true class label in each cluster.



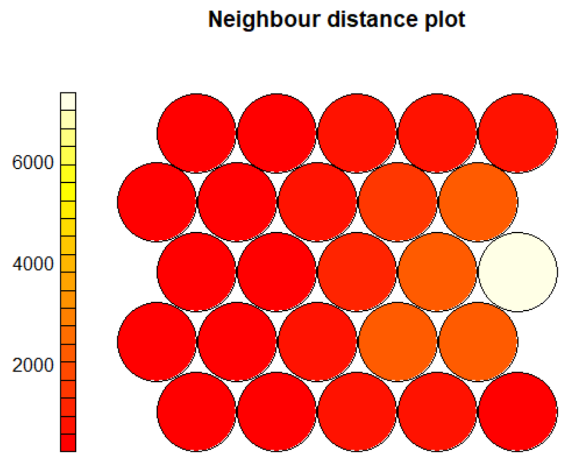
Frequency table in each cluster for 10-means clustering

By looking at the frequency table and summary table, we can easily figure out what is dominating class labels in each of these clusters. The interesting thing we noticed is that label 9 is dominating in both cluster 2 and 3, label 8 is dominating in both 4 and 7. Although label 6 is the dominating label in the first cluster, the numbers of label 0 and label 2 are also quite large. Thus, only 8 out of 10 labels seem to have dissimilarity, but these could also be due the randomness of the K-means algorithm itself.

Self-Organizing Map

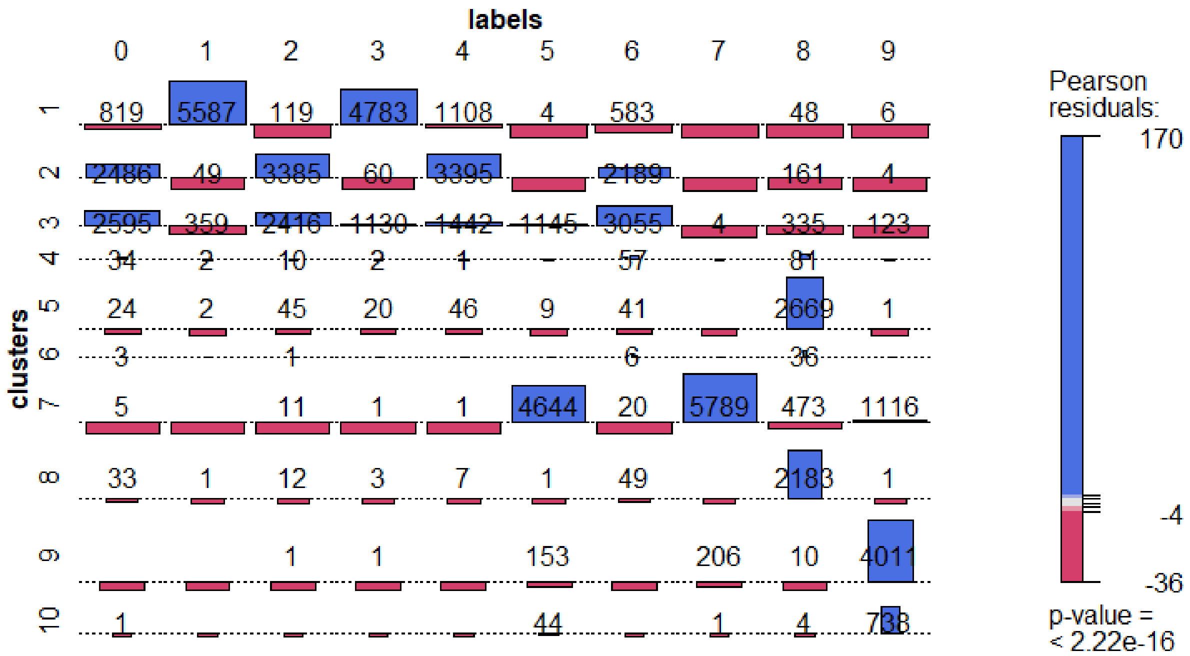
The second clustering method we use is the Self-Organizing Map. Self-Organizing Map(SOM) is an unsupervised machine learning technique used to produce a low-dimensional (typically two-dimensional) representation of a higher dimensional data set while preserving the topological structure of the data. Its advantage is to make high-dimensional data easier to visualize and analyze. Basically, SOM forms a semantic map where similar samples are mapped close together and dissimilar ones apart. Due to efficiency, we use a 5 by 5 grid to construct the map.

The plot shown below, represents how close each knot is. The more similar the color is, the closer each knot is. According to this plot, we can tell there are three specific clusters, given the color of knots shows significant differences(white, orange, and red). We can also tell from the red part in our plot that some of the features in our data are really close, and make it difficult for the map to distinguish them.



Plot of neighbor distance for Self-Organizing Map

Since we have prior knowledge that our data comes from 10 different labels. Therefore, we reduce those 25 grids to 10 clusters according to their distance between others. We then use this to predict our testing set.



Frequency table in each cluster for Self-Organizing Map

Thus, we can compare dominating label cross SOM and K-means:

Cluster	1	2	3	4	5	6	7	8	9	10
Label	6	9	9	8	7	4	8	5	2	1

(k-menas)										
Label (SOM)	1	4	6	8	8	8	7	8	9	9

Summary table of the dominating label in each cluster

As the table shows above, the frequency that each label appears is similar between k-means and SOM, yet they do not provide a clear separation between labels.

Multi-class Classification Model

Method 1: Random Forest

The random forest is a classification algorithm consisting of many decision trees. It uses bagging and features randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

In our random forest model, we fixed the ntree = 100, and 5-fold cross-validation is used for tuning the parameter “mtry” on the training set. The tuning process is shown in the table below:

Mtry	Accuracy	Kappa
1	80.59%	0.7843
5	86.98%	0.8553
10	87.65%	0.8627

Tuning Process for Random Forest

The final value of “mtry” was selected to be 10, since it has the highest value of prediction accuracy and Cohen’s Kappa statistic.

Here, the value of “Kappa” stands for the Cohen’s Kappa statistic. This is another performance measurement that is very useful when we deal with multi-class classification problems. Since in multi-class, the model accuracy does not provide the complete picture of the performance of our classifier. This value basically tells you how much better the classifier is performing over the performance of a classifier that simply guesses at random according to the frequency of each class.

After performing the model on the testing data, the classification confusion table is shown below:

Label	0	1	2	3	4	5	6	7	8	9
0	850	2	8	18	0	0	180	0	1	0
1	0	970	1	9	1	0	2	0	0	0
2	12	6	801	7	63	0	103	0	9	0
3	33	17	11	922	33	0	25	0	1	0
4	3	1	112	23	862	0	87	0	3	0
5	1	1	0	0	0	946	0	15	1	9
6	88	3	57	21	38	0	585	0	8	1
7	0	0	0	0	0	37	0	929	2	40
8	13	0	10	0	3	5	18	0	975	3
9	0	0	0	0	0	12	0	56	0	947

Confusion Table for Random Forest

We can see that category 8 (bag) is predicted the best with 97.5% accuracy while category 6 (shirt) is predicted the worst with only 58.5% accuracy. All other categories have higher than 80% accuracy. We also can see some mix-up between category 0, 2, 4, 6 (T-shirt, Pullover, Coat, Shirt) because they all have long sleeves so their shapes look similar.

Method 2: K-Nearest Neighbors (PCA)

K-Nearest Neighbors is a non-parametric classification algorithm. It determines which class an observation belongs to based on the k nearest observations in the training set. The default euclidean distance is used in the model. In order to reduce dimensionality, we also perform PCA on the data. PCA creates linear combinations of the predictors and hence linearly independent “principal components”. This technique reduces dimensionality and collinearity at the same time. To be exact, the training data is applied PCA to get principal components, which are fed to the KNN as new features, and this combination will produce a KNN model on a reduced-dimension data. The PCA is applied onto test data with the rotation matrix, and PCs from the testing set are fed to the KNN to be classified.

So two parameters need to be tuned for this model. The number of principal components and the K in KNN. Cross validation gives us K = 11 and 80% variation explained cut-off gives us 50 PCs. The model has a quicker runtime and is able to achieve 86% testing accuracy.

The confusion table for PCA-KNN model is shown below:

	0	1	2	3	4	5	6	7	8	9
0	845	5	15	30	1	0	163	0	1	0
1	1	964	0	7	0	0	1	0	0	0
2	10	0	753	10	66	0	78	0	7	0
3	21	21	12	892	24	2	23	0	4	0
4	8	1	124	43	825	0	81	0	6	0
5	0	0	0	0	0	870	1	20	4	13
6	103	9	94	17	82	0	643	0	11	0
7	1	0	0	0	0	79	0	912	5	39
8	9	0	2	1	2	2	10	0	957	0
9	2	0	0	0	0	47	0	68	5	948

Confusion Table for K Nearest Neighbor

We can see that the best prediction occurs at category 1 (Trouser), with an accuracy of 96.4%. The worst prediction is still at category 6 (Shirt), with 64.3% accuracy.

Method 3: Support Vector Machine

The next classification model we used is the support vector machine (SVM). We have learned that SVM can be used for binary classification problems. In that case, SVM is trying to find a hyperplane that maximizes the separation of the data points to their classes in an n-dimensional space. We can use a similar idea to extend the SVM for multi-class classification problems. There are two main approaches, the first one is called One-to-One approach. The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes. This approach breaks down the multi-class problem into multiple binary classification problems. Another approach is called One-to-Rest. In that approach, the breakdown is set to a binary classifier per each class.

In the One-to-One approach, we need to fit a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into account only the points of the two classes in the current split. In the One-to-Rest approach, we need to fit a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups: a group for the class points and a group for all other points.

Back to our dataset, we decided to use the One-to-One approach for this multi-class problem. In general, the One-to-One approach has better accuracy than the One-to-Rest approach while it costs more time to compute. Because One-to-One needs to fit k(k-2)/2 SVMs and One-to-Rest only needs to fit K SVMs. The kernel we choose for SVM is the radial kernel, which is the most commonly used in high dimensional problems. We have not performed cross-validation for SVM because it takes too long for the computer to calculate. The confusion table for testing data is shown below

Label	0	1	2	3	4	5	6	7	8	9
0	859	2	12	21	0	0	147	0	1	0
1	0	975	1	7	0	0	0	0	0	0
2	11	3	834	8	54	0	73	0	4	0
3	26	15	14	920	26	0	25	0	2	0
4	0	0	79	24	871	0	55	0	1	0
5	1	1	0	0	0	935	0	17	2	9
6	93	3	56	19	47	0	682	0	10	0
7	0	0	0	0	0	44	0	946	1	35
8	10	1	4	1	2	9	18	0	977	5
9	0	0	0	0	0	12	0	37	2	951

Confusion Table for Support Vector Machine

The overall accuracy for SVM is 0.895 which is better than random forest. Again, category 6 (shirt) has the worst prediction accuracy with 68.2%, and the rest of the category are above 80% accuracy. It seems like both of our classification models have some trouble predicting label 6. Let’s compare these three classifiers and see some of the results.

Testing data	Accuracy	Kappa
Random Forest	87.87%	0.8652
PCA-KNN	86.09%	0.8454
SVM	89.50%	0.8833

Table of accuracy and Kappa

Category 6 (Shirt)	Sensitivity	Specificity
Random Forest	0.5850	0.9760
PCA-KNN	0.6430	0.9706
SVM	0.6820	0.9747

Table of sensitivity and specificity in Category 6

Ensemble Model and Feature Engineering

Ensemble model is a general approach in statistical and machine learning that we can develop a better predictive model. The idea of ensemble learning is to use multiple individual model’s outputs as new sets of inputs and obtain better predictive performance than could be obtained from any of the individual models alone.

There are three main classes of ensemble methods: bagging, stacking, and boosting. Bagging involves fitting many decision trees on different samples of the same dataset and averaging the predictors. The most famous example of this type of ensemble is the random forest model as we used in the previous section. Stacking involves fitting many different model types on the same data and using another model to learn how to best combine the predictors. Boosting involves adding ensemble members sequentially that correct the predictions made by prior models and outputs a weighted average of the predictors.

Each of these methods has their own advantages and disadvantages. In our analysis, we use boosting as our ensemble method. One of the reasons we choose boosting is because this method uses linear combinations of a set of weak learners to minimize training errors. It also shows how much weight each learner has, this can help us understand which model is the dominating one in this ensemble method.

As the previous part shows, we fit random forest, support vector machine, and K-Nearest neighbor as our supervised model, and K-means and Self-Organizing map as unsupervised models. Here, we choose predicted labels of three supervised models, and predicted clusters of K-means, given the fact that K-means is better separated than the Self-Organizing map, as our input for the second stage. Unsupervised labels will be mapped to the testing data set through the same procedure they are mapped in the training set in order to keep fidelity.

To combine the learners and create a stronger learner, we use Adaptive Boosting algorithm. In AdaBoost, the output of the weak learners is combined into a weighted sum that represents the final output of the boosted classifier. During each iteration in the training process, more accurate learners will be assigned higher coefficients based on the incorrect predictions for each learner, and the training process will be concluded once all observations are correctly predicted or the max iteration is reached.

ensembled model1	Accuarcy	Kappa
training set	92.21%	91.35%
testing set	89.5%	88.33%

Table of accuracy for ensemble model1

As the figure shows above, the accuracy of the testing set and confusion matrix are exactly the same as SVM, and the importance is shown below.

	K-means	PCA+KNN	RandomForest	SVM
importance	0.05%	7.15%	7.97%	84.88%

Contribution of each model for ensemble model1

As the importance shows, SVM dominates our ensemble model, which is not surprising given the high accuracy SVM provided.

We then fit another ensemble model(ensemble model2), instead of having SVM prediction input because of its dominant status, we contain the first five principal components as input, because they explain 50% variability of whole variables.

ensembled model2	Accuarcy	Kappa
training set	88.16%	86.85%
testing set	87.21%	85.79%

Table of accuracy for ensemble model2

	K-means	PCA+KNN	Random Forest	PC1	PC2	PC3	PC4	PC5
importance	0.001%	55.18%	44.74%	0	0.005%	0.011%	0	0.063%

Contribution of each components for ensemble model2

According to the figure, K-Nearest Neighbor and Random Forest are two dominant methods in our ensemble model2, which is not surprising given the accuracy those two models provide. The first five PC and K-means are relatively trivial in this case, yet the accuracy of ensemble model2 is 87.21% for the testing set, which is relatively close to ensembled model1.

We then compare the run time for different models.

	K-means	PCA+KNN	Random Forest	SVM	PCA	Boosting(em1)	Boosting(em2)
run time (min)	3.18	1.46	95.42	122.38	0.98	1.72	3.65

Efficiency for each model

Although our ensemble model has fairly high accuracy in this problem, it brought us attention that the computation efficiency for this method is not very ideal. In practice, we may not want to build a model that could cost more than two hours to fit. Also noticed that besides SVM, the rest of the feature that we constructed has nearly no contribution, especially k-means. Therefore, we need to think of how to construct features carefully in the further analysis. We could also try different ensemble methods such as XGboost or stacking techniques to see if there is any improvement.

Conclusion

Our study focuses on how to utilize different algorithms and models to detect the differences in the image representations of the 10 classes of fashion styles. We saw some general patterns from the exploratory analysis that match our intuitions, including the clustering results showing some classes have large overlapping while some barely touch. Our classification results also show some incompetence on the visually similar classes, despite the method we use. The data set of interest is large in terms of dimensionality. So we observe large discrepancies in the efficiency between different models. In general, non-parametric models such as KNN and models that perform dimensionality reduction see a much higher efficiency. On the other hand, the same models are not as accurate compared to slower ones such as Random Forest and SVM. To address this issue, we use the Adaptive Boosting technique to incorporate the advantages of each model. Our final result from the ensemble method shows an outstanding preference toward SVM, which is less computationally costly compared to Random Forest, and has a better precision compared to KNN.