Statistical Learning Project

I-Lun, Tu

Qihong, Dai

1. Project Description and Summary

- **Goal**

Our goal for this project is to search for a predictive model that can produce the highest accuracy on the Fashion-MNIST dataset in a computationally efficient manner.

- **Approach**

Using various methods and ideas, we built in total 7 models as listed below. In many models we built, Principal Component Analysis (PCA) was utilized first to reduce the dimension to 20 to lower the computational cost.

1. Unsupervised Learning Methods
   - K-means Clustering – 10 clusters
   - Self-organizing Map – 9 clusters
2. Supervised Learning Methods
   - K-Nearest Neighbor Model – 85.45 % of Accuracy
   - Multiclass Logistic Regression – 76.81% % of Accuracy
   - Multiclass Support Vector Machine – 86.24 % of Accuracy
3. Ensemble Model
   - Multiclass SVM and Multiclass Light Gradient Boosting – 86.71 % of Accuracy
   - XGBoost with Loglikelihood Information – 91.05 % of Accuracy

- **Conclusion**

The ensemble methods with boosting frameworks achieved the highest accuracy. This result was expected as tree-based boosting methods were inherently suitable for multiclass classification. In addition, their learning structure and the resulting flexibility in tuning provided more sophisticated models. As a result, in this report, they produced uniformly better prediction accuracy in all classes. Also, with the help of PCA, we were able to train more computationally costly models, such as SVM, in a relatively efficient manner. In conclusion, the highest accuracy that we were able to achieve was higher than the corresponding benchmark gradient boosting method when this data was published. For other more traditional methods, we were able to match the corresponding benchmark performance with dimension-reduced data.

2. Literature Review

- **Best Accuracy**

The best accuracy in the current literature is 98.8%. The model is based on a specific type of architecture of Convolutional Neural Network, named LeNet-5.

- **Paper 1** - Fashion-MNIST classification based on HOG feature descriptor using SVM

The researchers used HOG (Histogram of Oriented Gradient) to preprocess the data and utilize multiclass kernel-based non-linear SVM to classify. The overall accuracy was 86.53%, which was better than the benchmark accuracy of 83.6% achieved through linear SVM.

The researchers divided each image into 16 regions. For each pixel in a region, the direction of change and the intensity of the greyscale were calculated to form two matrixes. Then, a histogram of 9 bins, with each bin having a rotation range of 20 degrees (0 to 180 degrees), was utilized to summarize direction and intensity information from the two matrixes. Thus, in addition to the greyscale information in the original data, direction information was obtained. The basic idea was that local object appearance could be characterized well by the distribution of local intensity gradients and edge direction. The smaller the size of the region was, the more detailed the local shape information would be, and the richer the feature space would become. The researchers considered the computational cost and chose a 4 by 4 grid, and the feature space was enlarged to be 1296 columns (36 X 36) from the original 784 columns (28 X 28).

Multiclass SVM generalized the idea of using hyperplane into using similarity score to separate observations. The predicted class was the one that attained the highest score. Maximizing the margin in this context meant searching for a classifier that could produce the largest difference in similarity score between the predicted class and the rest. In calculating this score, the inner product was used and thus the same kernel tricks, as in the binary SVM scenario, could be utilized directly.

- **Paper 2** - Comparison of Different Models for Clothing Images Classification Studies

XGBoost, as an ensemble learning method, can improve classification performance by iteration of weak classifiers, then learn adaptively. Compared to traditional Gradient Boosting, XGboost used a 2nd order Taylor approximation for better classification in each iteration. With more regularization terms, including eta, delta, alpha, and gamma, it offers better tuning performance. Also, the approximate and histogram algorithm greatly improved speed. Together with weighted quantile sketch, sparsity-aware splitting finding, cache-aware access, and blocks for out-of-core computation, XGBoost realized partial parallel learning, making boosting tree on big dataset become realistic in the real-world application. As a result, XGBoost, as an end-to-end scalable boosting system, is currently widely used by data scientists.

Luo used CNN, VGG16, and XGBoost for Fashion-MNIST classification. For XGBoost, he found the tuning process is fast (i.e. 22 min) and final accuracy reached 89.53%. For the tuning process, Luo used a classification tree as the weak learner, with cross-validation set to 2. After grid-search, he found the optimal hyperparameters: max_depth = 6, eta = 0.3. He also used subsampling and regularization terms to control the bias-variance tradeoff.

3. Summary Statistics and Unsupervised Learning

- **Frequency Table**

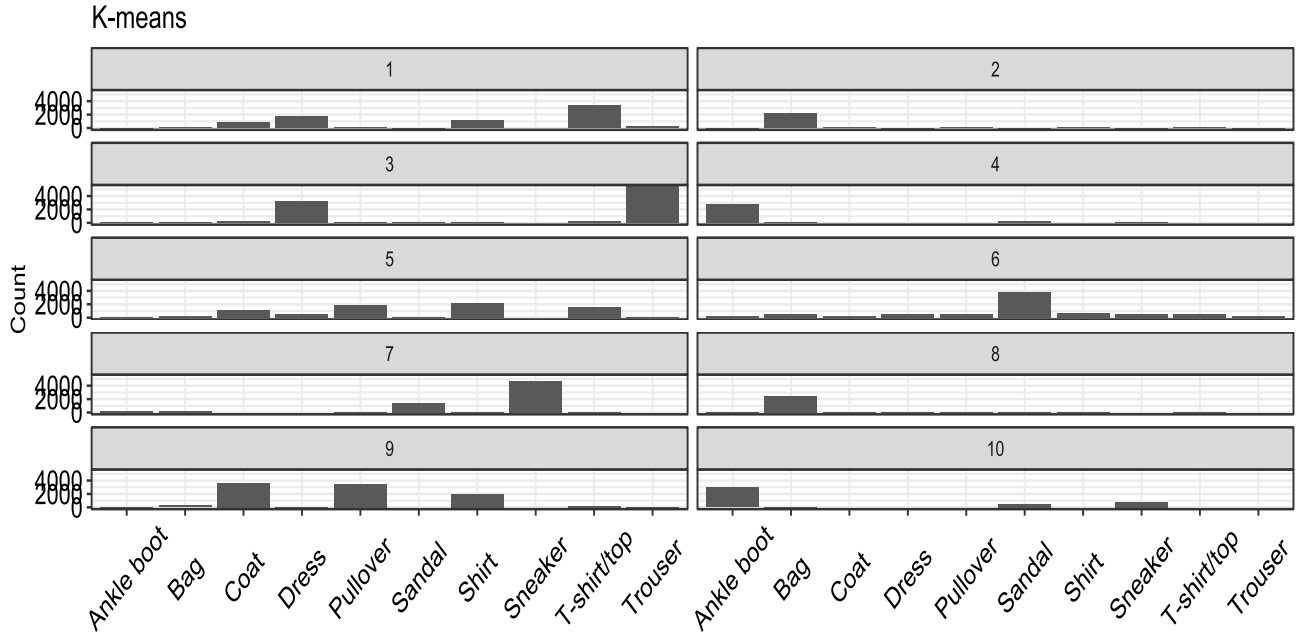
Both training and testing data sets are balanced with the frequency of each class being the same as the tables below showed.

	0	1	2	3	4	5	6	7	8	9
Training	6000	6000	6000	6000	6000	6000	6000	6000	6000	6000
Testing	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

- **Unsupervised learning: K-means and Self-Organizing Map**

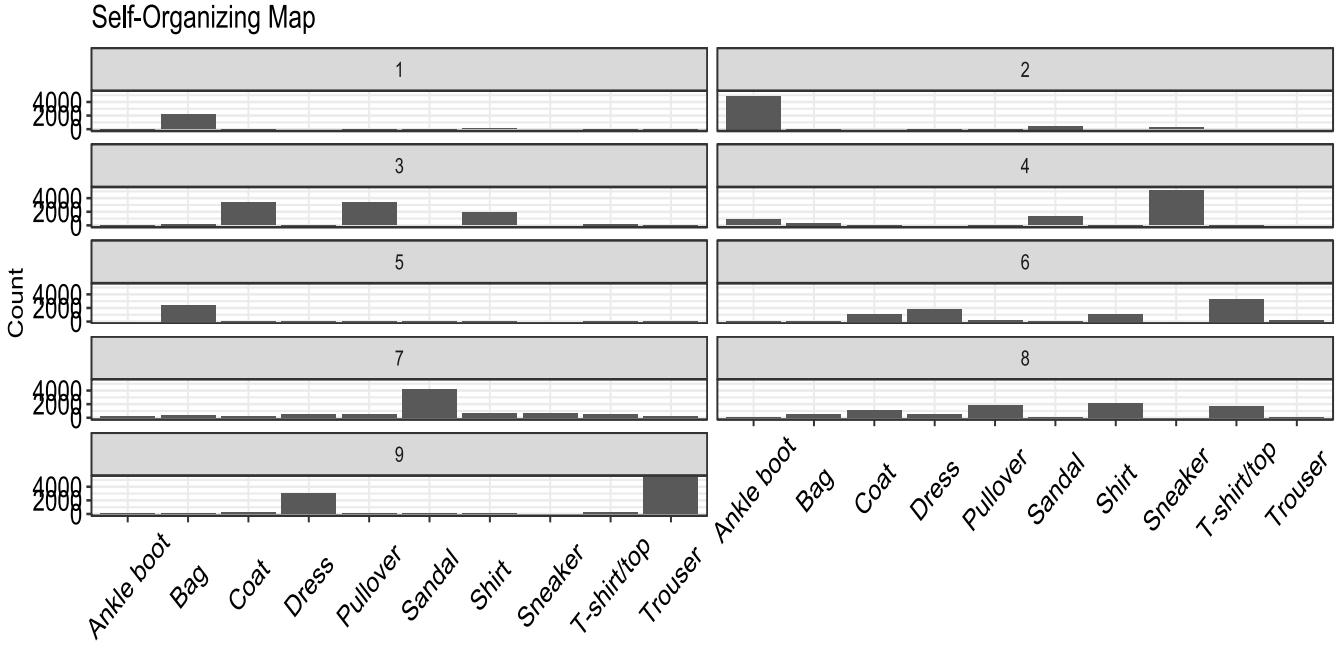
1. K-means

We chose 10 centers after tuning on a grid of 8 to 10 centers on the original data set. Although this was especially computationally costly, we would like to observe how the unsupervised method would interpret the data. From the plot, we noticed K-means was able to accurately separate Ankle boot, Bag, Sneaker, and Sandal, but was not able to perfectly describe the difference among all clothes. For dominating y label in each cluster: 1) T-shirt/top; 2) Bag; 3) Trouser and Dress; 4) Ankle Boot; 5) evenly distributed; 6) Sandal; 7) Sneaker; 8) Bag; 9) Coat and Pullover; and 10) Ankle Boot



2. Self-Organizing Map (SOP)

We chose 9 centers after tuning on the same grid as K-means and we also used the original data set. From the plot, we noticed SOP was similar to K-means that was able to accurately separate Ankle boot, Bag, Sneaker, and Sandal but was not able to perfectly describe the difference among all clothes. For dominating y label in each cluster: 1) Bag; 2) Ankle boot; 3) Coat and Pullover; 4) Sneaker; 5) Bag; 6) evenly distributed; 7) Sandal; 8) evenly distributed; 9) Dress and Trouser.

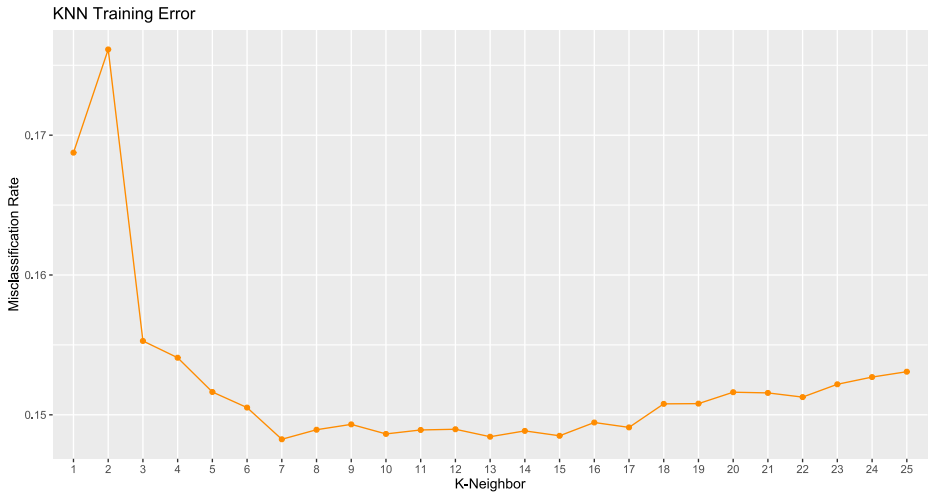


4. Multi-class Classification Model

- **K-Nearest Neighbor Classification**

Considering the computational cost and the curse of dimensionality, we first used PCA (Principal Component Analysis) to select the first 20 components. The first 20 components already contained approximately 80% of the overall standard deviation or information of the original data. Therefore, we thought it was adequate to use only the first 20 components to train the KNN model.

Principal components analysis would be affected by the center and scale of the data. Thus, we first centered the raw data; however, we did not scale it, since we thought that scaling was more suitable for features from different types of information. All the features in this data set were information regarding the intensity of the greyscale. The results showed that when K equals 7, the highest accuracy (85.18%) was obtained. The corresponding testing accuracy of this model was 85.45%.



- We used a grid of K from 1 to 25, and a 5-fold cross-validation for computational efficiency to test the training errors.
- The U-shape was clear. When training, the input data were first normalized for better results.

- **Multiclass Logistic Regression**

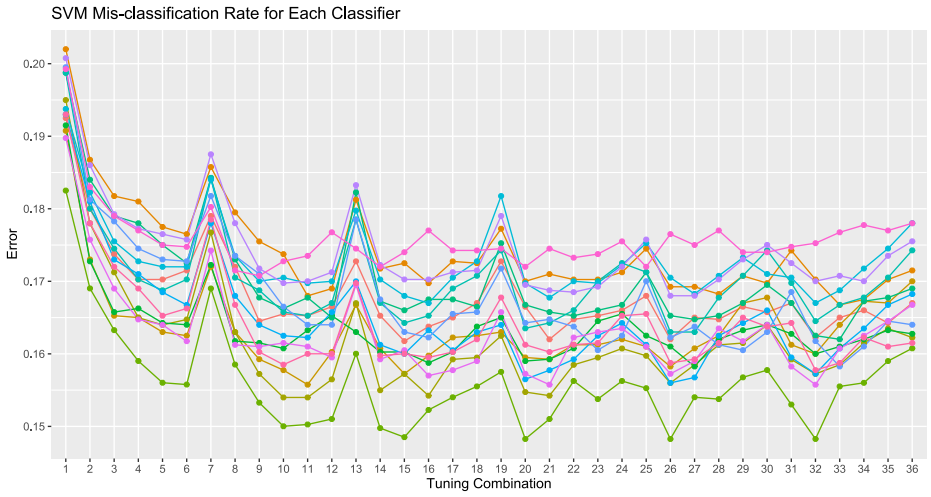
- We used a multiclass logistic regression, which was extended from binary to multi-class classification through One-versus-One (OvO) approaches. (For an explanation of this heuristic approach, please refer to the multiclass SVM section)
- We used Cross-validation via “caret” to specify that we’d like repeated cross-validation (more specifically 10 repeats of 5-fold cross-validation). Grid search for hyperparameter “decay” was conducted from seq(0, 1, by=0.1) and decay=0.3 showed the best performance (77.11%).
- Compared with the previous KNN model, logistic regression did not perform very well, with only 76.81% accuracy. From the confusion matrix (refer to the comparison section), we can verify that logistic regression was not good at differentiating Shirt from T-shirt and Coat. Also, logistic regression has quite a high error rate when classifying Pullover from Coat and Shirt.

- **Extension of Binary SVM to Multi-class SVM**

There are two heuristic ways to extend to multi-class classification. That is, one versus one (OvO) and one versus rest (OvR) approaches. For the OvO approach, k(k-1)/2 number of classifiers have to be trained and the predicted label is the majority of the vote from each classifier (Max Wins strategy). The advantage is that this method is more robust in its prediction, but the drawback is that this approach is more computationally expensive. The OvR approach, on the other hand, is more efficient, but due to the unbalanced positive and negative instances, it is suggested in the literature that this might have a negative effect on the results. As we mentioned in the review section, SVM can be generalized in a more non-trivial way. However, this approach involved solving a larger quadratic optimization problem and therefore would be computationally expensive as well.

OvO approach was used for this problem. To reduce the computational cost, we randomly split the PCA-reduced data into 15 subsets evenly. We trained 15 SVM classifiers. We voted to decide the predicted label (randomly broke ties). The literature showed that this method would reduce the computational complexity from Ω (n^2) to Ω (n^2/15). More importantly, the literature suggested that the predictive accuracy would maintain, though bootstrapping was utilized in the literature.

For each SVM classifier, we used the same non-linear radial-based Gaussian kernel, since the consensus was that the Gaussian kernel was effective for different data and classification purposes. Incorporating non-linearity would make our model much more flexible compared with a linear kernel.



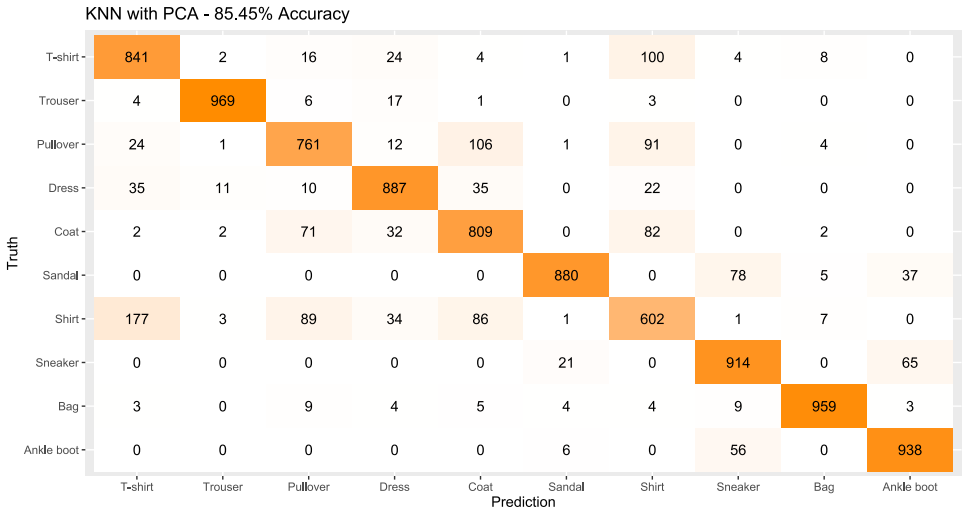
- We used 5-fold cross-validation and a grid (normalized data), with C evenly spaced from 1 to 15 (6 points), and sigma from 0.01 to 0.06 (6 points).
- The literature suggested that anywhere between 0.1 and 0.9 quantiles of the $\| x - x' \|$ statistics would produce good results.

We obtained the lowest training errors at the bottom of the U-shaped errors. The spikes appeared when a new combination of C was used. For each smaller classifier, the testing accuracy was no more than 85% (see below tables). However, the accuracy of the combined model was a bit higher. This result was expected since this method was like bagging weaker classifiers into a stronger one.

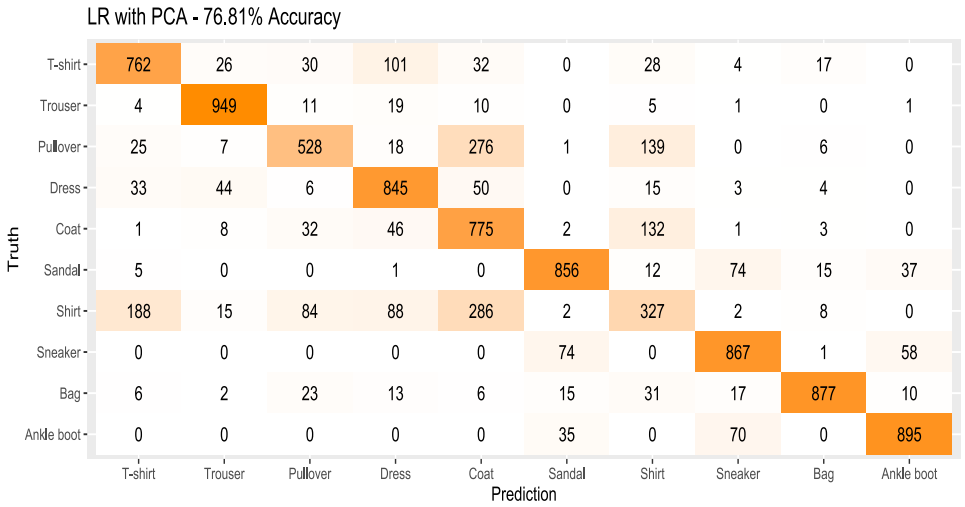
	1	2	3	4	5	6	7	8
Training	83.82%	83.33%	84.42%	84.60%	85.17%	84.17%	83.80%	83.70%
Testing	84.15%	83.92%	83.97%	84.34%	84.50%	83.87%	84.23%	84.12%
	9	10	11	12	13	14	15	Combined
Training	83.30%	84.40%	84.17%	83.20%	84.42%	82.92%	84.22%	NA
Testing	84.00%	84.62%	84.02%	84.36%	84.36%	83.87%	84.48%	86.19%

- **Comparison among Multiclass Classification Models**

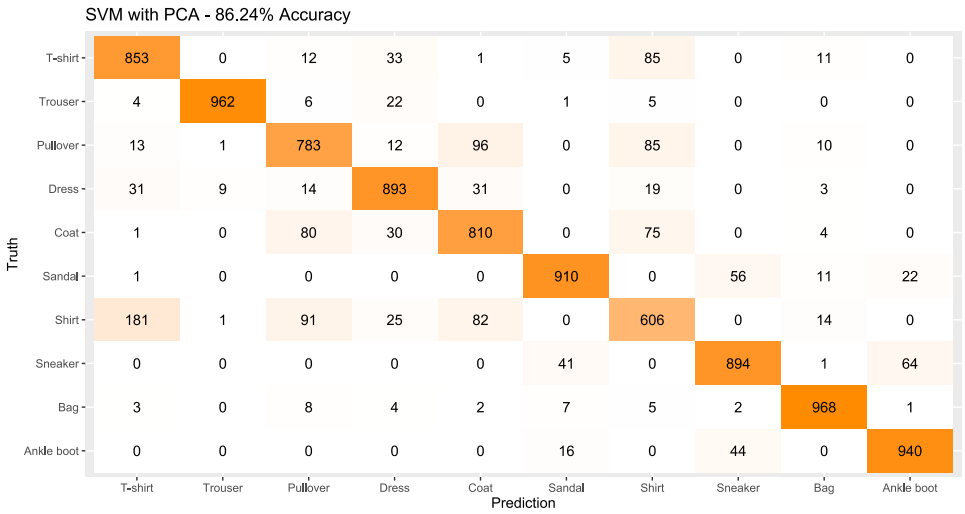
  Multiclass SVM achieved the highest accuracy; however, it also has the highest computational cost.



➢ The KNN model performed reasonably well considering the complexity of the data and the relatively simple method of the KNN approach. The reason may be that the original high dimensional data can be adequately represented by the PCA reduced low dimension data.



➢ Logistic regression suffered most when classifying shirt-like observations. Possible reasons may be that those observations were highly correlated. Thus, a regression-based method was more susceptible to this setting even after tuning.



➢ Multiclass SVM achieved the highest accuracy. We believed this boost of accuracy was because of aggregating smaller classifiers, as the performance of each classifier separately was quite similar to the KNN model.

4. Ensemble Model and Feature Engineering

For the ensemble model, we retained the same use of principal component analysis to reduce the dimension of data to 20 dimensions. From the results of the SVM model, we observed that overall speaking, the model performed worse in classifying T-shirts, pullovers, coats, and shirts. In addition, when the model did misclassify observations in this group, it would most likely still place them within this group. This result was expected since it was much easier to tell shirts apart from shoes or bags than other kinds of shirts. Therefore, we tried to improve the model specifically in this respect.

We aimed to train a specific model that was more capable of predicting one shirt from another. The idea of this ensemble model was as follows:

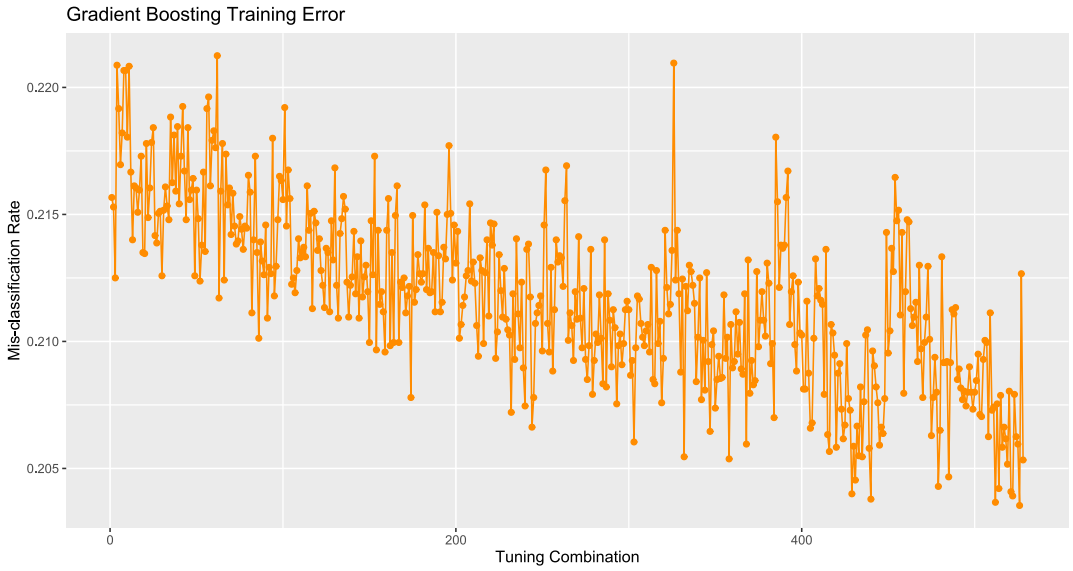
1. **Data:** use principal analysis to reduce the dataset to 20 dimensions
2. **First Stage:** use the previous SVM model to classify and filter out the observations which the SVM model classified as T-shirts, pullovers, coats, and shirts.
3. **Second Stage:** use the specialized model to classify the "difficult" observations again.
4. **Final Prediction:** combine the classification results of the other easier observations in the first stage and the classification results of the difficult observations from the second stage.

To build the specialized classifier, we first filtered all the observations of T-shirts, pullovers, coats, and shirts from the training dataset. We used the tree-based Light Gradient Boosting methods. The main difference of this method from the general gradient boosting methods was that in addition to growing decision trees level by level, it also grew trees in a leaf-wise manner. In effect, it had a lot more variations, and therefore, it could potentially fit the data better.

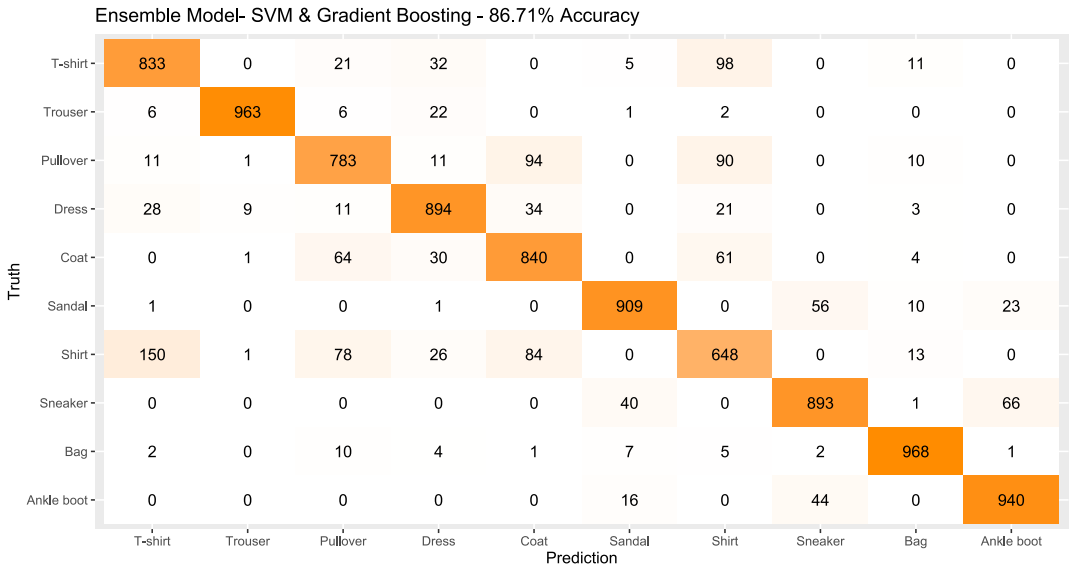
However, it was also more prone to overfit the data. Therefore, we made use of a finer grid of tuning variables, and we generated 1500 trees for each iteration with the total number of iterations being 375. For each model, we used 5-fold repeated cross-validation 30 times, and if the accuracy did not improve over 25 iterations, we stopped the process and moved on to the next. The main variables to control the complexity and their tuning ranges are as follows:

1. **Maximum depth:** from 5 to 12 with step size 1 (12 points)
2. **Maximum number of leaves:** from 20 to 160 with step size 20 (12 points)
3. **Minimum number of observations in a leaf:** from 160 to 20 with step size 20 (8 points)
4. We added an extra combination for more complex models:
    - Maximum Depth: 13 to 16 (4 points)
    - Maximum number of leaves: 180 to 240 (4 points)
    - Minimum observations in a leaf: 20 (1 point)
5. The expanded grid arranged the corresponding models from simple to complex, with a total number of 528 tuning combinations.

Within the gradient boosting structure, the decision was based on multiclass logistics regression and the objective was to minimize the multi-logloss. The resulting value would be mapped through the logit link function, and for each class (4 classes in this case), there will be a probability. We selected the highest probability as the predicted label.



From the above graph, we observed that as the model became more complex, the classification error lowered from 22 % to 20.5%. There was no clear U-shaped trade-off to be observed. However, the classification error was adequately lowered as a result of the tuning process, and the result suggested that a more complex model did not perform better. The model with the lowest error was structured by 7 levels, 200 leaves, and a minimum of 20 data in a leaf and achieved a 20.53% of training error.



The overall classification rate increased. However, this improvement was not uniform for each class, and there was some trade-off between T-shirt and Shirt. The main improvement was that in the second stage, the model was more adept at telling the difference of coats from other classes. The possible drawback of using a specific dataset may be that the model was not able to learn the difference between a particular set of classes and other easier-to-classify classes. We essentially used a hard “cut-off” rule to discriminate the data. Therefore, we would like to propose another way to utilize a more continuous method to discriminate the data.

- **Second Ensemble Model**

➢ First layer:

1. PCA with the first 10 dimensions as the first 10 features

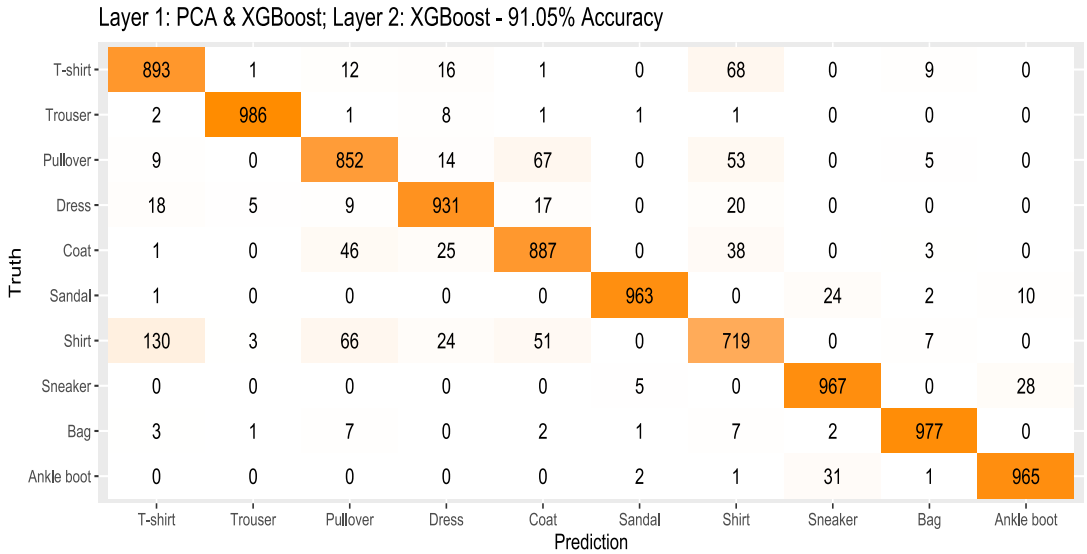
2. XGBoost classification using hyperparameter (eta = 0.1, gamma = 0, max_depth = 6, min_child_weight = 1, subsample = 1, colsample_bytree = 1). By doing these, we sought to find the optimal balance between eta (i.e. learning rate) and M (i.e. number of iterations). Then we established a base model with default settings of other parameters to generate the initial predicted likelihood of each sample on 0-9 labels. Thus, the probability of each sample classified as 0-9 labels constituted another 10 features. This additional information provided some suggestions for the later model of how likely or unlikely each observation belonged to a certain class.

➢ Second layer:

Based on 20 features from the first layer, we used XGBoost multi-classification as the second model. First, we conducted random hyperparameter search for 500 times, with max_depth = sample(3:10, 1), eta = runif(1, .01, .3), subsample = runif(1, .7, 1), colsample_bytree = runif(1, .6, 1), min_child_weight = sample(0:10, 1). The combination of hyperparameters resulting in a higher prediction accuracy 91.05%. The selected model was: max_depth = 9, eta = 0.08407436, subsample = 0.8386878, colsample_bytree = 0.9760058, min_child_weight = 3.

➢ Results:

The prediction accuracy passed 90%. We observed a uniform improvement in each class. Still, Shirt was the label most likely to be misclassified as T-shirt, Pullover, and Coat. In addition to the previously mentioned advantage of adding continuous data, such as log-likelihood information, we thought this improvement was also because the model took into consideration and learned all the differences at once, whether it was between-group or within-group of “shirt-like” or “shoe-like” classes.



Reference

- **Literature Review**

➢ Fashion-MNIST classification based on HOG feature descriptor using SVM

Kayed, Mohammed, Anter, Ahmed, Mohamed, Hadeer. Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture. 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE) Innovative Trends in Communication and Computer Engineering (ITCE), 2020 International Conference on. :238-243 Feb, 2020

Greeshma, K. V., & Sreekumar, K. Fashion-MNIST classification based on HOG feature descriptor using SVM. International. Journal of Innovative Technology and Exploring Engineering, 2019, 8(5), 960–962.

Herwin Alayn Huillcen Baca, Flor de Luz Palomino Valdivia, Ivan Soria Solis. Comparison of HOG Feature Descriptor using SVM with Convolutional Neural Network using SVM for Image Classification. REVISTA DE INVESTIGACIÓN EN CIENCIA, TECNOLOGÍA Y SOCIEDAD (CTS-UNAJMA), vol1, N°2,2020

Han Xiao, Kashif Rasul, Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms

Crammer, Koby, Singer, Yoram. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. Journal of Machine Learning Research. Spring 2002

➢ Comparison of Different Models for Clothing Images Classification Studies

Luo, J. (2020, October). Comparison of Different Models for Clothing Images Classification Studies. In Proceedings of the 2nd International Conference on Artificial Intelligence and Advanced Manufacture (pp. 13-17).