STAT 542

Final Project Report

Instructor: Ruoqing Zhu

Team member: Fengqing Qiu (fqiu3)

Dingjia Chen (dingjia2)

Yilun Fu (yilunf2)

May 5, 2022

1 Overview

The Fashion Mnist is a widely used dataset with a large amount of dimensions. It is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 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.

Figure 1: Example of dataset(first 5 classes)

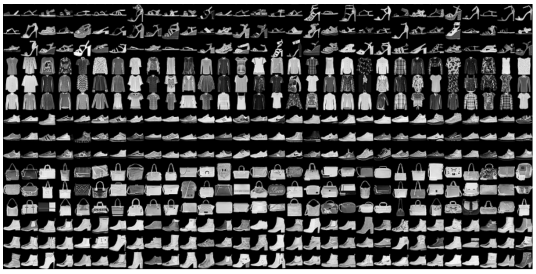
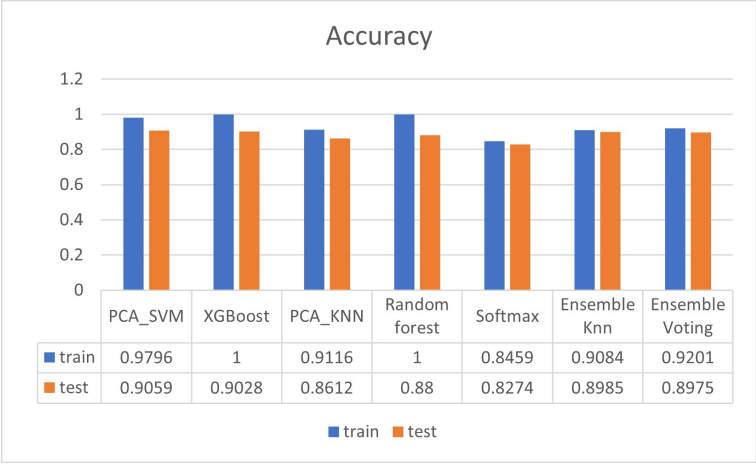


Figure 2: Example of dataset(Last 5 classes)

The motivation of this project is to apply the knowledge we learned from classes comprehensively and flexibly. Therefore, the project is aiming to create a dataset that can accurately measure the data in the dataset classification model. To realise this goal, we studied various clustering algorithms and classification models, observed and evaluated their performance, and finally developed two ensemble models with the highest accuracy we can reach.

For the accomplishment of the project, we found two clustering algorithms, which were helpful in data separation. The prediction accuracies in the testing set of classification models we applied were 90.28%(XGBoost), 88.00%(Random Forest), 90.59%(SVM) and 86.12%(KNN). We also modified a binary classification model, Logistic, made it a multi-class classification model, and reached an accuracy of 82.74% in the testing set. We also developed two ensemble models with an accuracy of 89.85% and 89.75% in the testing set.



	PCA_SVM	XGBoost	PCA_KNN	Random forest	Softmax	Ensemble Knn	Ensemble Voting
train	0.9796	1	0.9116	1	0.8459	0.9084	0.9201
test	0.9059	0.9028	0.8612	0.88	0.8274	0.8985	0.8975

Figure 3: Summary of all models

2 Literature Review

There are many researchers devoting themselves to constructing a better model with less prediction error. Nowadays, the classification accuracy of proposed models can reach over 96%. Foret et al. (2020) [1] achieved an accuracy of 96.61%, introducing the sharpness function into the target loss

function. Tanveer et al.(2021) [2] obtained an accuracy of 96.91% in the fashion mnist dataset. They used fixed operations to fine-tune DARTS. However, both these two models are deep learning models.

In the study of Hua and Mo (2020) [3], they proposed a multi-layer self-organizing map to improve the performance of clustering. They connected multiple SOM networks using a cascaded manner, indicating that the outputs of the last SOM networks are combined and become the input of the next SOM. It led to the ensemble of multiple SOM models.

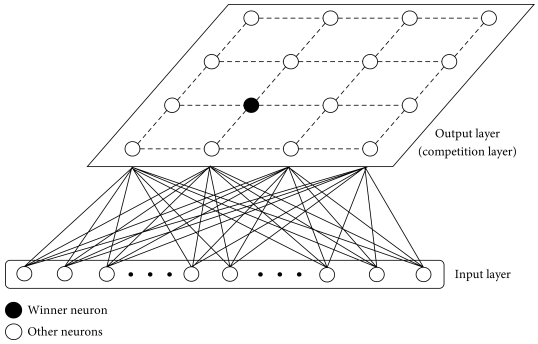


Figure 4: Structure of 2-layer cascaded SOM network

The SOM network can create a map between high-dimensional data and low-dimensional data. Also, due to the insensitivity of high-dimensional data to minor changes in some dimensions, it succeeded in ignoring some error outputs of the formal SOM network, which helped in improving the accuracy. The accuracy of pattern recognition increased by 4% on Fashion Mnist with the cascaded SOM network of 2-layer.

Liao and Couillet (2019) [4] solved the limitation of the least squares support vector machine in large dimensional analysis. They assumed that the input was a two-class Gaussian mixture model,i.e the input data has two means μ_1, μ_2 and two covariance matrices C_1, C_2.So for the training data (x_i, y_i), x_i either belongs to class one or belongs to class two, which depends on y_i is 1 or -1. Map x_i to the feature space H with the function $\psi : x_i \mapsto \psi(x_i) \in H$. then we can find a function $g(x) = w^T\psi(x) + b$ to minimizes the training errors $e_i = y_i(w^T\psi(x_i) + b)$. It leads to the corresponding objective loss function with penalty:

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min} \quad & L(\mathbf{w}, e) = \|\mathbf{w}\|^2 + \frac{\gamma}{n}\sum_{i=1}^{n} e_i^2 \\ \text{such that} \quad & y_i = \mathbf{w}^{\mathsf{T}}\boldsymbol{\varphi}(\mathbf{x}_i) + b + e_i,\ i = 1, \ldots, n \end{aligned} \tag{1}$$

The solution with Lagrange multipliers is $w = \sum_{i=1}^{n} \alpha_1\psi(x_i)$. Once it's solve, the decision function can be represented as:

$$g(\mathbf{x}) = \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{k}(\mathbf{x}) + b \tag{2}$$

where b is solved with corresponding α, $k(x) = \psi(x)^T\psi(x_j))_{j=1}^{b} \in R^n$. Liao and Couillet applied this method to the Fashion-Mnist dataset, obtaining a series of classification errors that were close to 0.

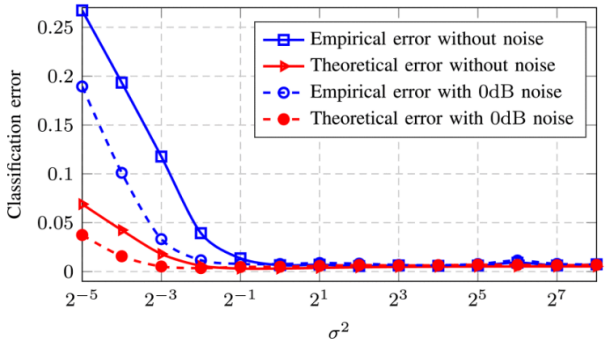


Figure 5: Model behaviour in Fashion Mnist dataset

3 Clustering

3.1 Data Loading and Frequency Table

All data are download from the Kaggle website. In order to make subsequent operations more convenient and accurate, we performed normalization on the data as soon as the data was loaded in the program.

Table 1: Frequency table of the training and testing data

training set		testing set	
label	count	label	count
0	6000	0	1000
1	6000	1	1000
2	6000	2	1000
3	6000	3	1000
4	6000	4	1000
5	6000	5	1000
6	6000	6	1000
7	6000	7	1000
8	6000	8	1000
9	6000	9	1000

3.2 PCA

Before we get into modeling, high dimensional problem must be considered since we have 784 different pixels. The parameters pixels are all on the same scale and evenly distributed between 0 and the max value of 255. There is no need to scale them and centering would be the only thing to be set during PCA.

Considering that the main purpose of PCA is to reduce the dimension and speed up the algorithm here. But at the same time, we want to ensure that PCA maximizes the overall variance of the data. To cover over 98% of the total variance explained by raw data, 349 principle components are used.

3.3 t-SNE

t-SNE is something called nonlinear dimensionality reduction. This algorithm allows us to separate data that cannot be separated by any straight line and to visualize high-dimensional data by giving each datapoint a location in a two or three-dimensional map. We can see there are even some subclusters in one big cluster.

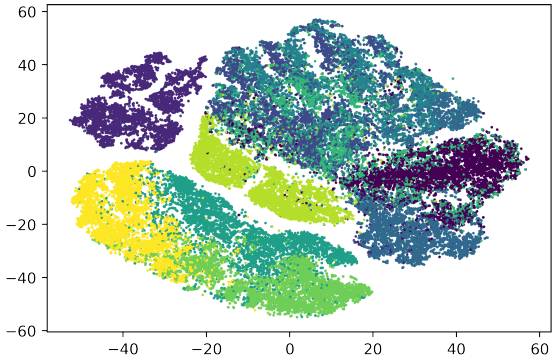


Figure 6: t-SNE of training set

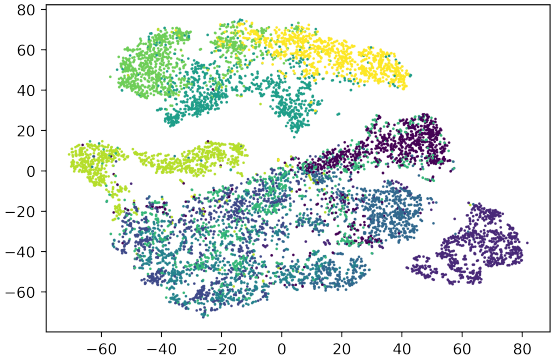


Figure 7: t-SNE of testing set

3.4 SOM

A self-organizing map (SOM) is a clustering technique that helps you uncover categories in large datasets, such as to find customer profiles based on a list of past purchases. It is a special breed of unsupervised neural networks, where neurons (also called nodes or reference vectors) are arranged in a single, 2-dimensional grid, which can take the shape of either rectangles or hexagons.

3.4.1 Heatmaps

We could identify observations with cells on the map by assigning each observation to the cell with representative vector closest to that data point's stat line. The "count" type SOM does exactly this, and creates a heatmap based on the number of players assigned to each cell.

It can help us to visualize the count of how many samples are mapped to each unit on the map. This metric can also be used as a measure of map quality. Large values in some map areas suggests that a larger map would be beneficial. Empty nodes indicate that your map size is too big for the number of samples.

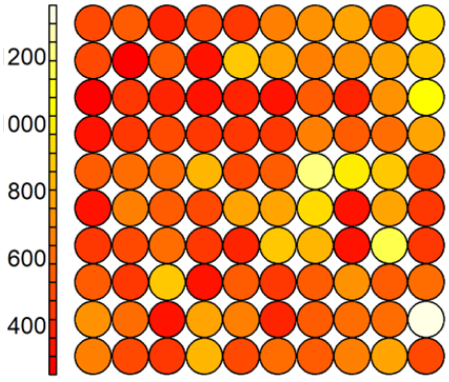


Figure 8: Count plot

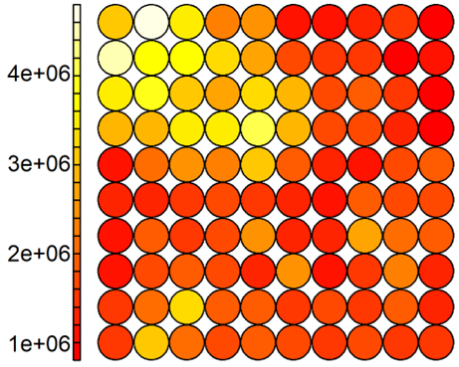


Figure 9: Mapping distance plot

Mapping Distance Plot indicates that the cells are colored depending on the overall distance to their nearest neighbors, which allows us to visualize how far apart different features are in the higher dimensional space.

3.4.2 Model Fitting

Plotting using type = "codes" we get the standard side by side visualization the observation stats (Codes X) and the observation position prediction (Codes Y).

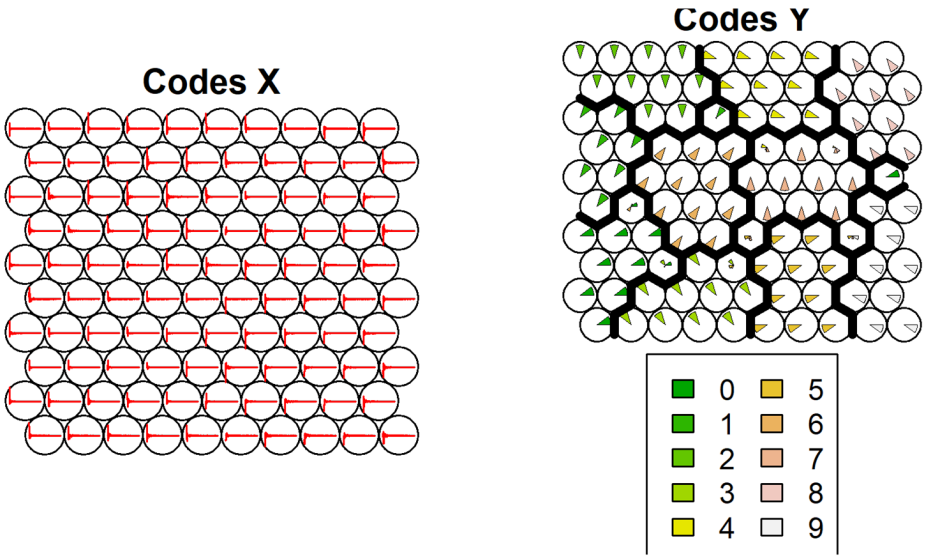


Figure 10: Fitting results

We cannot see a clear plot for the left one since we have too many principle components or pixels here. If we only have several pixels then the plot of codebook vectors corresponding to X will generate

a great plot to visualize which principle components or pixels play in important roles for a specific cluster.

The plot on the right provides a view with respect to Y. Remember we used 100 clusters at the beginning so there are totally 100 circles. To get the cluster boundaries, hierarchical clustering is used to cluster the codebook vectors into 10 different classes in accordance with having 10 classes of Fashion-MNIST dataset. In more details, distance matrix is computed at the codebook y vectors' level with respect to clusters (100 in total) and followed by hierarchical clustering. The colors in the plot are all about different classes set to clusters. So as a conclusion, roughly 14 clusters are merged from 100 clusters from the plot. One thing that need to be mentioned is the labels on the plot of Codes Y represent the true class for each cluster (1-100).

3.4.3 Prediction and Classification

In general, SOM is used for clustering and this is an unsupervised learning method. However, we have also tried to use SOM for classification of the test data which can be regarded as an extension of our analysis. In a nutshell, the accuracy of this model is 0.7895 with 95% CI (0.7814, 0.7975).

Table 2: Confusion matrix of SOM

		Reference									
	Class	0	1	2	3	4	5	6	7	8	9
Prediction	0	771	13	7	57	3	2	215	0	2	0
	1	0	936	3	11	2	0	7	0	2	0
	2	22	8	666	5	128	1	123	0	18	0
	3	73	34	17	853	41	2	48	0	17	0
	4	7	1	159	43	693	0	96	0	6	0
	5	1	1	0	0	0	855	0	93	11	43
	6	110	7	139	27	128	6	498	0	15	1
	7	1	0	0	0	0	79	0	828	16	72
	8	14	0	9	4	5	10	13	2	912	1
	9	1	0	0	0	0	45	0	77	1	883

3.5 K-Means

K-means clustering aims to assign a number of cluster centers based on the number of clusters given. Each data point is assigned to the cluster whose center is nearest to it. The algorithm aims to minimize the squared Euclidean distances between the observation and the center of the cluster to which it belongs.

To find the optimal number of clusters, we checked the distortion score for 2 to 30 clusters, and accordingly, we decided to separate the train data into 8 clusters.

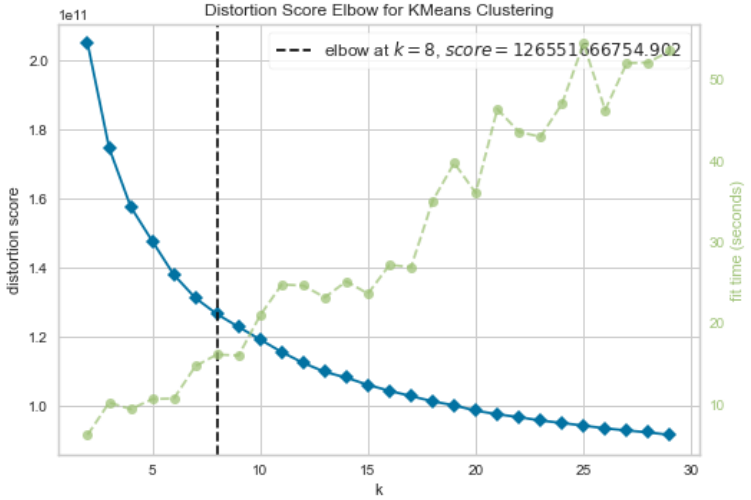


Figure 11: Elbow method result

After applying K-Means clustering, to be more aware of the algorithm's performance, we generated a 3D plot to see the distribution of each cluster. Meanwhile, we generated a figure to exactly know the labels' distribution in each cluster.

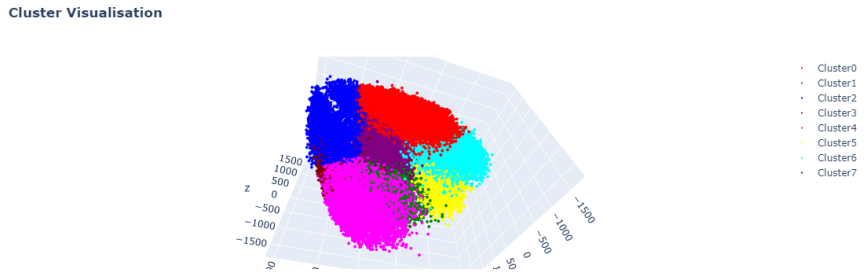


Figure 12: 3D distribution

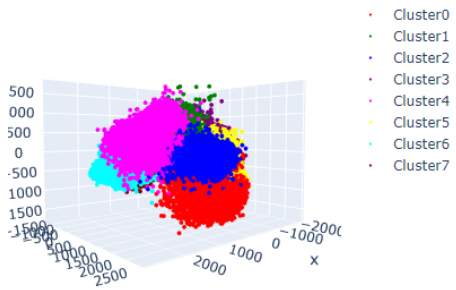


Figure 13: 3D distribution from different angle

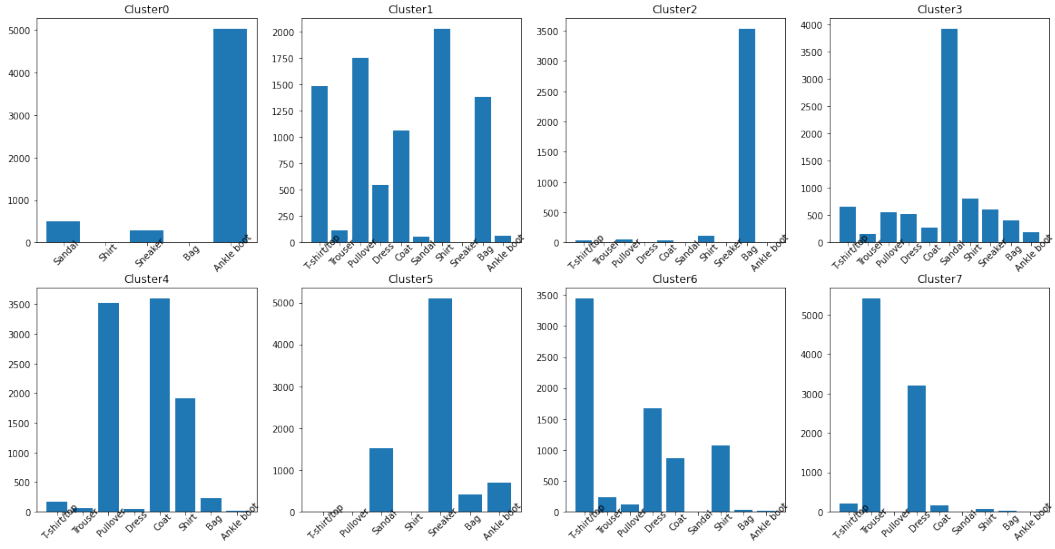


Figure 14: Clusters' component

From the above figure, we can conclude that:

- Only in Cluster 1 and Cluster 4, we cannot decide which label is dominating.

- For the rest of the clusters, the corresponding dominating labels are Ankle boot(0), Bag(2), Sandal(3), Sneaker(5), T-shirt/top(6), and Trouser(7).
- Our clusters help to separate the labels, though we still have limitations.

4 Classification

4.1 Gradient Boost

Boosting is one of the most powerful learning ideas introduced in the last twenty years. It was originally designed for classification problems, but it can profitably be extended to regression as well. The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee." Here we used the decision tree as our weak learner and set XGBoost to do multiclass classification using the softmax objective.

The boosted tree model is a sum of such trees, $f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$, induced in a forward stagewise manner. At each step in the forward stagewise procedure one must solve

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)) \tag{3}$$

Because the traditional 'gbm' package runs too slowly when we first tried it, and specifically it took us ten hours to get the results. Then we switch to using another package called 'XGBoost' which automatically do parallel computation on Windows and Linux, with OpenMP.

When PCA is used, the test accuracy of XGBoost model is lower than when raw data is used without PCA. The same situation occurs when we fit random forest. When it comes to the Softmax model, there isn't much of a difference between using PCA and not using PCA. However, Knn or SVM with PCA performs better than one with raw data.

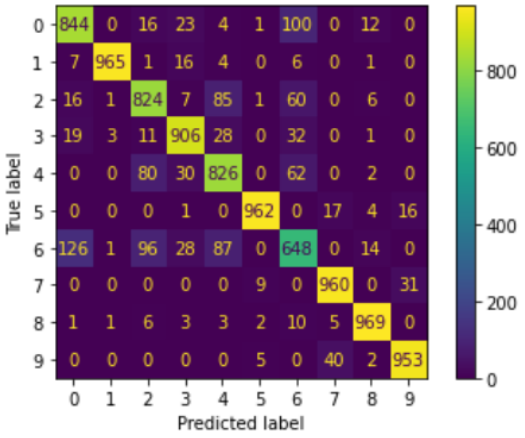


Figure 15: Confusion matrix of classical 'gbm'

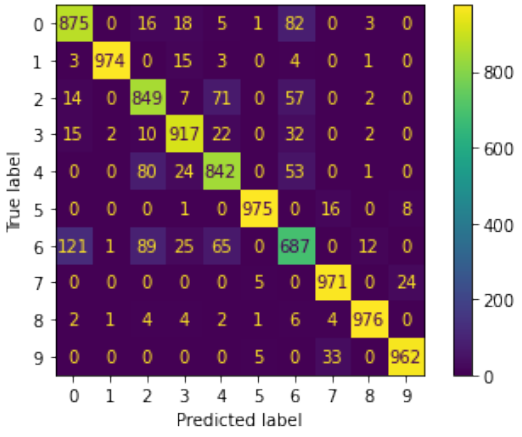


Figure 16: Confusion matrix of XGBoost

The figure on the left is the confusion matrix of classical 'gbm', and it has test accuracy 0.8857. The right one is by using the Extreme Gradient Boosting package, which produces an excellent result with test accuracy 0.9028.

4.2 Random Forest

Random forests is a substantial modification of bagging that builds a large collection of de-correlated trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune.

The number of decision trees can be tuned to get the best prediction accuracy. No matter how many estimators we use, train accuracy is always 1 which is perfectly fitted. As for the test accuracy, it reaches the highest one when 900 trees were used in the model.

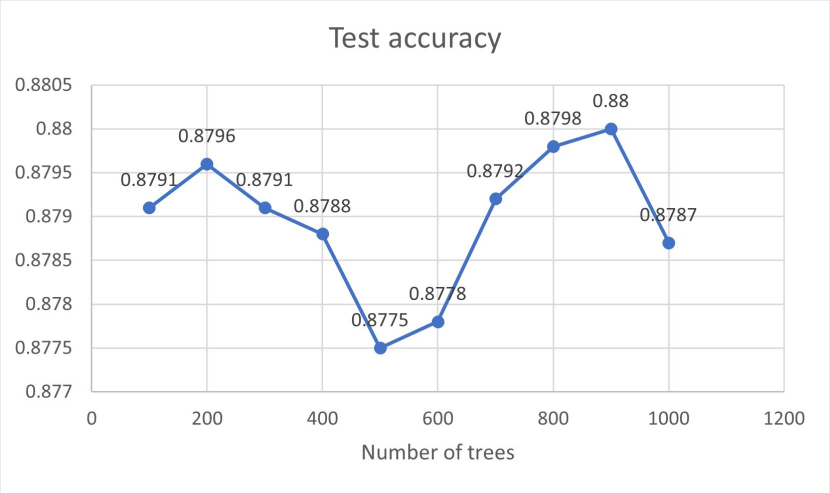


Figure 17: Test accuracy(Random forest)

From the result, we should choose the number to be 900 based on test accuracy.

4.3 KNN

K-Nearest Neighbor (KNN) is a nonparametric method that predicts a target point with the average of nearby observations in the training data. Because the operation speed of knn is not particularly slow, we tried two versions, which were used with the original data and the data after PCA to fit the model.

The number of neighbors is our tuning parameter with a set of 1 to 10.

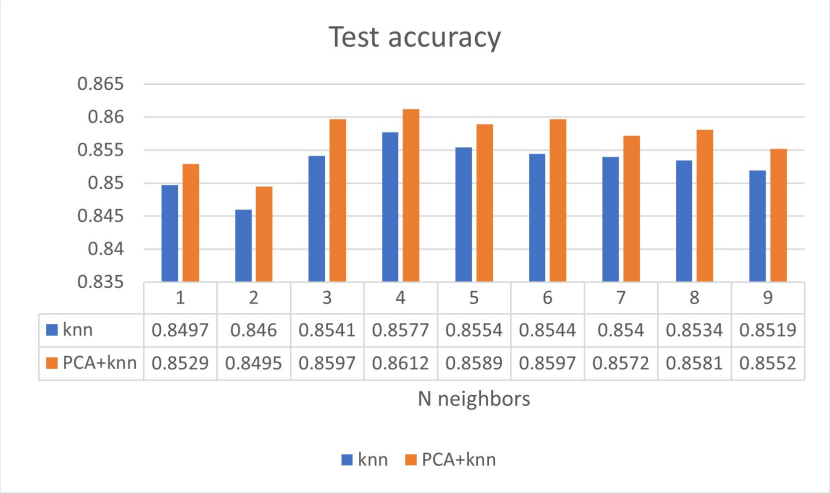


Figure 18: Test accuracy(KNN)

Two different versions give the same result, so we choose k = 4 as our tuning result. And this will be used in ensemble method.

4.4 SVM

Support Vector Machines (SVM) are popularly and widely used for classification problems in machine learning. In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.

When training an SVM with the Radial Basis Function (RBF) kernel, two parameters must be considered: C and gamma. The parameter C, common to all SVM kernels, trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly. Gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected. Here we use default gamma but tune the parameter C.

$$\gamma_{default} = \frac{1}{n_{features} \times Var(X)} \tag{4}$$

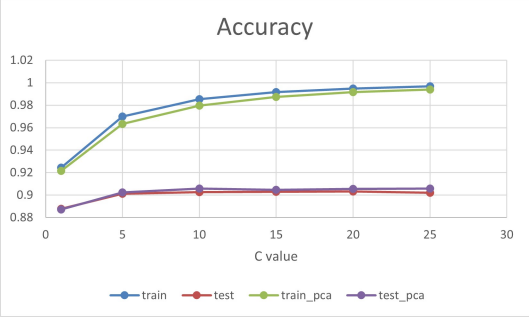


Figure 19: PCA vs. no PCA

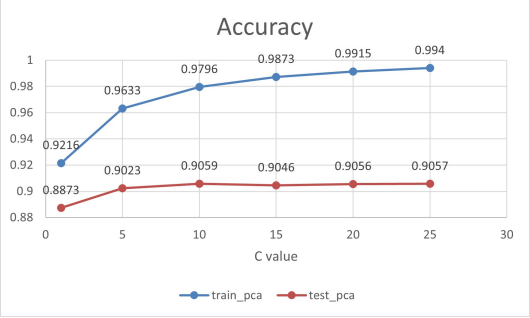


Figure 20: Accuracy(PCA-SVM)

Although train accuracy is a little bit lower when we don't use PCA, test accuracy is slightly higher after PCA processing. There isn't much of a difference between Cs greater than 10, and test accuracy peaks at C = 10. Therefore, our final SVM model should contain C value equals to 10 and use data with PCA.

4.5 Softmax

The traditional logistic regression is only for binary classifications. Softmax function is a generalization of the logistic function to multiple dimensions. It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.

The confusion matrix is below and the test accuracy is 0.8274.

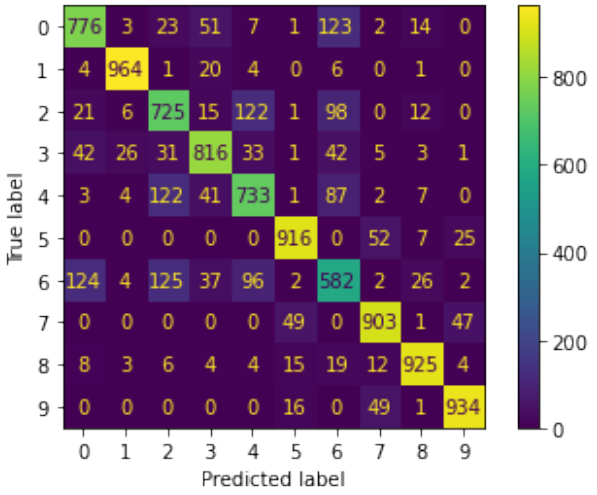


Figure 21: Confusion matrix of Softmax

5 Ensemble Model and Feature Engineering

5.1 Stage 1

The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models. Ensemble learning can be broken down into two tasks: developing a population of base learners from the training data, and then combining them to form the composite predictor.

Based on previous performance, we use four inputs for the first stage, they are: PCA-KNN(k=4), RandomForest(trees=900), XGBoost(objective=softmax), PCA-SVM(default gamma, C=10). We combined these four test predictions as the output for stage 1 and each column can seem like a new variable for stage 2.

Table 3: Example Input For Stage 2

	New x_1	New x_2	New x_3	New x_4
Obs1	9	9	9	9
Obs2	2	2	2	2
Obs3	1	1	1	1
…	…	…	…	…
Obs10000	5	5	5	5

5.2 Stage 2

5.2.1 KNN

We use KNN as the classification model and there is still the need to tune the number of neighbors to get better performance. The total number of test data is 10000, and we have 10000 new observations after fitting four models at stage 1. We'd like to split them up into an 8,000-observation training set and a 2,000-observation testing set. When KNN is employed as the model in stage 2, we get the following result.

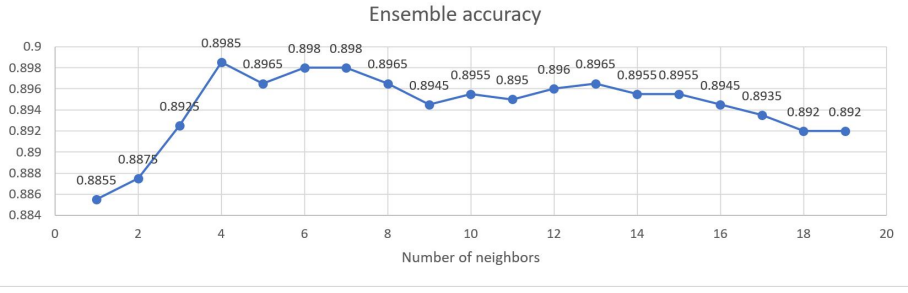


Figure 22: Test accuracy(Ensemble knn)

We can get the conclusion that the best ensemble accuracy appears to have 0.8985 when 4 neighbors were used. However, when it comes to the number of neighbors, their accuracy rates are similar, ranging between 0.89 and 0.90.

5.2.2 Hard Voting

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on the highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models

and finding the accuracy for each of them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

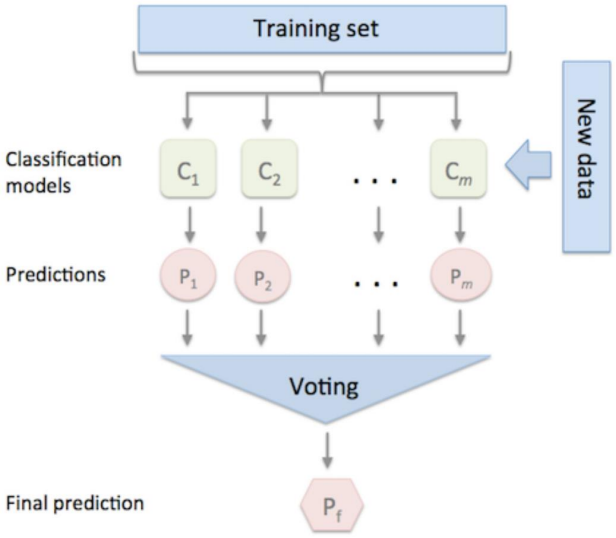


Figure 23: Voting flowchart

Voting Classifier supports two types of voting.

- Hard Voting: Predict the class with the largest sum of votes from models
- Soft Voting: Predict the class with the largest summed probability from models

We will use hard voting here, with a voting ensemble result of 0.9201 for train accuracy and 0.8975 for test accuracy. This result is similar to what we have in the Knn ensemble.

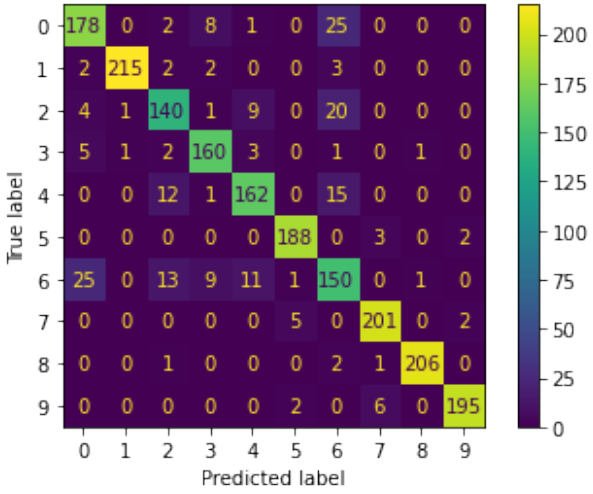


Figure 24: Confusion matrix of Voting

References

[1] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[2] Muhammad Suhaib Tanveer, Muhammad Umar Karim Khan, and Chong-Min Kyung. Fine-tuning darts for image classification. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4789–4796. IEEE, 2021.

[3] Wenqi Hua and Lingfei Mo. Clustering ensemble model based on self-organizing map network. *Computational Intelligence and Neuroscience*, 2020, 2020.

[4] Zhenyu Liao and Romain Couillet. A large dimensional analysis of least squares support vector machines. *IEEE Transactions on Signal Processing*, 67(4):1065–1074, 2019.