COMP5318 MACHINE LEARNING AND DATA MINING —Assignmen 1

Bingtao ZHAO SID:490041116

October 2020

1 Introduction

The goal of this research is to correctly classify a given data set according to its characteristics. This job contains a total of 30,000 training data and 5,000 test data (the first 2000 of the test data already have labels). This assignment needs to complete the code of the classifier, and process the data, and also need to compare three different classification methods to understand their differences and their impact on the final result. At the same time, the classification part of machine learning is included, and the basic knowledge and coding ability are tested, so this research is very important. In order to meet the requirements, I tried PCA to pre-process the data, and the classification algorithm used KNN, Naive Bayesian, and Logistic Regression as the algorithm of the classifier respectively, using the training set for algorithm training, and using 2000 labeled test sets for testing performance.

2 Methods

2.1 Pre-processing

The data set is composed of 28*28 grayscale images, and contains a total of 10 categories:

Class	Class Name
0	T-shirt/Top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

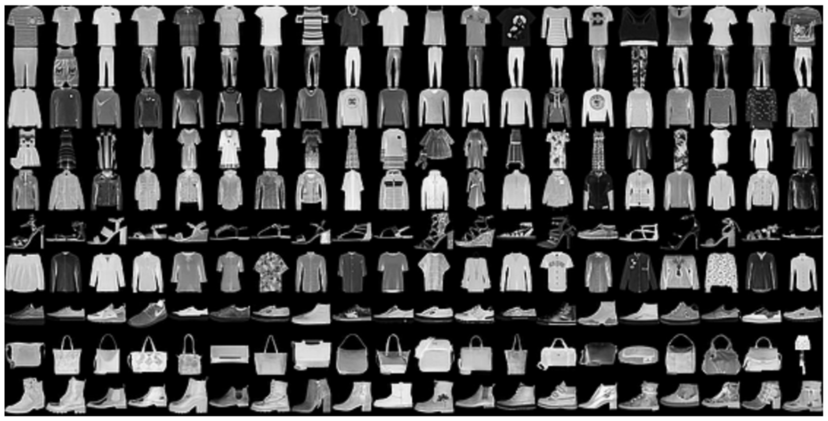


Figure 1: Dataset Sample

2.1.1 PCA

Principle Component Analysis (PCA) is a commonly used data processing method in the field of machine learning. It transforms the original data into a set of linearly independent representations of each dimension through linear transformation, which can be used to extract the main feature components of the data and the dimensionality reduction of high-dimensional data, and after the transformation This set of variables is what we call the principal component. Its purpose is to reduce the dimension of extremely large data to an acceptable amount, thereby reducing the impact on the consumption of computing resources and the complexity of the algorithm model[1]. The processing process is as follows:

(1) Transpose the original data, each row represents a dimension;
(2) Each row (representing an attribute field) is zero-averaged, that is, the average value of this row is subtracted;
(3) Get the covariance matrix of the original data;
(4) Find the eigenvalue of the covariance matrix and the unit vector of the corresponding eigenvector;
(5) Arrange the eigenvectors in rows from top to bottom according to the corresponding eigenvalues into a matrix, and take the first k rows to form a matrix P;
(6) Multiply the matrix P obtained above with the standardized data to obtain the data after dimensionality reduction to k dimensions;

After calculating the covariance of the data, by reducing the data to two dimensions, the data distribution map as shown in the Figure 2.

By choosing to retain 90% of the feature value data, the data dimension of the training set and test set is reduced from 784 to 84 as shown in Figure 3.

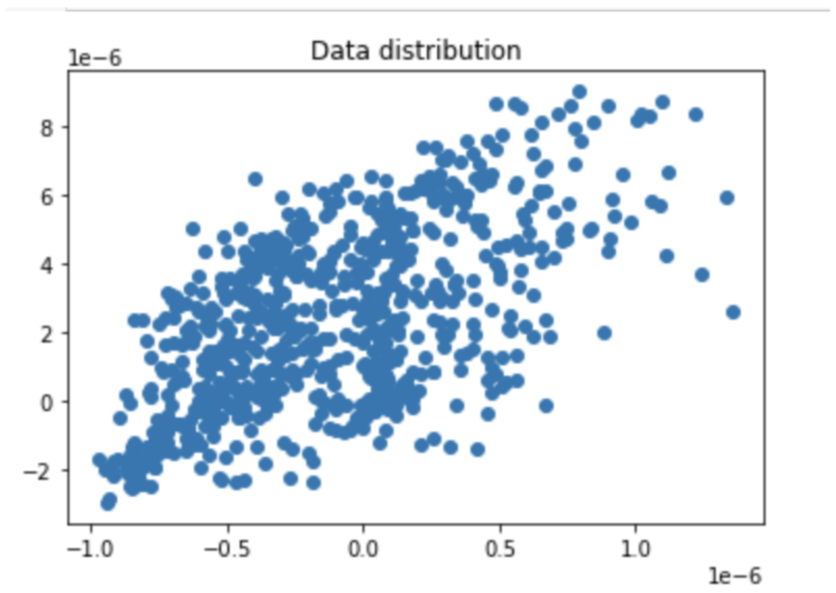
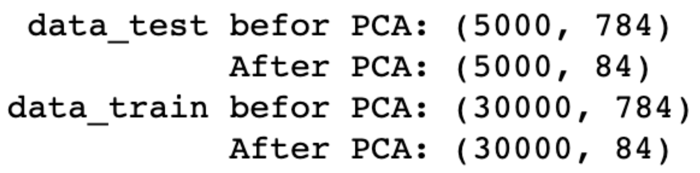


Figure 2: Date Distribution



```
 data_test befor PCA: (5000, 784)
           After PCA: (5000, 84)
data_train befor PCA: (30000, 784)
           After PCA: (30000, 84)
```

Figure 3: Date Demansion After PCA

2.2 KNN

2.2.1 Introduction of KNN

K- Nearest Neighbor(KNN) is a simple algorithm for classification and regression. Given a training data set, for a newly input data instance, by finding the K instances closest to the instance in the training data set, most of these K instances belong to a certain class, then the input instance is classified into this class in. (This is similar to the idea that the minority obey the majority in real life) The value of k here needs to be manually selected. If the value of k is too large, it will result in selecting too large a range of training data for prediction. At this time, it is far from the input instance (not similar) Training examples will have an impact on the results, making predictions wrong, and the increase in the value of k means that the overall model becomes simple. On the contrary, if the value of k is too small, the overall model will become complicated and overfitting will easily occur. Steps of KNN algorithm:

(1) Calculate the distance between the test data and each training data;
(2) Sort according to the stepwise relationship of distance;
(3) Pick the K points with the smallest distance;
(4) Determine the frequency of occurrence of the category of the first K points
(5) Return the category with the highest frequency among the first K points as the predicted category of the test data.

2.2.2 Application of KNN in Assignment

The key value in the knn classifier that determines the effect of the classifier is the value of k. A small value of k is prone to overfitting, which makes the model more complicated. A large value of k will result in a training instance that is far away from the input instance (not similar). It plays a role in the prediction and makes the prediction wrong. The increase of the k value means that the overall model becomes simple.

In order to select the appropriate value of k, through the for loop, K = 1, K=2...K=9 are tested for accuracy respectively, and the results are shown in the Figure 4 below.

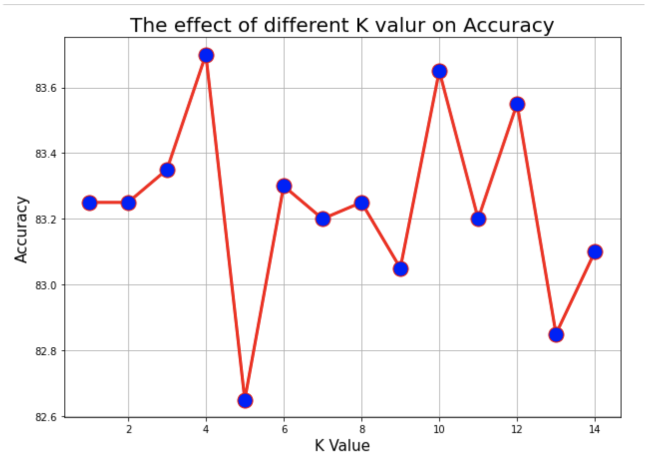


Figure 4: The effect of different K value on Accuracy

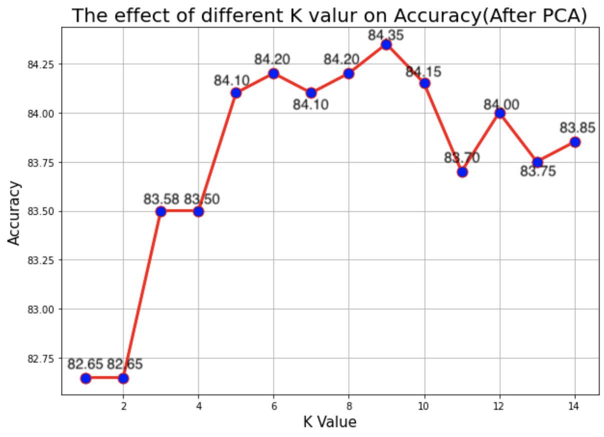


Figure 5: The effect of different K value on Accuracy (After PCA)

2.3 Naive Bayesian

2.3.1 Introduction of Naive Bayesian

Naive Bayes classifier refers to a supervised learning algorithm based on Bayes' theorem, and is a probabilistic classifier. The feature is that each pair of features of the classification are independent of each other. Therefore, computational difficulties are avoided (directly estimate the joint probability based on limited training samples). Three common models of Naive Bayes include Gaussian, Bernoulli and polynomial implementation.

(1) Gaussian Naive Bayes-to deal with continuous variables, the data is required to conform to the Gaussian distribution (a certain ability to deal with the problem of sample imbalance).

(2) Bernoulli Naive Bayes-can only handle binomial distribution data, good at processing data sets with short text, text mining. After the data is normalized, the prediction effect becomes lower, and a threshold must be set, which is not sensitive to sample imbalance.

(3) Polynomial Naive Bayes-dealing with discrete variables. The features it involves are often times, frequencies, and counts. The polynomial naive Bayes in sklearn does not accept negative input and is greatly affected by the data structure.

2.3.2 Application of Naive Bayesian in Assignment

The Naive Bayes algorithm uses a Gauss-based model. And the use of resampling method is to maximize the use of training data, because we cannot access new data during training, so we must use statistical methods to estimate the performance of the model on new data. By using different numbers of data sets to test the model, comparing the time spent and the accuracy rate, it is found that when the number is 10, the overall performance is the best.

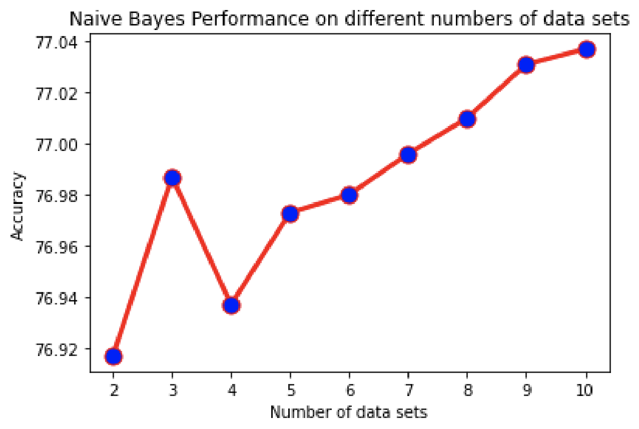


Figure 6: Naive Bayes Performance on different numbers of data sets (1)

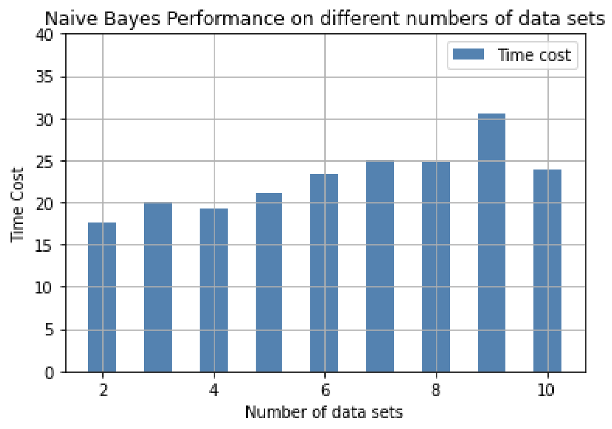


Figure 7: Naive Bayes Performance on different numbers of data sets (2)

2.4 Logistic Regression

2.4.1 Introduction of Logistic Regression

Logistic regression is a generalized linear regression analysis model. It is one of the most common model methods in the machine learning field. It is often used as a baseline for processing various tasks. Although it is a regression model, it is often used for classification problems. Currently, it is often used in data mining, automatic disease diagnosis, economic forecasting and other fields.

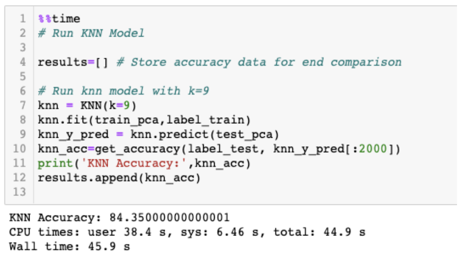
2.4.2 Application of Logistic Regression in Assignment

Input data x to get y, and then by doing sfotmax calculation on this y, the result is turned into a probability vector. Make the difference between the obtained probability vector and the real result to calculate loss, which is log loss. Use loss to find the gradient of the coefficient w of this logistic regression model, assuming that the obtained gradient is $\delta, thenthenewcoefficientwisw(new) = w(old) - gradient\delta.Throughtherepeatedoperationoftheaboveprocess, thepurposeofupdatingthecoefficientisachieved,$

3 Experiments result and discussion

3.1 Result of KNN

After optimization,when k=9, the results of the KNN model are as follows.



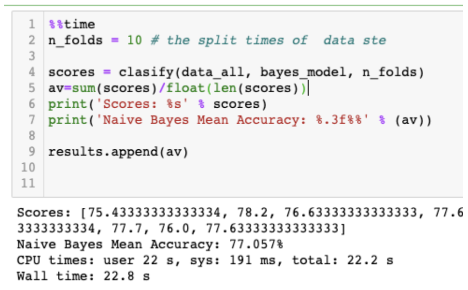
```
%%time
# Run KNN Model

results=[] # Store accuracy data for end comparison

# Run knn model with k=9
knn = KNN(k=9)
knn.fit(train_pca,label_train)
knn_y_pred = knn.predict(test_pca)
knn_acc=get_accuracy(label_test, knn_y_pred[:2000])
print('KNN Accuracy:',knn_acc)
results.append(knn_acc)

```

```
KNN Accuracy: 84.35000000000001
CPU times: user 38.4 s, sys: 6.46 s, total: 44.9 s
Wall time: 45.9 s
```

Figure 8: KNN Model result

3.2 Result of Naive Bayes

After checking the time spent and the accuracy rate to optimize the number of split data sets, it is found that when the number is 10, the overall performance is better.

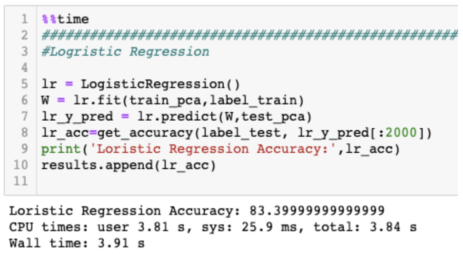


```
%%time
n_folds = 10 # the split times of  data ste

scores = clasify(data_all, bayes_model, n_folds)
av=sum(scores)/float(len(scores))
print('Scores: %s' % scores)
print('Naive Bayes Mean Accuracy: %.3f%%' % (av))

results.append(av)
```

```
Scores: [75.43333333333334, 78.2, 76.63333333333333, 77.6
3333333334, 77.7, 76.0, 77.63333333333333]
Naive Bayes Mean Accuracy: 77.057%
CPU times: user 22 s, sys: 191 ms, total: 22.2 s
Wall time: 22.8 s
```

Figure 9: Naive Bayes Results

3.3 Result of Logistic Regression



```
%%time
##################################################
#Logistic Regression

lr = LogisticRegression()
W = lr.fit(train_pca,label_train)
lr_y_pred = lr.predict(W,test_pca)
lr_acc=get_accuracy(label_test, lr_y_pred[:2000])
print('Loristic Regression Accuracy:',lr_acc)
results.append(lr_acc)
```

```
Loristic Regression Accuracy: 83.39999999999999
CPU times: user 3.81 s, sys: 25.9 ms, total: 3.84 s
Wall time: 3.91 s
```

Figure 10: Logistic Regression Results

3.4 Comparison between models

Among the three classifiers, KNN consumes the most time, while logistic regression consumes the least time.

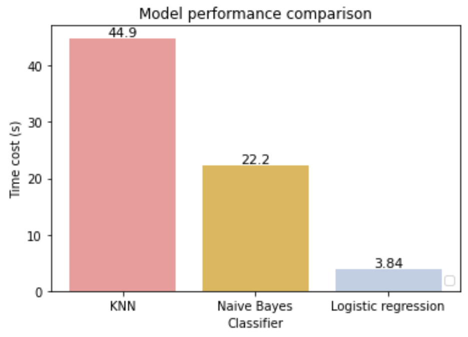


Figure 11: Three Models Time Spent

It can be seen from the figure that KNN has the highest accuracy rate, reaching 84.35%, logistic regression is 83.4%, and Naive Bayes has the lowest accuracy rate of 76.98%.

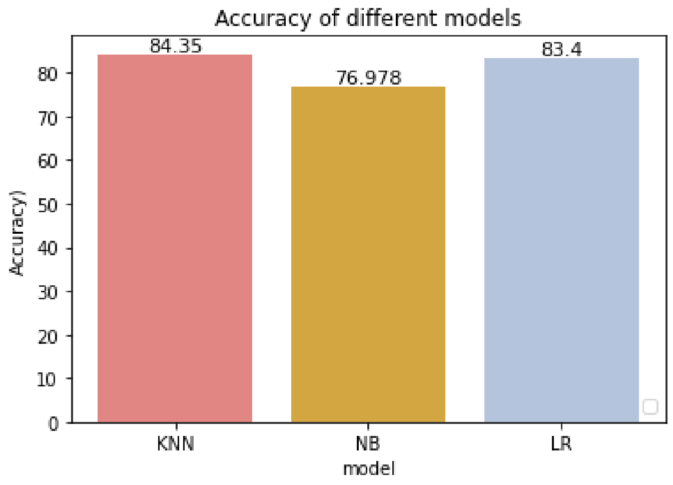


Figure 12: Accuracy of different models

Comparing the accuracy with the sklearn classifier, the result is similar to that shown in the figure

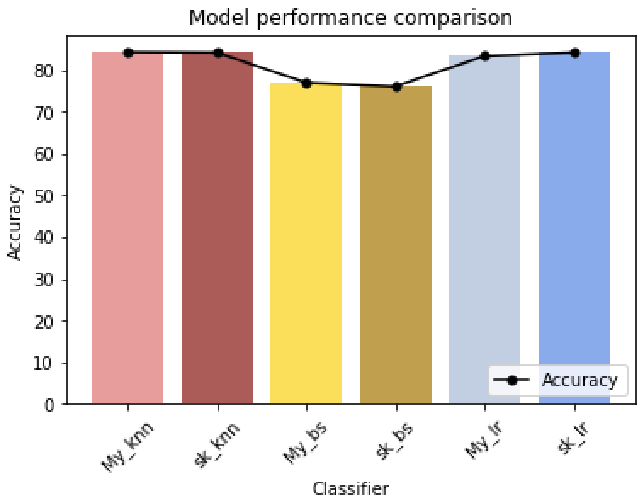


Figure 13: Compared with Sklearn's Results

Compared with the running time of sklearn's classifier, sklearn's classifier runs faster than three classifiers.

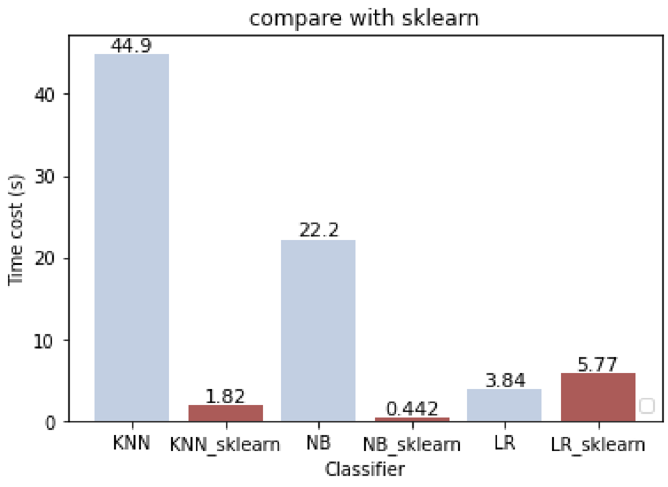


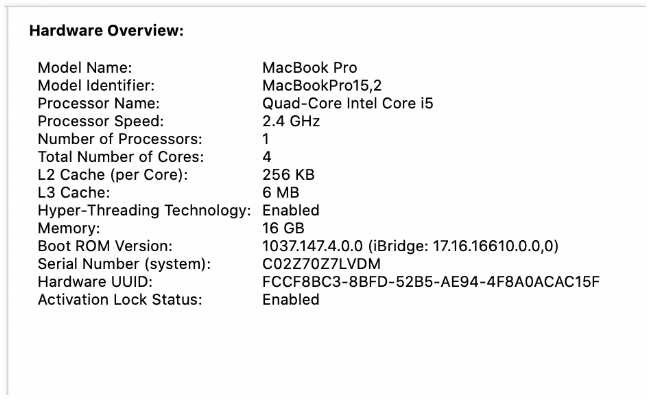
Figure 14: Time spent compared to sklearn

4 Conclusion

The characteristics of the three classifiers are different. According to Figure 11, it can be seen that the KNN time cost is the most, and the logistic regression classifier has the least time and the highest accuracy rate among the three (Figure 12). Naive Bayes has no outstanding performance in terms of running time and accuracy. In addition, directly using sklearn to implement the three classifiers has a clear advantage in time consumption (Figure 14), but the accuracy rate is not much different, indicating that the model accuracy rate is not inaccurate.

5 Hardware Overview

The computer hardware configuration is shown in the figure below.



```
Hardware Overview:

  Model Name:                  MacBook Pro
  Model Identifier:            MacBookPro15,2
  Processor Name:              Quad-Core Intel Core i5
  Processor Speed:             2.4 GHz
  Number of Processors:        1
  Total Number of Cores:       4
  L2 Cache (per Core):         256 KB
  L3 Cache:                    6 MB
  Hyper-Threading Technology:  Enabled
  Memory:                      16 GB
  Boot ROM Version:            1037.147.4.0.0 (iBridge: 17.16.16610.0.0,0)
  Serial Number (system):      C02Z70Z7LVDM
  Hardware UUID:               FCCF8BC3-8BFD-52B5-AE94-4F8A0ACAC15F
  Activation Lock Status:      Enabled
```

Figure 15: Hardware Overview

References

[1] Jolliffe, I. T. Principal Component Analysis. Springer-Verlag. 1986: 487 [2012-01-24]. ISBN 978-0-387-95442-4. doi:10.1007/b98835.