COMP5318 Assignment 1

SID: 510138903

Name: Yichen Chen

Uni: Yche3494

1. Introduction

In the past few decades, science and technology have changed with each passing day. Take the mobile phones which we use them every day for example, iPhone can use 3D structured light technology to support unlock and recognize the face now. And if you turn on the AI camera, the mobile phone can use various image recognition and classification algorithms to change the parameters of camera, to take more expressive photo.

How can we step into the field of these exciting technologies? I think this study can be the first step. As a data science student, I think the purpose of this study is to initially build our understanding of machine learning, and gradually accumulate the following practical experience:

Through the classification of 28x28 pixel images, we can accumulate experience in data preprocessing methods such as PCA and minmaxscale, and intuitively feel the disaster of dimension curse and the convenience brought by dimensionality reduction technology via the running time of the algorithm; we can exercise our python code ability by implementing different classifier algorithms such as KNN, SVM, decision tree, etc., import libraries and learn their parameters, cultivate our sensitivity to different classifiers, and generate classifier algorithms Practical understanding; via quantitatively analyzing the accuracy performance of different parameters under the k-fold training set, we can accumulate practical experience of tuning parameters; comparing the accuracy performance of different models on this project data set, learn about different scenarios(data) suitable for different models

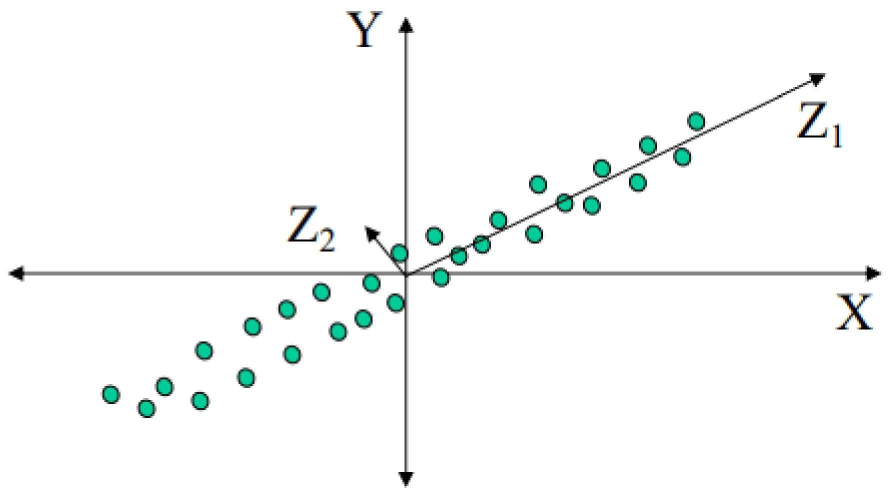
I think the importance and purpose of this study is to guide us to understand these technologies in principle, through hands-on practice to generate a deeper understanding, so that we can use them in future studies and work, and even improve them one day.

2. Methods

2.1 Preprocessing：

PCA:

Principal component analysis, PCA is a common un-supervised learning method. It is used for dimensionality reduction of high-dimensional data. It reconstructs the axis of the data in a way that maximizes the variance.
For example, we want to reduce the dimensionality of a set of two-dimensional data into one-dimensional data:



[1]That is, we plan to project some points on a two-dimensional surface onto a one-dimensional line, we will hope that the projection are as scattered as possible. Intuitively, if they are very close, even overlap, there will become a same data, which decrease the entropy of whole data.
Here, the best way is to choose Z1 axis as PCA1, because along Z1 axis come out the largest variance. In the process of PCA selecting features, the coordinate axes are all orthogonal, thus, if we continue find Z2, Z2 will be perpendicular to Z1.
In high-dimensional space, we use covariance to measure the degree of dispersion of data.

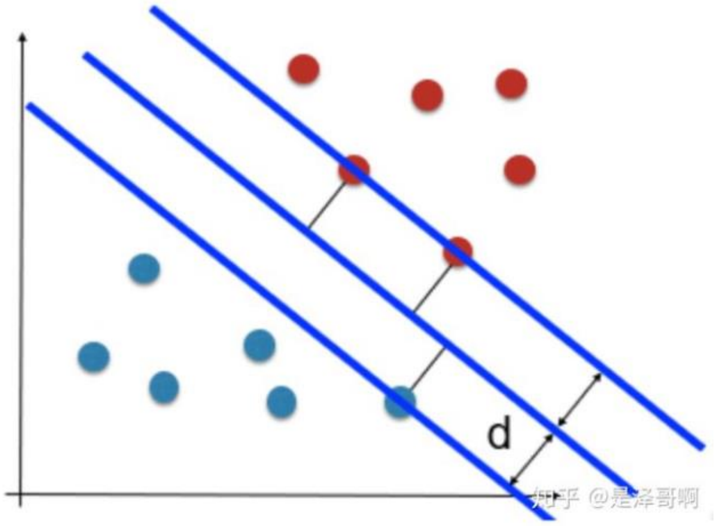
$$Cov(a,b) = \frac{1}{m-1}\sum_{i=1}^{m}(a_i-\mu_a)(b_i-\mu_b)$$

PCA can:
Avoid Curse of Dimensionality; noise reduction; avoid overfitting; make features become independent of each other. [2]

Classifier methods:

2.2 SVM:

[3]Support Vector Machine is a supervised learning non-parametric method. Its decision boundary is maximum-margin hyperplane. The support vectors are points in the sample that are closest to this hyperplane.



Use this equation to describe a hyperplane:

$$w^T x + b = 0$$

We can get distance from a point to hyperplane:

$$\frac{|w^T x + b|}{||w||}$$

where:

$$||w|| = \sqrt{w_1^2 + \dots w_n^2}$$

We know that support vector are the points whose distance to hyperplane equal to d, and the other point greater than d. Thus:

$$\begin{cases} \dfrac{w^T x + b}{||w||} \geq d \quad y = 1 \\ \dfrac{w^T x + b}{||w||} \leq -d \quad y = -1 \end{cases}$$

For the convenience of calculation and derivation set ||ω||·d = 1 :

$$y(w^T x + b) \geq 1$$

2.3 Decision Trees:

[4]Decision Trees is a supervised learning non-parametric method.
We choose a new attribute which has highest information gain as next determine attribute, where: Information gain (Set 1 to Set 2) = T1 – T2, Ti is entropy.

Entropy H(S) of data set S:

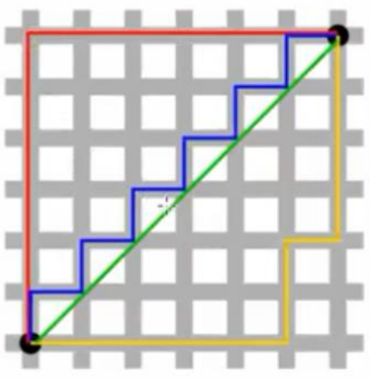
$$H(S) = I(S) = -\sum_i P_i . \log_2 P_i$$

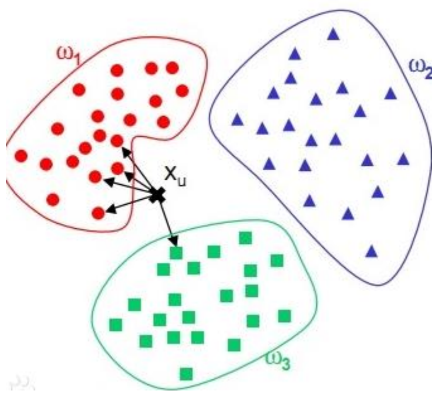
P_i - proportion of examples that belong to class *i*

2.4 KNN:

[5]Nearest neighbor is an supervised non-parametric method.

1. Normalize the data. Like Minmaxscale.
2. Calculate the distance between data you want to classify and sample training data. This distance can be Euclidean distance and Manhattan distance and so on. This is one of the hyper-parameters
3. Find out the nearest k samples. Where k is the number of the neighbor, which is one of the hyper-parameters.
4. Count these k samples' labels, and take the label with the most occurrences as the label of the data to be classified.





Where green line shows Euclidean distance, blue, red, yellow lines show Manhattan distance.

2.5 Naïve Bayes:

[6]Naïve Bayes is a supervised learning non-parametric method which based on the Bayes Theorem.

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Assume two assumptions:

1. Between different features no mutual influence, they are irrelative, independent.
2. The weight of each features are equal.

And for numeric attributes, we apply equation below instead of P :

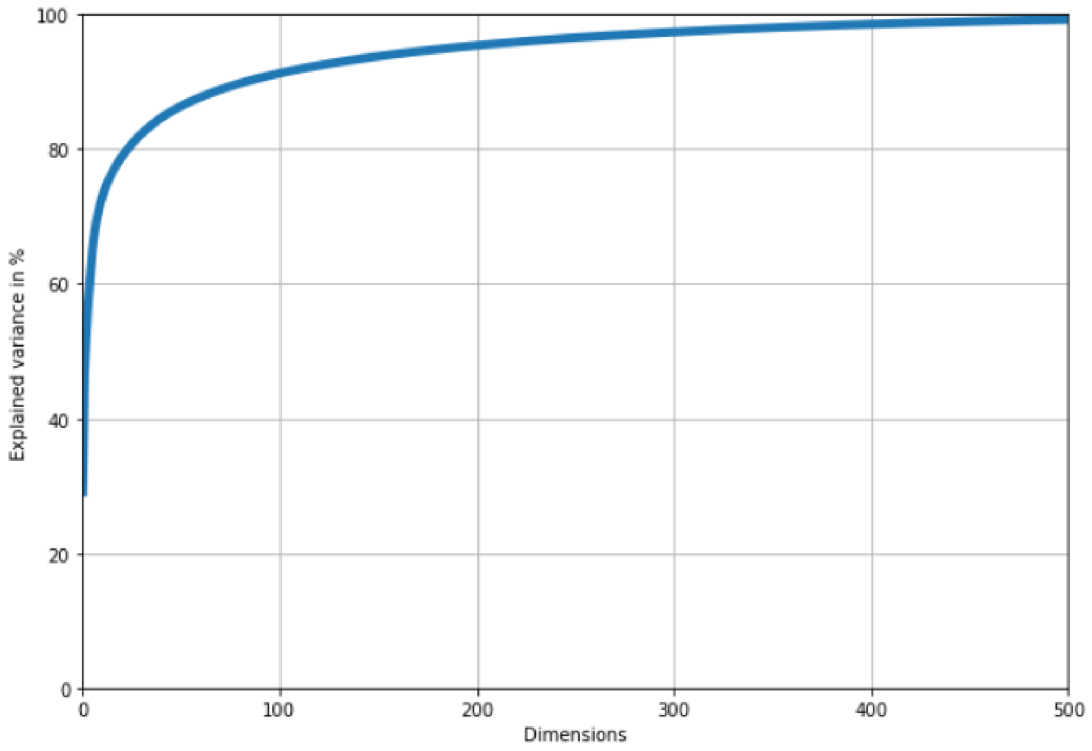
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where f(x) is the probability density of x obeys normal distribution n(mu, sigma).

3 Experiments and Results

3.1 Preprocessing：

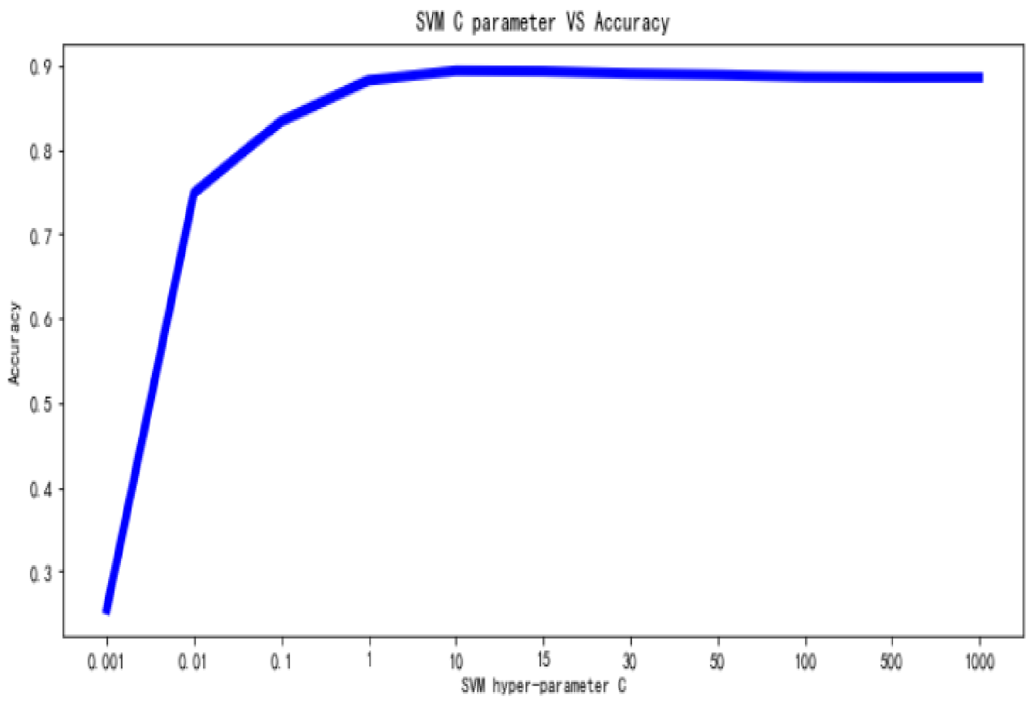
In order to choose appropriate dimensions of PCA, I use np.cumsum to calculate Explained variance by specific Dimension. And the result is showing below:

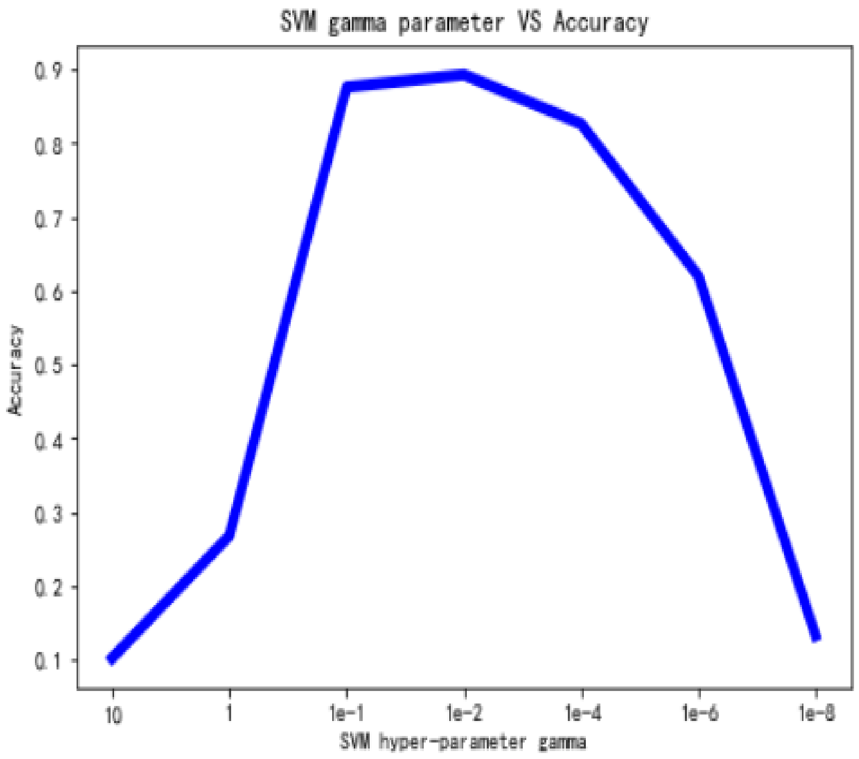


Finally, I chose n = 150 components which can retain close to 95% variance.

3.2 Tuning hyper-parameters:

I will use two line charts of SVM C and gamma parameters to slightly show part of process of tuning hyper-parameters.





As we can see from the chart, the line keeps rising from C = 0.001 and reaches the turning point at about C = 12 with 90% accuracy, and then began to maintain a very small rate of descent. Keeping the other hyperparameters the same, we choose C=12 as best hyper-parameter. Keep C = 12, tuning gamma. The line chart shows that around 1x10^-2 has best performance. Actually I use the combination of gamma and C and via Grid-search process to get the best hyper-parameter combination of gamma and C, which is not exactly same to the charts above. They here just for showing the process.
The combination , for example, (C = 5, gamma = 0.0015), (C = 5, gamma = 0.15), (C = 12, gamma = 0.0015)……(C = 20, gamma = 0.15).

The best combination of parameters for all the models I chose is listed below:

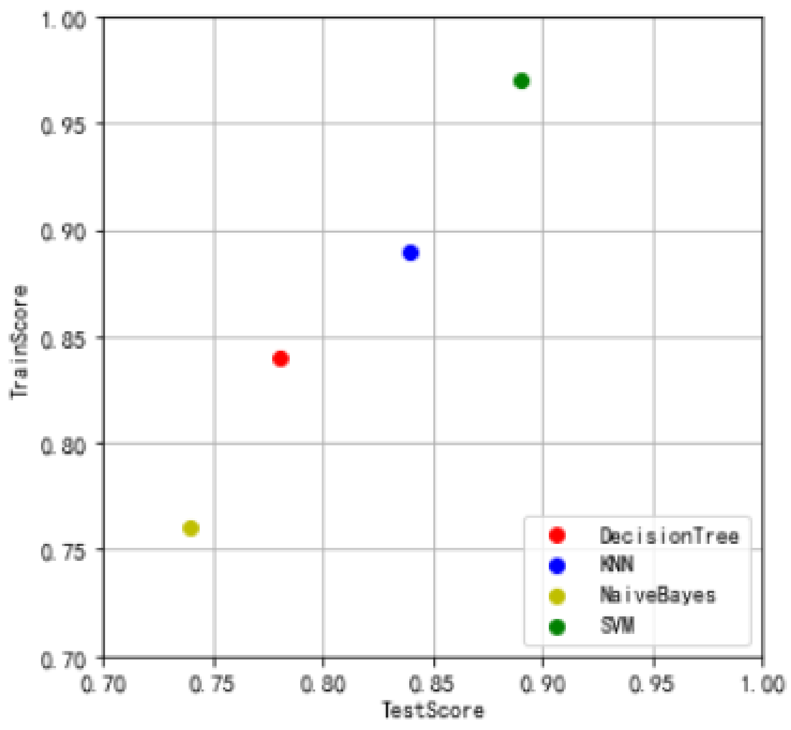
Decision Tree: `(criterion='entropy', max_depth=10)`
KNN: `(n_neighbors=7)`
Naïve Bayes: `(var_smoothing=0.0)`
SVM: `(kernel = 'rbf',C=12, gamma=0.015)`

3.3 Training result:

We can see experiment results between different algorithms from the table below.

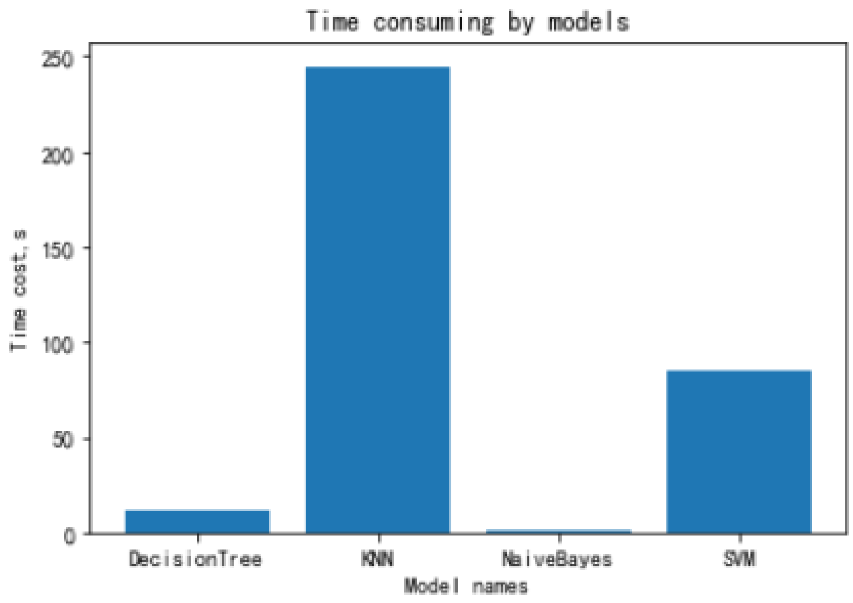
Model	DecisionTree	KNN	NaiveBayes	Svm
Training accuracy	0.84	0.89	0.76	0.97
Test accuracy	0.77	0.84	0.74	0.89
Time consuming, s	11.9	254.5	0.5	82.8

Make a plot for the result.



We can see from the plot that SVM have both highest Training score 0.97 and Test score 0.89.

Then make a histogram for time of running models.

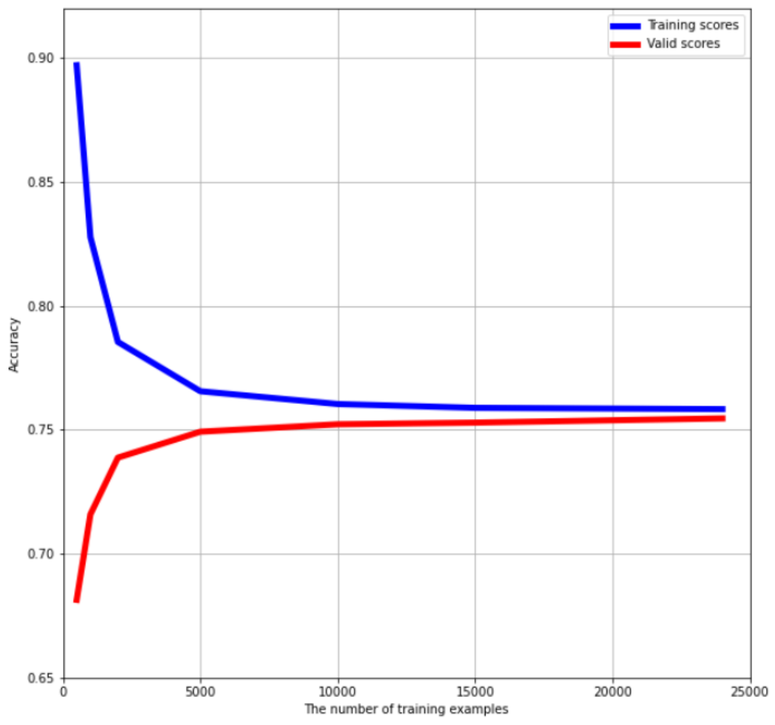


We can see from the histogram that KNN consumed largest time about 4 minutes, while Naïve Bayes is the lowest one at time consuming.

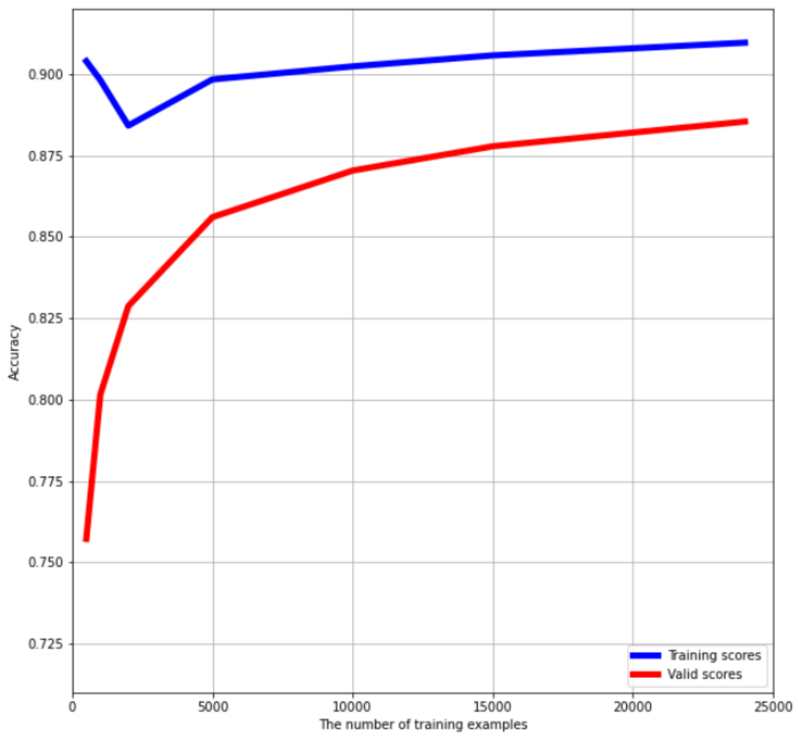
3.4 Explore more:

[7]Obviously SVM has the highest accuracy of training and test data set. I tried to explore more information. I checked sklearn's documentation about how to evaluate the model, in addition to accuracy, we can also use learning curve, which can explore and compare the impact of data set size on accuracy , I do here for SVM and NaiveBayes, using package “learning_curve” from sklearn. (Details see coding part 6.1)
“A learning curve shows the validation and training score of an estimator for varying numbers of training samples. It is a tool to find out how much we benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error.“[7]

And results are shown below via line chart:



For naïve bayes, both the validation score and the training score converge to a small value while we adding samples(validation data). That means we can not get any benefit from increasing samples any more. The upper limit of the Naïve bayes model is low, that means the model is too simple



If the training score is much larger than the validation score when the training sample is the largest, then increasing the training sample is likely to increase the generalization ability.
We can see that even at the maximum of the sample number, the training score of SVM is

still much higher than that of validation, and it has not fully converged, there is a continuing upward trend. This shows that SVM can continue to benefit from increasing the sample size. The upper limit of SVM is very high. This often indicates that this is a complex model. Obviously, increasing the sample size is a profitable business, which can reduce the risk of under-fitting without increasing the risk of over-fitting.

In general, in terms of time consumption, we can see that KNN is the most complicated model, followed by SVM, while NB and DT having short time consumption may mean they are simple models. This may be the one of reasons why accuracy of NaiveBayes and DecisionTree are lower.

However, “If the training score and the validation score are both low, the estimator will be underfitting. If the training score is high and the validation score is low, the estimator is overfitting and otherwise it is working very well. A low training score and a high validation score is usually not possible.”[7]

SVM can still increase the upper limit of the model by increasing the data, the reason already has been discussed above. But consider the sharp difference between training score and test score(0.97 and 0.89), I can conclude that SVM is obviously over-fitting.
The upper limit of the NaiveBayes model is low, and the adjustable hyper-parameters have little effect. The test score is close to the training score. It may because that the model is too simple. Or maybe there are still a lot of variables having correlation between each other after PCA, which does not meet the assumption of NaiveBayes: the variables are not correlated.
Knn, DecisionTree also have the possibility of overfitting having high accuracy of training much greater than test score.

4 Report – Conclusion

Totally I think three of the four models are at risk of over-fitting, one of the reason causing these may be k number of k-fold. The larger the k, the smaller the bias and the larger the variance. The suggestion for future work is the k-fold should be reduced to 5 or 3 to reduce the risk of over-fitting to increase robustness.
Another reason causing over-fitting may be PCA. Perhaps PCA retains too many dimensions, which brings too much noise and correlation between features. It may be possible to establish the accuracy of the four models under different PCA situations to measure this situation.
In addition, for future work, it can be seen from the learning curve chart of SVM and Naivebayes that adding more data is a general way to improve accuracy. At present, SVM still has a lot of space for improvement. And it is the best performing model at this stage, Its model is not so complex that takes a lot of time to run, it’s lower than KNN’s consuming actually.

5 Appendix

5.1 How to run code:

If you want to run all the processes (including tuning), you need to shift+enter block by block.

If you only want to run specific block, make sure you have run block 1.1 preparations for importing packages. And run the other necessary blocks to define functions.

Block 1.2 is a function for timer, which is necessary to run for your evaluation of model's time consuming, pre-processing and so on.

Block 1.3 is for loading data, block 5 is for output data.

Grid search CV's function has been defined in block 1.4 PCA, run it if necessary.

If you do not want to run tuning, just ignore block 2.1~2.4.

Block 3.1~3.4 is for evaluating training and test scores and time consuming.

In addition, I have already used the output square under the function body to indicate the time consumption details of some key parts. All the places that need printing accuracy have also been printed.

5.2 References:

[1] Nguyen Hoang Tran. Dimensionality Reduction Reference: Müller and Guido ch. 3.4.1: 142-157, Geron ch.8: 219-226, Witten ch.8.3: 305-307 COMP5318 Machine Learning and Data Mining semester 2, 2021, week 6b Nguyen Hoang Tran Based on slides prepared by Irena Koprinska (irena.koprinska@sydney.edu.au)

[2] [Ze A, [Machine learning] Dimensionality Reduction - PCA【机器学习】降维——PCA https://zhuanlan.zhihu.com/p/77151308]

[3] [Ze A, [Machine learning] 支持向量机 SVM 【机器学习】降维——PCA https://zhuanlan.zhihu.com/p/77151308]

[4] Nguyen Hoang Tran. Decision Trees Reference: Witten ch.4.3 and ch.6.1, Tan ch.3.3 COMP5318 Machine Learning and Data Mining semester 2, 2021, week 5a Nguyen Hoang Tran Based on slides prepared by Irena Koprinska (irena.koprinska@sydney.edu.au)

[5]https://baike.baidu.com/item/%E6%9B%BC%E5%93%88%E9%A1%BF%E8%B7%9D%

E7%A6%BB/743092?fr=aladdin

[6] Nguyen Hoang Tran. Naïve Bayes. Evaluating Machine Learning Methods. 1 Reference: 1) Witten ch.4.2, Tan ch.5.3 2) Witten ch.5: 128-131, Tan: ch.4.5, Müller & Guido: ch.5 COMP5318 Machine Learning and Data Mining semester 2, 2021, week 4 Nguyen Hoang Tran Based on slides prepared by Irena Koprinska (irena.koprinska@sydney.edu.au)

[7] https://scikit-learn.org/stable/modules/learning_curve.html