Report for Practical Assignment

Yuhang Tian (5219728)

Emiel van der Meijs (4567528)

0. Preparation

0.1.1 (4 points) For each of the five algorithms list key strength and key weakness. Use no more than 250 words in total (+- 50 per algorithm).

- GaussianNB:

✓ It can perform much better compared to the other four classifiers when the samples are independent predictors.

✓ The training speed is faster even with a large dataset.

✓ It requires a small amount of training data.

× It highly relies on the assumption of independent predictors.

× It has the Zero Frequency problem.

- DecisionTreeClassifier:

✓ It requires less pre-processing work.

✓ It can deal with the null values.

✓ It is easy to interpret after it has been built.

× Insignificant variations in the data lead to global changes in the tree structure.

× Training and optimization are expensive.

× The ability to predict continuous values is worse.

- KNeighborsClassifier:

- ✓ It is instance-based learning, which can be faster.
- ✓ New data can be added seamlessly.
- × It performs worse in large datasets.
- × It performs worse with higher dimensions.
- × It is sensitive to varying scales, noises, missing values and outliers.

- SVC:

- ✓ It performs well if there is a clear margin of separation.
- ✓ It is more effective in higher dimensions.
- ✓ The kernel function is strong to solve a complex problem.
- × It is difficult to be interpreted after being built.
- × The tuning process is tough.

- LogisticRegression:

- ✓ It is highly interpretable
- ✓ It requires less preprocessing work
- ✓ It performs well on a linearly separable dataset
- × Linear boundaries cannot solve non-linear problems.
- × It may be over-fit on high dimensional datasets.

0.1.2 (3 points) Carefully read the Scikit-learn hyper-parameter documentation for each of the five algorithms. Based on this documentation explain how the previously mentioned hyper-parameters affect the algorithms and their performance. Express yourself clearly and provide your reasoning. Use no more than 300 words in total (+- 75 per algorithm). Note: You don't have to write anything about the Naive Bayes since it has not hyperparameters of interest.

- DecisionTreeClassifier:

max_depth and min_samples_leaf are the hyper-parameters relevant to tree pruning. In terms of max_depth, its default value is none which means it will expand the tree until all leaves are pure. Turning to min_samples_leaf, the minimum number of samples

required to be at a leaf node, its default value is 1. For a small dataset or a dataset with the relatively small number of features, people can ignore to set these values. Otherwise, people can restrict tree structure to avoid overfitting.

- KNeighborsClassifier:

n_neighbors is the number of neighbors used for voting, and its default value is 5. When deciding the class of a sample, it will look k nearest samples around it, and assign the sample to the class which contains most samples in this k-sample group. weights decides the weights of voting. 'uniform' represents all the k samples voting are equal weights; whereas 'distance' represents the nearer samples have prior rights.

- SVC:

C is the regularization parameter, and the strength of the regularization is inversely proportional to C. C is used for avoiding overfitting problem. kernel decides the type of the kernel function. Since it has various kernels, it can process both linear and non-linear problems well through changing the kernel. 'rbf' kernel is widely used which is the Gaussian kernel, mapping the lower dimensions to the higher ones.

- LogisticRegression:

penalty is used to specify the norm in the penalization. 'l2' penalty has four optimization algorithms, 'newton-cg', 'lbfgs', 'liblinear', 'sag'; whereas 'l1' penalty has only one, 'liblinear'. C is the regularization parameter, and the strength of the regularization is inversely proportional to C. C is used for avoiding overfitting problem.

random_state is a pseudo-random generated number. If people want to have the same shuffle each time, it should be fixed.

1. American Census

1.1.1 (1 point) Explore the features and target variables of the dataset. What is the right performance metric to use for this dataset? Clearly explain which performance metric you choose and why. Use no more than 125 words.

We prefer to choose F1-score as our performance metric for this dataset. There are mainly three reasons for this choice:

- This is a real-life binary classification problem, for which the F1-score is widely used.
- When we explore the label set, we find that the set is imbalanced. There are approximately 4K people have the salaries above the threshold, but about 12K people are below that threshold. F1-score is a better metric to evaluate the imbalanced class distribution which has a large number of actual negatives because the return value of F1-score is a harmonic mean of precision and recall.
- We are familiar with this type of metric.

1.1.2 (1 point) Algorithmic bias can be a real problem in Machine Learning. So based on this, should we use the Race and the Sex features in our machine learning algorithm? Clearly explain what you believe, also provide us with arguments why. Note this question will be graded based only on your argumentation. Use no more than 75 words.

It depends on the target or the goal of our model. If we advocate using these features to dig out and to report the problem relative to racial and gender discriminations, or even age and academic discriminations, these features should be taken into consideration, since we want to use these features to find out the biases, to report them, and even to reduce and eliminate them, in order to build an impartial and harmonious society.

1.2.1 (2 points) This dataset hasn't been cleaned, yet. Do this by finding all the missing values and handling them. How did you handle these missing values? Clearly explain which values were missing and how you handled them. Use no more than 100 words.

There are 1573 samples, which contain at least one missing feature values, in the training set. We used two different ways - dropping the incomplete samples or filling the missing values - to preprocess the data. Dropping them is the simplest method, but we will lose about 10% of the precious training samples; whereas, filling them in manually will add noise. For the filling method, we will create a new group " Unknown" for the missing categorical features and use mean value to fill in the missing numeric features.

1.2.2 (2 points) All Scikit-learn's implementations of these algorithms expect numerical features. Check for all features if they are in numerical format. If not, transform these features into numerical ones. Clearly explain which features you transformed, how you transformed them and why these transformations. Use no more than 75 words. (You might want to read the preprocessing documentation of Scikit-learn for handy tips.)

There are some categorical features which are "workclass", "education", "marital-status", "occupation", relationship", "race", "sex" and "native-country". We used OrdinalEncoder to encode these features. OrdinalEncoder can change the categorical features into numeric integer formats, which will become acceptable to the algorithm implementations.

1.2.3 (Bonus 2 point) Have you done any other data preprocessing steps? If you did, explain what you did and why you did it. Use no more than 100 words.

We have done two more pre-processing steps.

1. In the feature of education, we find that the division is too specific for primary educations compared to other periods of education. Therefore, we merge some of the primary educational periods according to the US educational system, since over-specialized data may lead to overfitting.

2. After encoding the features, the scopes of them vary with each other. Therefore, we created two re-scaled versions for the data to eliminate the varying scales. The first one is a normalized version, and another one is a standardized version.

1.3.1 (1 point) Now set up your experiment. Clearly explain how you divided the data and how you ensured that your measurements are valid. Use no more than 100 words.

We decided to use KFold cross-validation to train and test our models because it can make full use of the dataset, and avoid the overfitting problem to some extent. The value of **n_splits** is set to 10, which means it will use 1 fold for validating, the other 9 folds for training in one round. In terms of ensuring our measurements, we plotted the learning curve for each training process in order to compare the F1-scores.

1.3.2 (2 points) Fit the five algorithms using the default hyper-parameters from section 2.1. Create a useful plot that shows the performances of the algorithms. Clearly explain what this plot tells us about the performances of the algorithms. Also, clearly explain why you think some algorithms perform better than others. Use no more than 150 words and two plots (but 1 is sufficient).

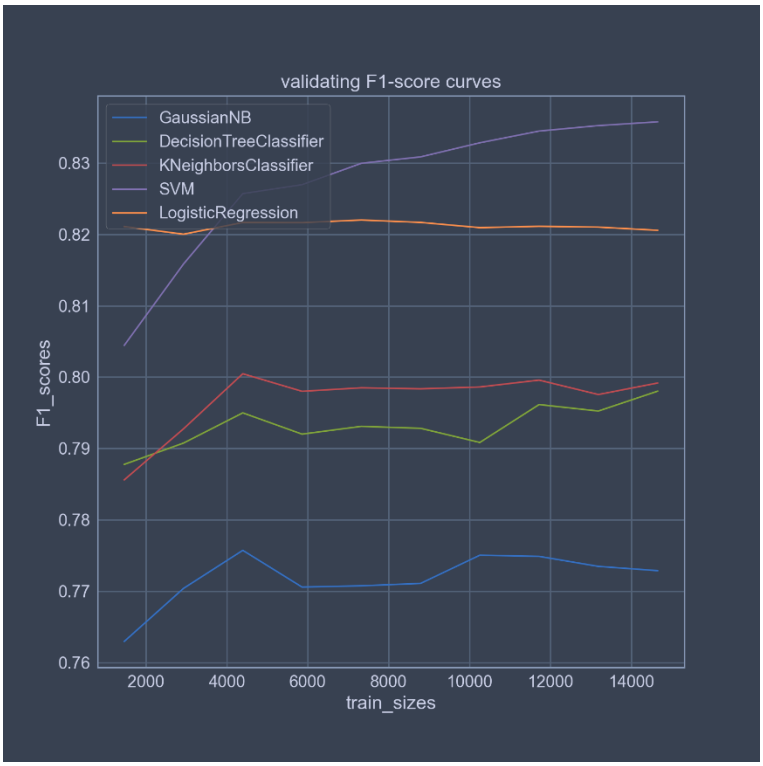

Figure 1 Testing Learning Curve

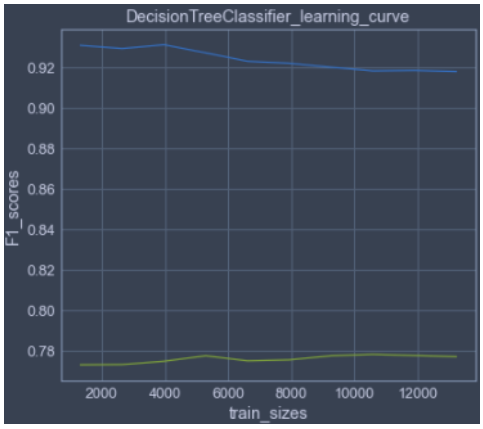

Figure 2 An example of Decision Tree Classifier

In figure 1, it shows the learning curve of testing curves. Using the default setting, SVM is the currently best algorithm among others, since it always has the highest F1-score among the others no matter how large the training size is.

The performances of the algorithms strongly rely on the training samples and their initial hyper-parameters setting. If the SVM performs well on this dataset, this dataset probably has a relatively clear margin of classification, which means there are a few overlapping samples of two different classes.

The fourth one is Decision Tree, but it should be noted that it's training curve and testing curve do not have a convergent trend with the increase of training size, as shown in

figure 2. Therefore, it's performance may be promoted after hyperparameter tuning. The last one is Gaussian classifier, which can be adversely influenced by the real distribution of the samples.

1.3.3 (2 points) Now perform hyper-parameter tuning on the key hyper-parameters you have previously identified. Clearly explain what you did to be systematic, what you did to get fair results, what trade-off accuracy vs resources trade-off, etc. Use no more than 200 words. Note: First focus on tuning the default hyper-parameters, this should be sufficient. Only look at others if time permits it.

Systematic: In order to exhaustively search the best parameters, we adapt GridSearchCV. Using this method, we can define a group of values for each parameter of a model, which will be tested independently each time, and GridSearchCV will return the best combination of values for each algorithm.

Fairness: We cannot access the best values before training and validating. Therefore, we need to perform trial-and-error. Initially, we will set a rough range of values to test their performances. Then, we will narrow down the ranges of the parameters according to the feedback of GridSearchCV. We also set multiple random states to get the best result for each model, because the random state can influence accuracy, even though the effect is not significant.

Trade-off: The first limitation is the number of samples. Therefore, we will feed the model with all samples every round. The second limitation is the wide range of parameters, which can cause the problem of long training times if the ranges for each parameter are set too wide. The third one is the trade-off between accuracy and overfitting. Therefore, we need to stop narrowing down when the accuracy is not significantly promoted after tuning.

1.3.4 (2 points) Compare the performance of the algorithms with and without hyper-parameter tuning. How did the tuning affect your result? Clearly explain the results

and the differences. Use no more than 100 words and two plots (but 1 is sufficient).

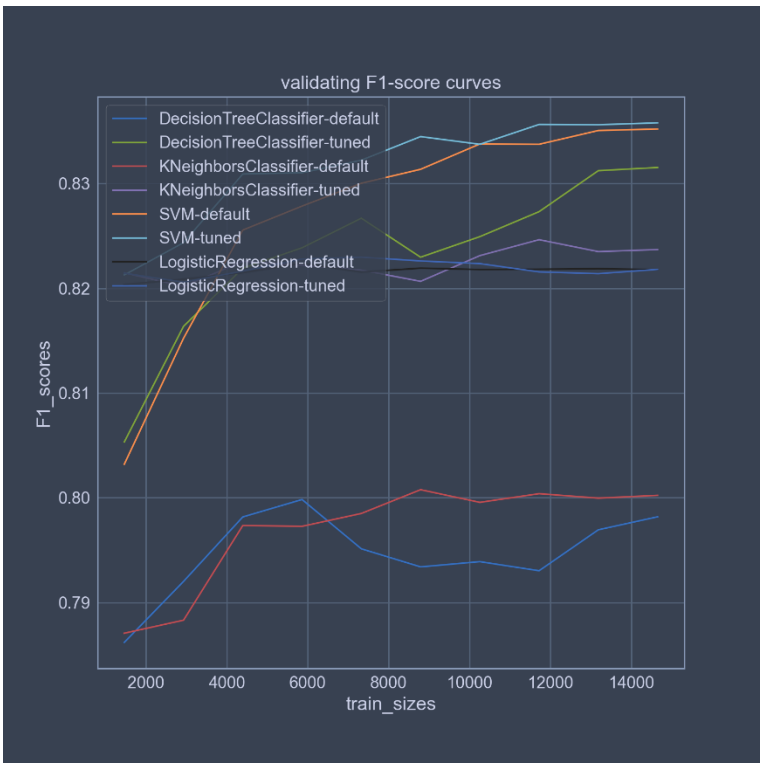

Figure 3 Hyper-parameter Tuning

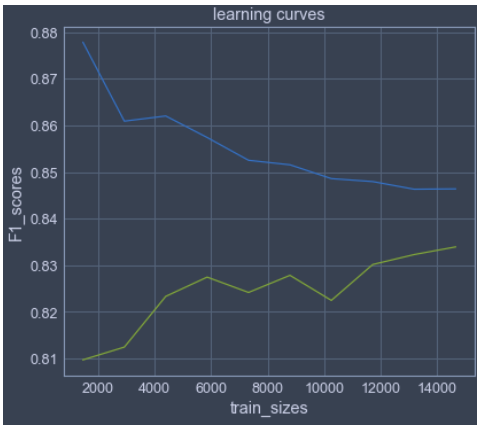

Figure 4 An example of Decision Tree Classifier

In figure 3, After hyper-parameter tuning, all the accuracies of the algorithms have been improved significantly, apart from SVM. SVM still performs well, however, as it still takes the first position F1-score. DecisionTreeClassifier, which was the worst algorithm (except Gaussian), now becomes the second best. Following that are K-Neighbors and LogisticRegressionClassifiers. We do not provide the Gaussian here, since it is not required to adjust hyper-parameters. Back to DecisionTreeClassifier, it now has the trend of convergence as the training size increases, as shown in figure 4. Hence, hyper-

parameters tuning can promote the learning ability of a model.

1.3.5 Select your best algorithm for this dataset and use it to make your predictions for the unknown samples. Please note in your algorithm which algorithm you chose.

We decide to choose Support Vector Machine with the following hyper-parameters:

- C=5,
- kernel="rbf",
- random_state=30

2. MNIST

2.1.1 (1 point) Explore the dataset by plotting the same image from both datasets side by side. How do these images compare? Which dataset do you expect to perform better? Clearly explain why you suspect that. Use no more than 75 words.

The 28*28 image has a higher resolution compared to 8*8, and the former is more understandable for human vision. Because of the higher resolution, the former dataset contains more pixels for each image, which means more information can obtain from it.

28*28 dataset contains more features (suppose that each pixel is a feature) for each image. Assuming that the ability of an algorithm is enough to handle all pixels well, 28*28 will perform better.

2.2.1 (3 points) Examine the features of both the datasets and decide if you need to do any data cleaning or preprocessing. If not, clearly explain why not. If yes, clearly explain why and what you did. Use no more than 100 words. (You might want to read the additional reading materials).

The features are the intensities of the pixels. Both datasets are divided by 255, to map the original range of [0, 255] to [0, 1]. This is a kind of normalization, which can aid some algorithms which use gradient descent. Because the large inputs will lead to a large gradient each time, the learning rate should be very small. However, the learning rate is initialized randomly. If it is set too large at the beginning, it may skip the optimum.

We also flatten the image into a vector, since an estimator only accepts dimensions less or equal to 2.

2.3.1 (1 point) Now set up your experiment. Clearly explain how you divided the data and how you ensured a valid measurement. Use no more than 100 words.

We decide to use KFold cross-validation to train and test our models because it can make full use of the dataset and avoid the overfitting problem to some extent. The value of n_splits is set to 5, which means it will use 1 fold for validating, the other 4 folds for training in one round. In terms of ensuring our measurements, we plotted the learning curve for each training process in order to compare the accuracy. (we do not use F1 because the dataset is balanced)

2.3.2 (2 points) Fit the five algorithms using Scikit-learn's default hyper-parameters. Create a useful plot that shows the performances of the algorithms. Clearly explain what these plots tell us about the performances of the algorithms. Also, clearly explain why you think some algorithms perform better than others and why some of them perform better on one dataset than the other. Use no more than 200 words and two plots (but 1 is sufficient).

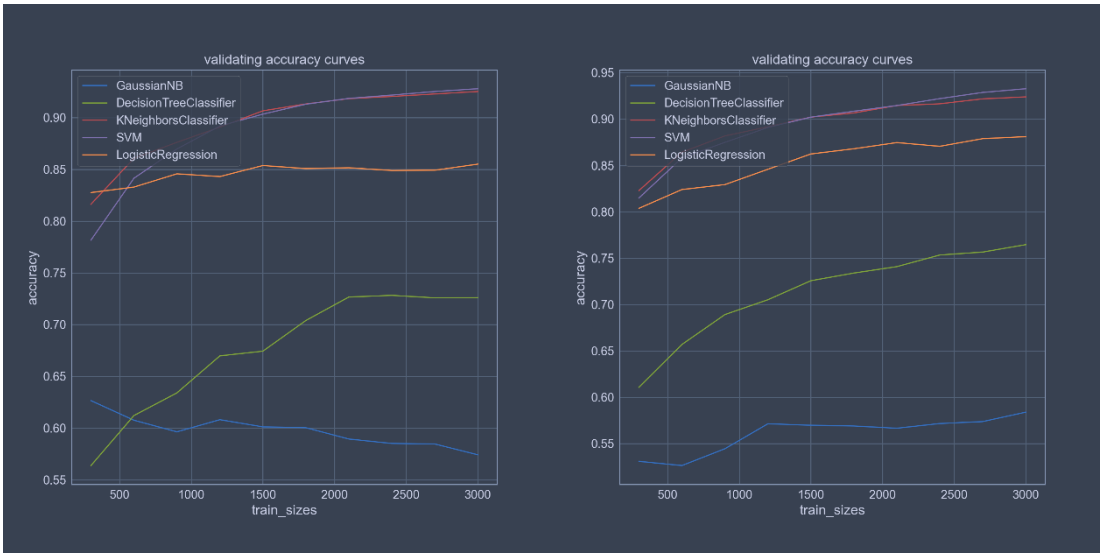

Figure 5 28x28 learning curves

Figure 6 8x8 learning curves

Figure 1 shows the performance of five algorithms on the 28x28-pixel dataset, while figure 2 shows the performance on the 8x8-pixel dataset. No matter in which dataset, SVM always presents the best learning ability, as it has the highest accuracy.

The performance of the algorithm depends on the real distribution of the samples and the initial hyper-parameter. Since the digital writings are manufactured, Gaussian may perform worst on this kind of data. The ability of Decision Tree to deal with the numeric features is weak, so, it may perform worse on this kind of data compared to discrete features. The rest of the three algorithms are doing well. They may have appropriate hyperparameters initially. If SVM performs well on this dataset, this dataset probably has a relatively clear margin of classification, which means there are a few overlapping samples from the ten different classes. K-Neighbour places in second, as it has a slightly lower accuracy compared to SVM. Unlike the American census dataset, the links between pixels and links between images may stronger, leading to the better performance of K-Neighbor. Logistic regression finishes third, as the gradient descent may be quite suitable to process image data.

2.3.3 (2 points) Now perform hyper-parameter tuning on the key hyper-parameters you have previously identified. Clearly explain what you did and how you did this. Use no more than 200 words.

Systematic: In order to exhaustively search the best parameters, we adapt GridSearchCV. Using this method, we can define a group of values for each parameter of a model, which will be tested independently each time, and GridSearchCV will return the best combination of values for each algorithm.

Fairness: We cannot access to the best values before training and validating. Therefore, we need to trial. Initially, we will set a rough range of values to test their performances. Then, we will narrow down the ranges of the parameters according to the feedback of GridSearchCV. At the same time, we will check the learning curve of the current best algorithm to monitor the overfitting problem.

Trade-off: The first limitation is the number of samples; therefore, every round we will feed the model with all samples. The second limitation is the wide range of parameters, and especially it will spend a long period for training if the ranges are set too wide.

Therefore, we need to stop narrow down when the accuracy is not significantly promoted after tuning.

2.3.4 (2 points) Compare the performance of the algorithms with and without hyper-parameter tuning. Also, make a comparison with your original baseline. How did the tuning affect your result? Clearly explain the results and the differences. Use no more than 200 words and two plots (but 1 is sufficient).

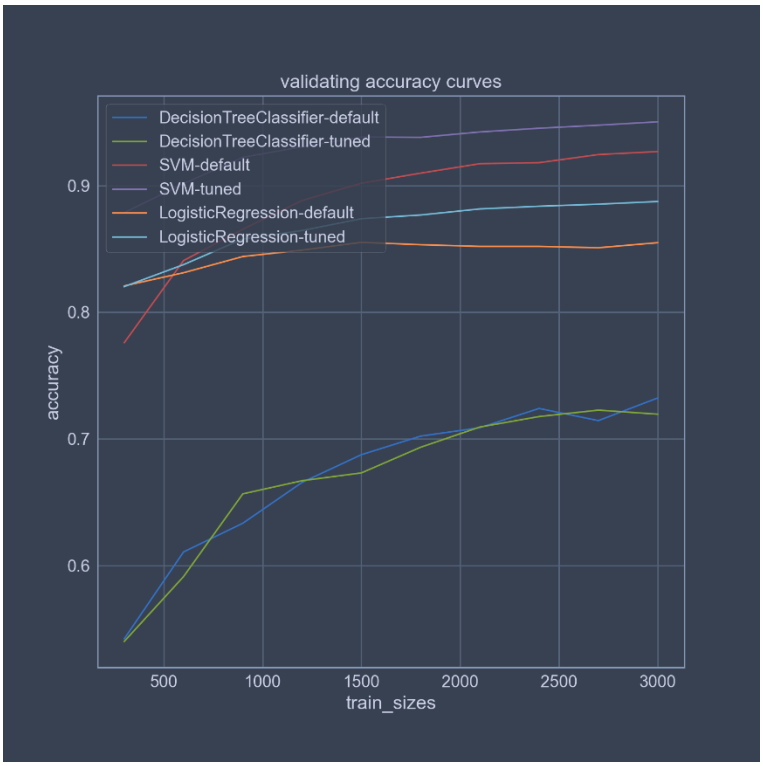

Figure 7 28x28 learning curves after tuning

In figure 3, it shows the learning curves of the algorithms after being tuned. Gaussian has no hyper-parameter tuning requirements, and K-Neighbor has the same values of the hyperparameter before and after being tuned, so, they are not presented in the graph. Due to the overfitting problem of Decision Tree algorithm, which is caused by a large quantity of input features, we have to restrict its max depth rather than use the default value “none”. Its accuracy goes down, but the overfitting is reduced.

Turning to the current best two models, SVM is still on top, followed by logistic regression. The learning abilities of both algorithms have been improved after tuning. As we decrease the hyperparameter “C”, which is a kind of intensity of the penalty, the models tend to become more acceptable for other new samples. Tuning hyperparameters has a small but noticeable effect on the performance of an algorithm.

2.3.5 (1 point) Compare the performance of the algorithms with the 8x8 and 28x28 features. What effect do the additional features have? Also, state what you think causes this effect. Use no more than 75 words.

The results are quite similar for both datasets. The additional features cannot significantly improve the performance of the algorithms. Higher resolution may contribute more to human understanding, but it means less to the machine. It seems like the lower resolution keeps the structure of the data. This can be a useful method for training high-quality images, in which we can reduce the pixels of an image similar to what convolutional layers does in CNN.

2.3.6 Select your best algorithm for this dataset and use it to make your predictions for the unknown samples. Feel free to use either the 8x8 or 28x28 features. Please note in your report which algorithm and feature set you chose.

We decide to choose Support Vector Machine with the following hyper-parameters and use 28x28-pixel dataset.

- C=6,
- kernel="rbf",
- random_state=40

3. Conclusion

3.1.1 (3 points) Which conclusions can we draw about the five algorithms examined during this assignment? For each algorithm briefly discuss the key thing you noticed about it during this assignment. Use no more than 250 words in total (+-50 words per algorithm).

- Gaussian:

Due to the limitation of the assumption of independent predictors, Gaussian performs the poorest in both tasks. Therefore, when dealing with the tasks relevant to social investigation and image recognition, it may not be the first choice.

- DecisionTreeClassifier:

DecisionTreeClassifier comes in at a third place in the first task but performs worse in the second task. Since predicting continuous values is a weaker point for this classifier, when dealing with the dataset containing lots of continuous features, it cannot be the first choice. In American census dataset, there are 8 discrete features, so it doesn't perform the poorest.

- KNeighborsClassifier:

Rather than "drawing boundaries for different classes", it uses "similarity with neighbours" to judge the class of a sample. When people distinguish two images, they compare the similarity between pixels. Similarly, so it can perform well in the second task. Also, after re-scale pre-processing, it can perform even better.

- SVC:

SVC is the best classifier in our tasks because it can change its kernel to fit various datasets. It works well with even unstructured and semi-structured data like images and trees. It is a good tool to solve local optima problem with high dimensions.

- LogisticRegression:

In both tasks, although it is not the best one, it performs better than several algorithms. We thought this kind of algorithm can be extended to be a more complex and powerful classifier. Indeed, we found that many other powerful and widely used algorithms are based on it, like neural networks.