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.

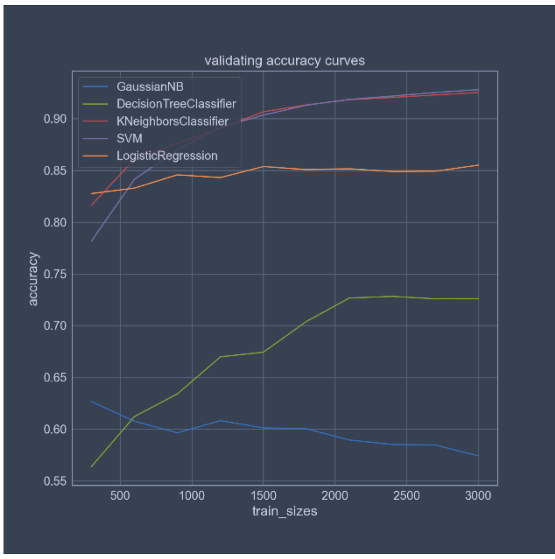
2.2.1 (3 points) Examine the features of both the datasets and decide if you need to do any data cleaning or preprocessing. If not, clearly explain why not. If yes, clearly explain why and what you did. Use no more than 100 words. (You might want to read the additional reading materials).

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]. It is a kind of normalization. It can contribute to 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. Learning rate is initialized randomly. If it is set to large at the beginning, it may skip the trough. We also flatten the pixels, as an estimator only accepts dimension 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 KFlod 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 rest 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).



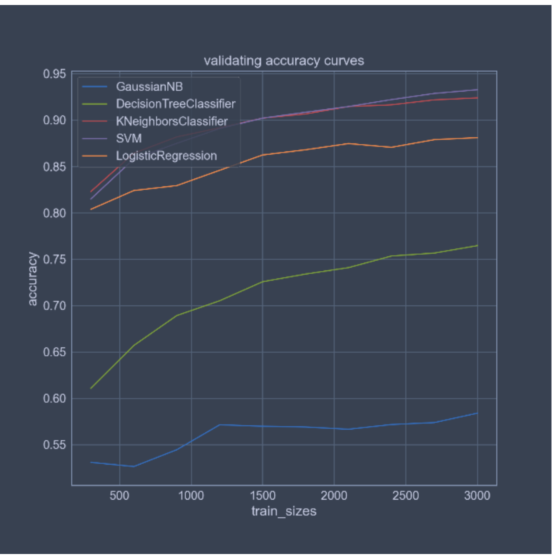


Figure 1 28x28 learning curves

Figure 2 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 present the best learning ability, owning 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. The rest of the three algorithms are doing well. They may have appropriate hyperparameters initially. If SVM performances well on this dataset, this dataset probably has a relatively clear margin of classification, which means there are a few overlapping samples of ten different classes. K-Neighbor locates at the second position, which has a slightly lower accuracy than SVM. Unlike American census dataset, the links between pixels and links between images may stronger, leading the better performance of K-Neighbor. Logistic regression owns the third ranking, because 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).

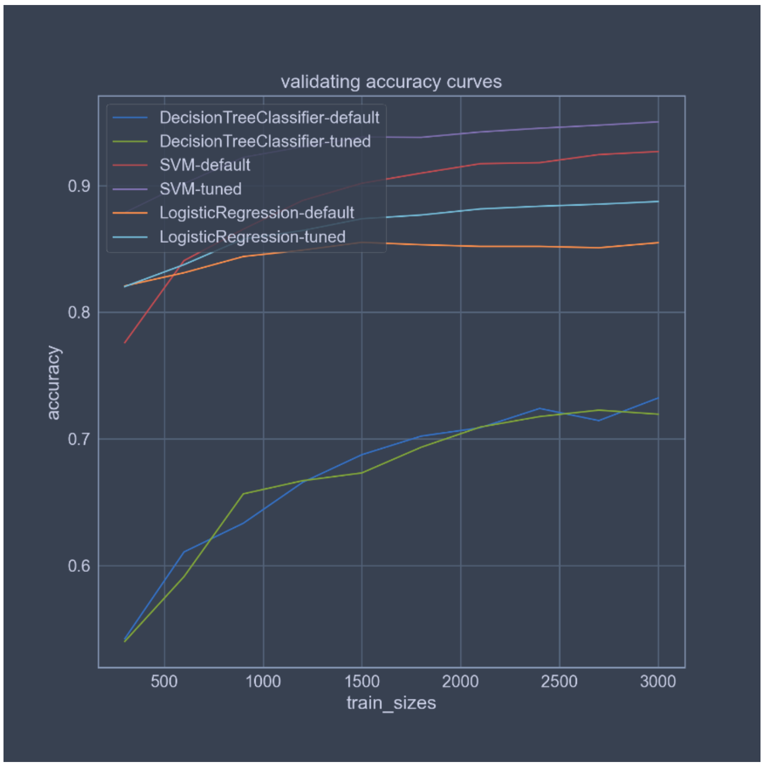


Figure 3 28x28 learning curves after tuning

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 it. Due to the overfitting problem of Decision Tree algorithm, which is caused by a large quantity of features input, 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 at the top score, 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 can promote the performance of an algorithm to some extent.

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. 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