Project2 Report:  
Handwritten digits

# Introduction

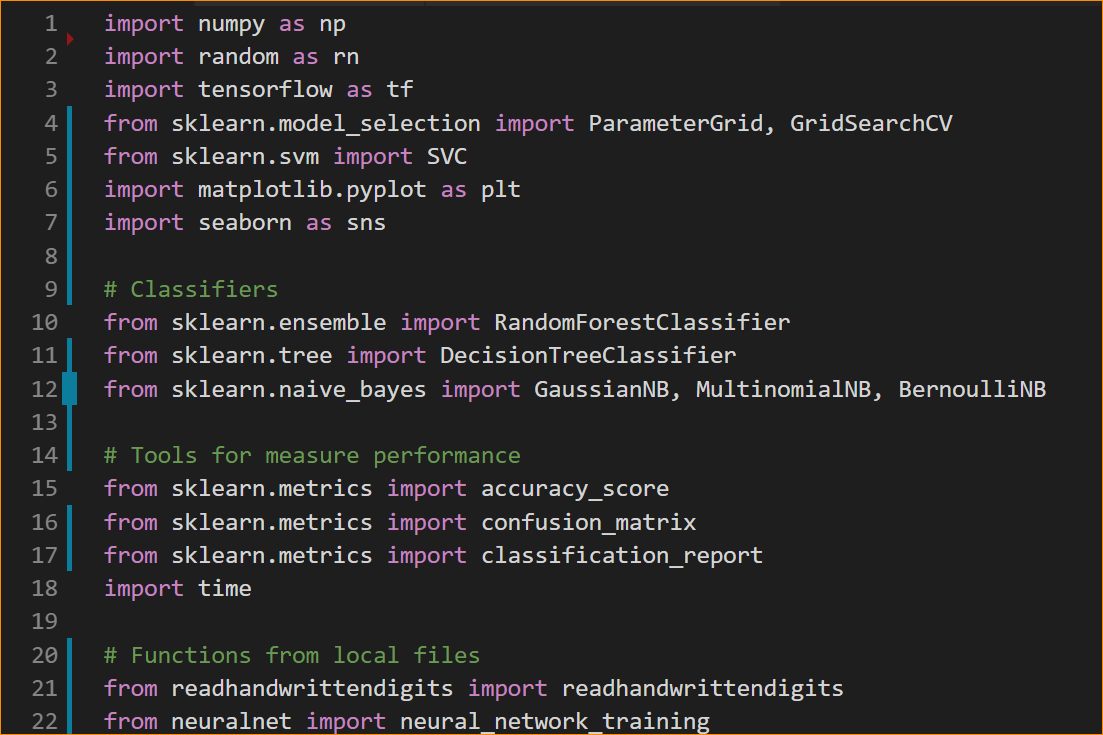
This program uses three different types of classification algorithms to recognizes different handwritten digits, and the goal is to classify handwritten digits correctly based on the training that is done for each classifier.

The program is given a set of data containing handwritten digits. The data is split into a training set and a test set. The training set is used to train the algorithms. This is necessary process as they will be able to predict a handwritten digit to an actual digit. At the end of the training, the algorithms are going to predict handwritten digits in the test data. This confirms the learning process of the algorithms to see if the algorithms are doing what it is supposed to do.

Using machine learning to tackle this issue may improve resource and time for this small company.

# Preprocessing steps

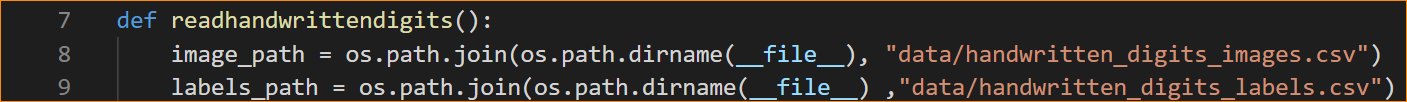
The steps needed to start the project was to install the necessary program, in this case, Python. Necessary libraries and imports must also be installed to use the machine learning algorithms and other features.



The data that is used on the algorithms are downloaded from MittUiB, which is from the MNIST database. The data contains 70,000 examples which is used to train the algorithm and confirm that they are working.

The handwritten files are saved in a folder ‘data’ and are accessed by python file ‘readhandwrittendigits.py’.

The python file uses relative path to access the data, which is to avoid use of absolute path and having the files in the same folder as the python file.



A Python version 3.6 was installed instead of the newest version 3.7, because the version 3.7 were not compatible with the library tensorflow which is used to run neural network algorithm from keras.

# Candidate algorithms and choice of candidate

The process behind choosing an algorithm was to pick three types of classification according to their performance.   
In this project, the chosen classifications are Naive Bayes, Random Forest and Neural Network, and each of these classifications are very different from each other.

## Naive Bayes

Naive Bayes is more lightweight which means that is quicker and faster than the other two. It performs well when we have multiple classes and are working with text classification. It converges quicker if the conditional independence holds, resulting in less training data needed. It is quicker because it requires less model training time relative to the two other algorithms.

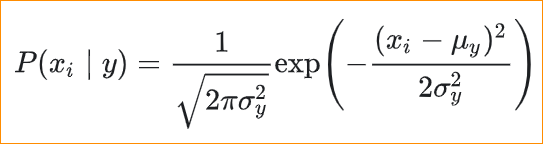
Naive Bayes also doesn't need hyperparameters because it doesn't need lot of data to perform well. It only needs enough data to understand the probabilistic relationship of each attribute with the output variable.

There are different types of Naive Bayes models that uses different formulas to calculate the likelihood, and since Naive Bayes are fast the program uses three types of Naive Bayes algorithms: Gaussian, Multinomial and Bernoulli.

Gaussian Naïve Bayes

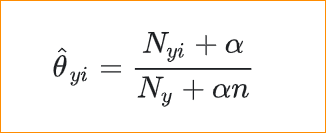
Gaussian is the easiest algorithm to work with because you only need to estimate the mean and the standard deviation from the given training data.

The formula used in Gaussian Naive Bayes to find the likelihood of the features:



Multinomial Naive Bayes

Multinomial Naive Bayes are used in text classification and estimate each parameter with a smoothed version of maximum likelihood:



Bernoulli Naive Bayes

There may be multiple features but each one is assumed to be binary-valued variable, according to Bernoulli Boolean. Therefore, the algorithm requires samples to be represented as binary-valued features vectors, which is handled by the BernoulliNB classifier.

The decision rule for this classifier is based on:

## 

## Random Forest

Random Forest is a flexible algorithm. It doesn't require much hyper-parameter tuning to output a good result. By the name, the Random Forest is essentially a collection of Decision Trees and was therefore chosen instead of Decision Tree (see more about Decision Tree under). The Random Forest is a form for Bagging method, because the algorithm takes *n* Decision Trees and trains each of them separately. At the end, the algorithm then takes the average of the predictions to get a result. With Random Forest there is less chance for overfitting, because bagging reduces variance which helps to avoid overfitting.

## Neural Network

Neural network is a machine learning algorithm used to model complex patterns in datasets using layers such as input, hidden and output layers and activation functions. Each layer has many neurons where the neurons are use for feature-detecting. If we have enough number of layers and neurons, the network can perform very well as image recognition. The training in neuron network uses labeled dataset of inputs that are tagged with their intended output response. In each layer, the data is calculated with a weight and an activation function.

Neural networks are good model with images and is reliable for tasks involving many features because of the splitting it does to layered network.

Neural network will perform better when given a large dataset, making it more accurate when predicting data, and once a network is trained, it can predict data very fast.

## Other classifications

Support Vector Machine

There are other classifications that were considered, like Support Vector Machine and Decision Tree Classifier. The SVM is are great machine learning algorithm for image recognition. SVM is fast and accurate with a small dataset and few classifications but performs poorly when given a large dataset. SVM are also poor for multiclass classification which we have in our case with ten classifiers.

My observation with SVM is that it took too long time to finish with the given dataset. It never finished training the model and therefore was not chosen as my candidate.

Decision Tree Classifier

Decision Trees are easy to interpret and visualize and can handle both numerical and categorical data. Decisions Trees can also perform well on large datasets while being fast. With that said, Decision Trees tend to overfit, especially when the data is large because the tree may become very deep. That is why Random Forest is chosen over Decision Tree, because Random Forest reduces the chance for overfitting.

## Choice of hyperparameters

The advantage

## Conclusion

The three candidates for this project are Naive Bayes, Random Forrest and Neural Network. Naive Bayes algorithms are picked as a candidate, because it is fast and doesn’t need to build a new model each time it receives new data.   
While Decision Tree being a powerful classification, Random Forrest was picked as a candidate because it uses bagging method to reduce the chance for overfitting.   
Neural network can become very accurate when given lot of training data and are also easy to experiment with (layers, neurons and activation function).

All three works well with large datasets, which useful for this task.

# Performance measure

The performance of these chosen algorithms was based on time and accuracy.

For each model, we first train the classification model with .fit() on dataset X containing the handwritten images and dataset y containing the classifier. Afterward, we use .predict() on the testset X to get the predicted digits. At the end, we calculate the accuracy score by seeing how many predicted digits from .predict() were predicted correctly with regard to the testset y.

The best algorithm was determined by the highest accuracy despite having higher running time.

# Model selection schemes

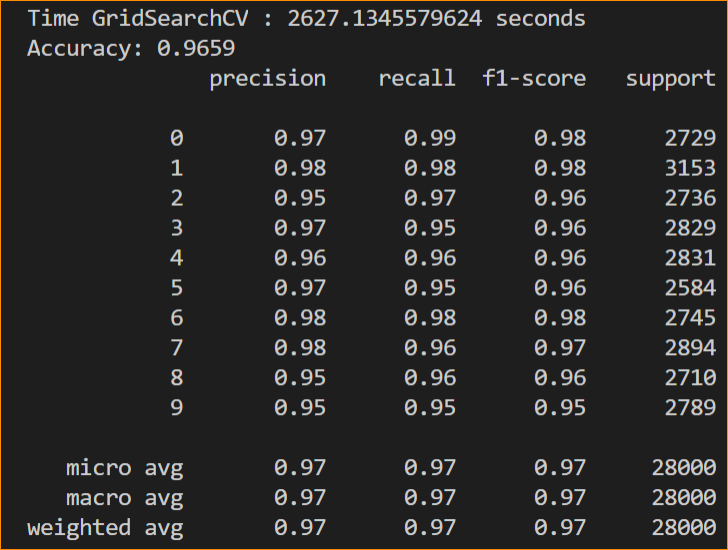
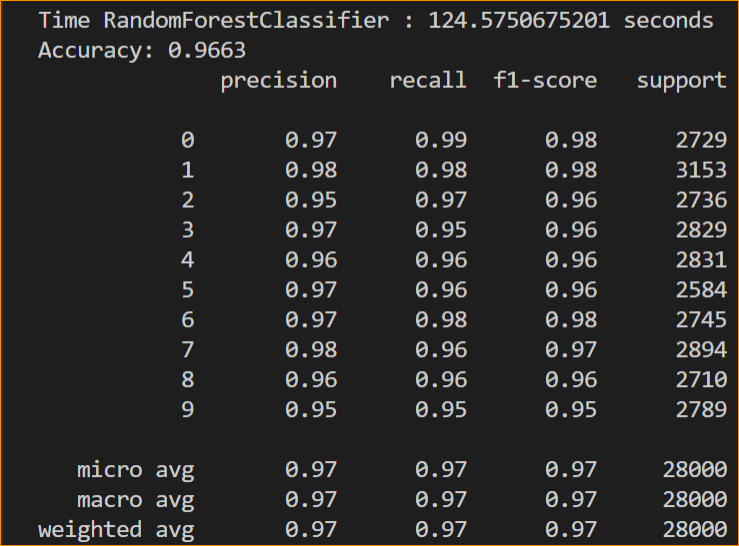
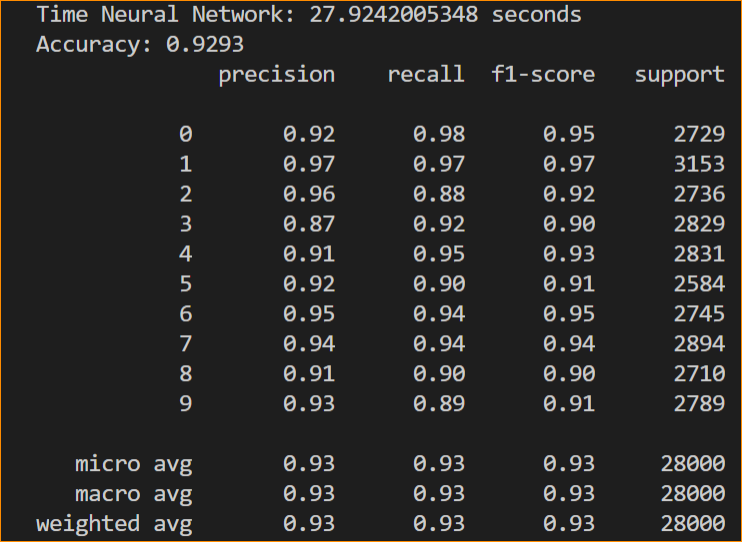
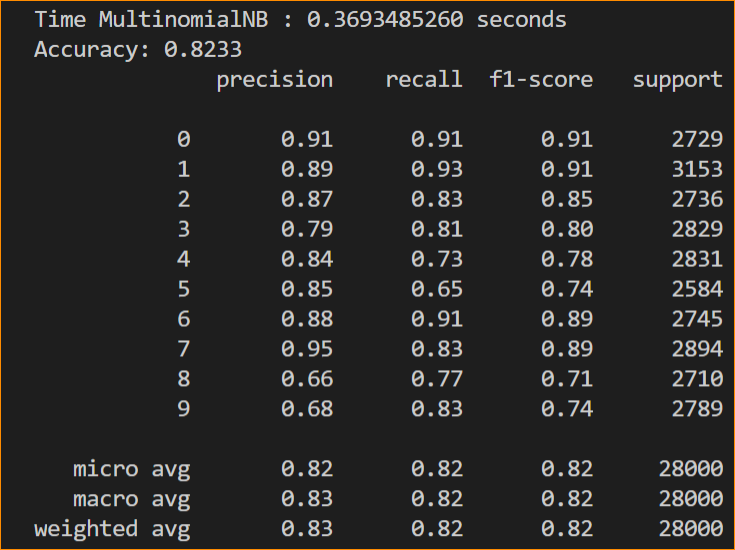
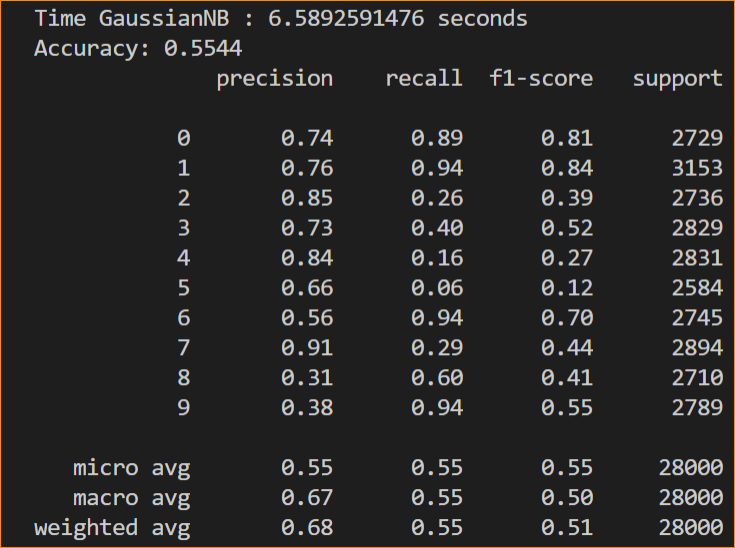
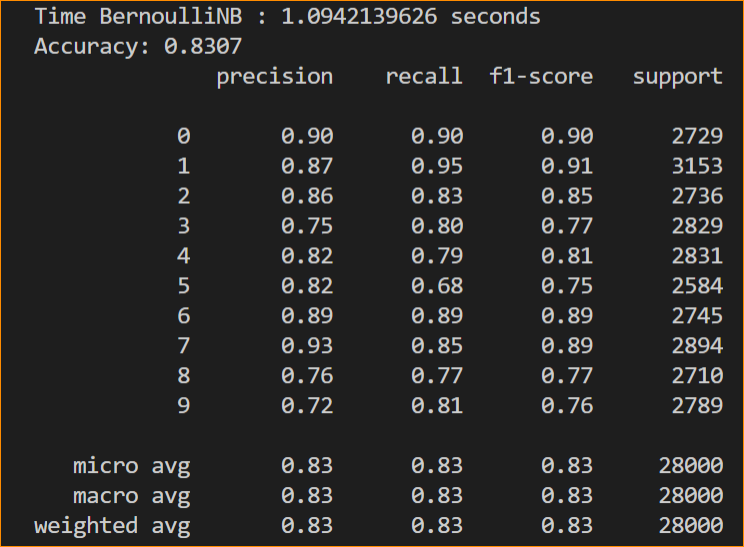
# Final classifier

The final classifier is

# Expected performance in production

# Improvement

With more time and/or computing resources, the neural network classifier will be more accurate if it the number of epochs and size of batch is increased. The computing time will get affected heavily, but the accuracy will increase.



https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd