CSE474/574: Introduction to Machine Learning (Fall 2018)
Project 3: Classification

UBIT: sassikes
Person no: 50291558

Introduction:

This project is to implement machine learning methods for the task of classification. You will first implement an ensemble of four classifiers for a given task. Then the results of the individual classifiers are combined to make a final decision.
The classification task will be that of recognizing a 28_28 grayscale handwritten digit image and identify it as a digit among 0, 1, 2, ..., 9. You are required to train the following four classifiers using MNIST digit images.

1. Logistic regression, which you implement yourself using backpropagation and tune hyperparameters.
2. A publicly available multilayer perceptron neural network, train it on the MNIST digit images
3. and tune hyperparameters.
4. A publicly available Random Forest package, train it on the MNIST digit images and tune
5. hyperparameters.
6. A publicly available SVM package, train it on the MNIST digit images and tune hyperparameters.

Experimental Setup:

The following procedure has to be followed step by step,

1. ***Extract feature values and labels from the data***: Download the MNIST dataset and USPS dataset from the Internet and process the original data file into a Numpy array that contains the feature vectors and a Numpy array that contains the labels.

2. ***Data Partition***: The MNIST and USPS datasets is originally partitioned into a training set and a testing set. You will use this partition and train your model on the training set.

3. ***Train model parameter***: For a given group of hyper-parameters such as the number of layers and the number of nodes in each layer, train the model parameters on the training set of MNIST.

4. ***Tune hyper-parameters***: Validate the classification performance of your model on the validation set. Change your hyper-parameters and repeat step 3. Try to find what values those hyper-parameters should take so as to give better performance on the testing set.

5. ***Evaluate on testing sets***: Test the trained models on both MNIST test set and USPS data. Discuss your findings.

6. Report your observations.

UBIT: sassikes
Person number: 50291558

Logistic Regression:

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

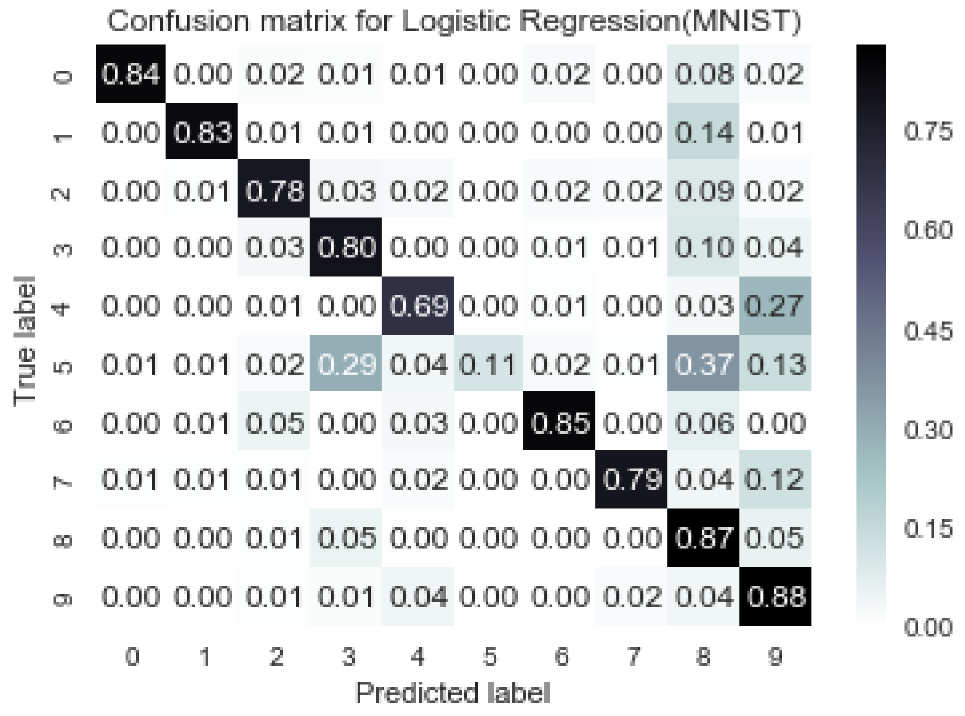
Model
Output = 0 or 1
Hypothesis => Z = WX + B
hΘ(x) = softmax (Z)

$$\mathrm{Cost}(h_\Theta(x), Y(\mathrm{actual})) = \begin{cases} -\log(h_\Theta(x)) & \text{if } y=1 \\ -\log(1-h_\Theta(x)) & \text{if } y=0 \end{cases}$$

The logistic regression has been implemented using softmax function. The model has been trained using MNIST dataset and the tested using both MNIST and USPS datasets.

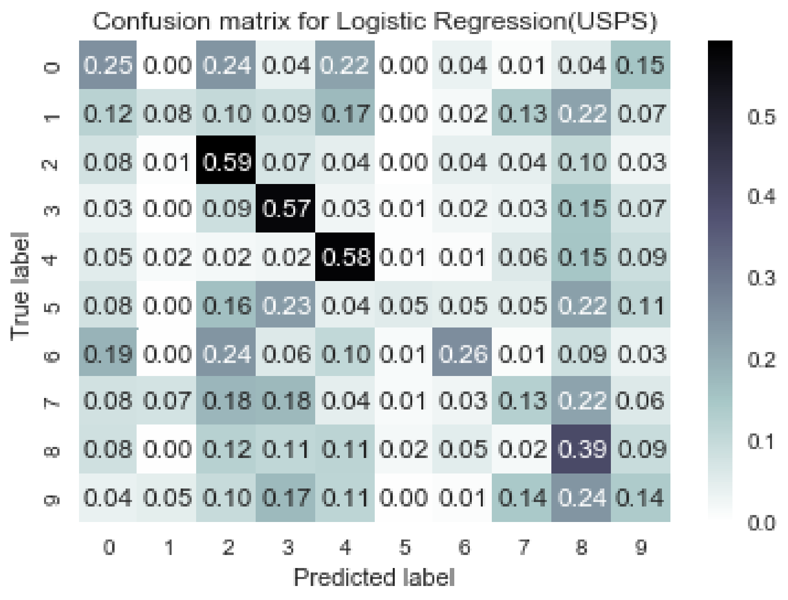
The accuracy and confusion matrix are as follows,

Accuracy for MNIST using Logistic Regression: 0.75



	0	1	2	3	4	5	6	7	8	9
0	0.84	0.00	0.02	0.01	0.01	0.00	0.02	0.00	0.08	0.02
1	0.00	0.83	0.01	0.01	0.00	0.00	0.00	0.00	0.14	0.01
2	0.00	0.01	0.78	0.03	0.02	0.00	0.02	0.02	0.09	0.02
3	0.00	0.00	0.03	0.80	0.00	0.00	0.01	0.01	0.10	0.04
4	0.00	0.00	0.01	0.00	0.69	0.00	0.01	0.00	0.03	0.27
5	0.01	0.01	0.02	0.29	0.04	0.11	0.02	0.01	0.37	0.13
6	0.00	0.01	0.05	0.00	0.03	0.00	0.85	0.00	0.06	0.00
7	0.01	0.01	0.01	0.00	0.02	0.00	0.00	0.79	0.04	0.12
8	0.00	0.00	0.01	0.05	0.00	0.00	0.00	0.00	0.87	0.05
9	0.00	0.00	0.01	0.01	0.04	0.00	0.00	0.02	0.04	0.88

Accuracy for USPS using Logistic Regression: 0.30

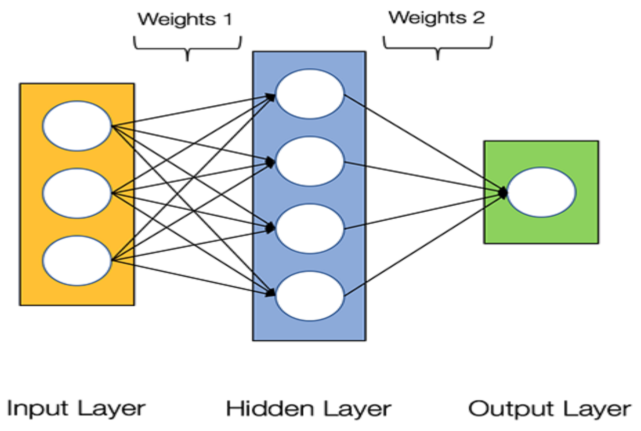


True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	0.25	0.00	0.24	0.04	0.22	0.00	0.04	0.01	0.04	0.15
1	0.12	0.08	0.10	0.09	0.17	0.00	0.02	0.13	0.22	0.07
2	0.08	0.01	0.59	0.07	0.04	0.00	0.04	0.04	0.10	0.03
3	0.03	0.00	0.09	0.57	0.03	0.01	0.02	0.03	0.15	0.07
4	0.05	0.02	0.02	0.02	0.58	0.01	0.01	0.06	0.15	0.09
5	0.08	0.00	0.16	0.23	0.04	0.05	0.05	0.05	0.22	0.11
6	0.19	0.00	0.24	0.06	0.10	0.01	0.26	0.01	0.09	0.03
7	0.08	0.07	0.18	0.18	0.04	0.01	0.03	0.13	0.22	0.06
8	0.08	0.00	0.12	0.11	0.11	0.02	0.05	0.02	0.39	0.09
9	0.04	0.05	0.10	0.17	0.11	0.00	0.01	0.14	0.24	0.14

Neural Networks:

Most introductory texts to Neural Networks brings up brain analogies when describing them. Without delving into brain analogies, I find it easier to simply describe Neural Networks as a mathematical function that maps a given input to a desired output.

Neural Networks consist of the following components
An input layer, x
An arbitrary amount of hidden layers
An output layer, ŷ
A set of weights and biases between each layer, W and b
A choice of activation function for each hidden layer, σ. In this tutorial, we'll use a Sigmoid activation function.
The diagram below shows the architecture of a 2-layer Neural Network (note that the input layer is typically excluded when counting the number of layers in a Neural Network)



UBIT: sassikes
Person number: 50291558

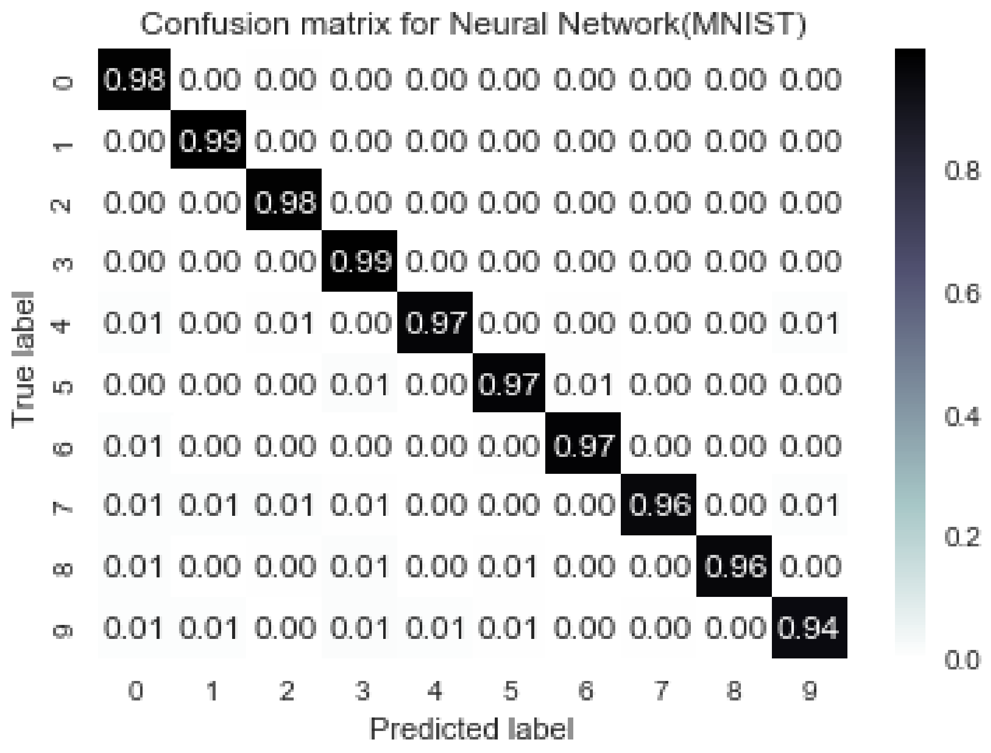
The neural network here is build using the keras framework. The input number of the nodes of the neural network should be same as the number of features and the output number of nodes should be same as the number of the classifiers required.

Tuning hyperparameters:

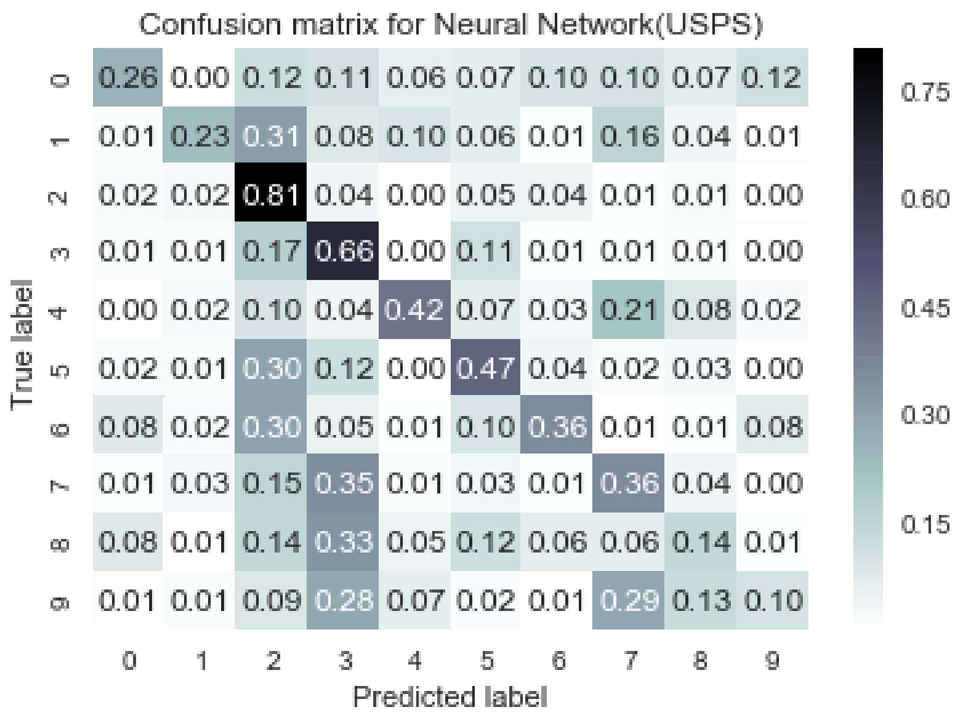
# hidden layers	# hidden layer units	Epochs	Optimizer	Batch size	Accuracy
2	32	10	Adam	128	94
2	397	10	Adam	128	97
2	32	10	SGD	10	96
2	297	10	SGD	10	90

Considering the best setting we get the following confusion matrix and accuracies for MNIST and USPS:

Final accuracy (on MNIST data): 0.97



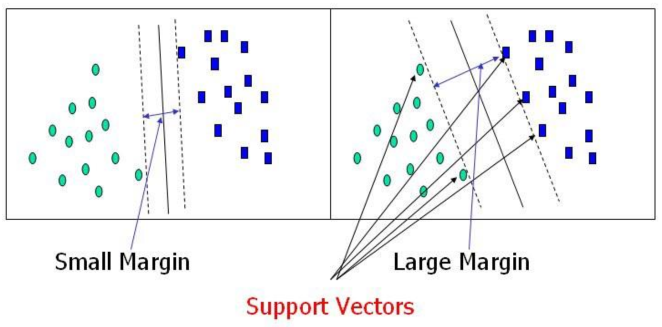
Final accuracy (on USPS data): 0.35



Support Vector Machine:

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

SVM to core tries to achieve a good margin. *A margin is a separation of line to the closest class points.*A *good margin* is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.



UBIT: sassikes
Person number: 50291558

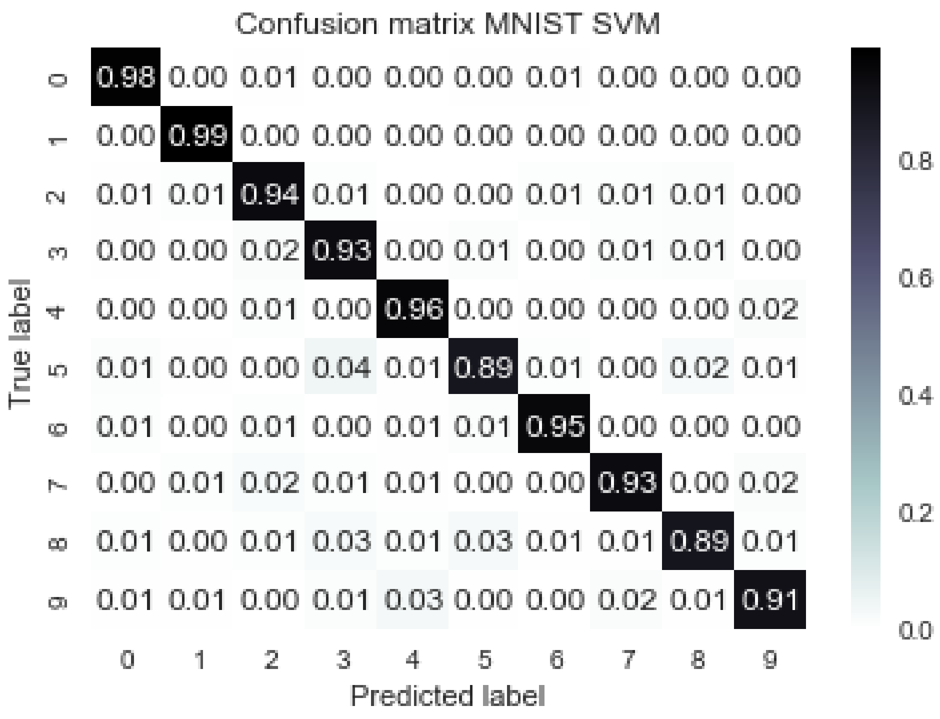
Settings:

Using linear kernel (all other parameters are kept default)
Final accuracy for SVM (on MNIST data): 0.91
Final accuracy for SVM (on USPS data): .31

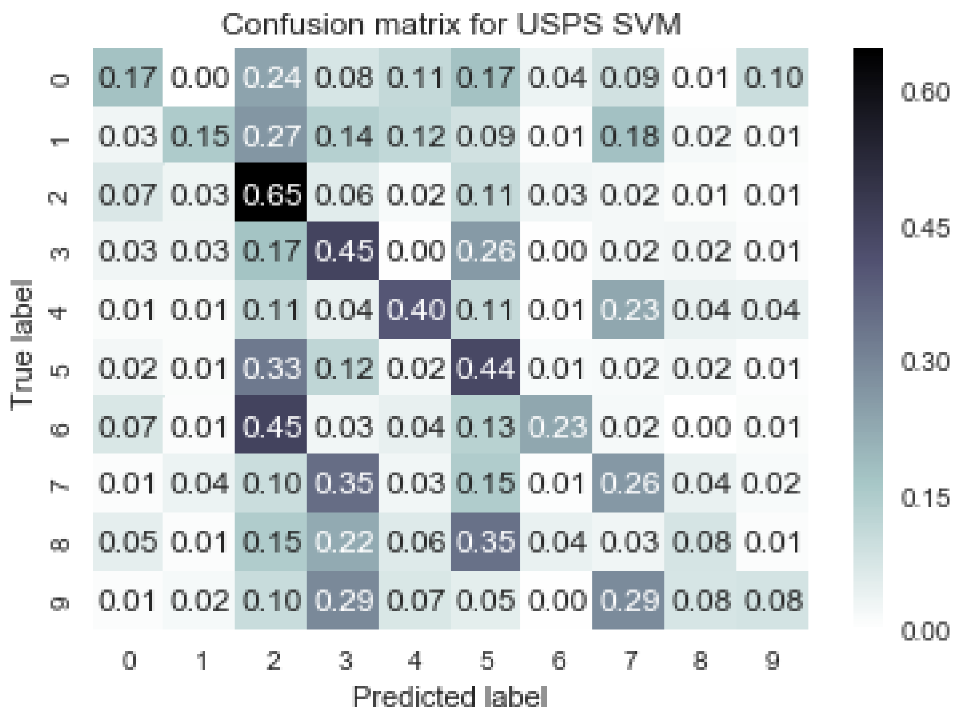
Using radial basis function with value of gamma setting to 1 (all other parameters are kept default)
Final accuracy for SVM (on MNIST data):.90
Final accuracy for SVM (on USPS data): .27

Using radial basis function with value of gamma setting to default (all other parameters are kept default)
Final accuracy for SVM (on MNIST data):.94
Final accuracy for SVM (on USPS data): .39

Considering the best setting we get the following confusion matrix and accuracies for MNIST and USPS:



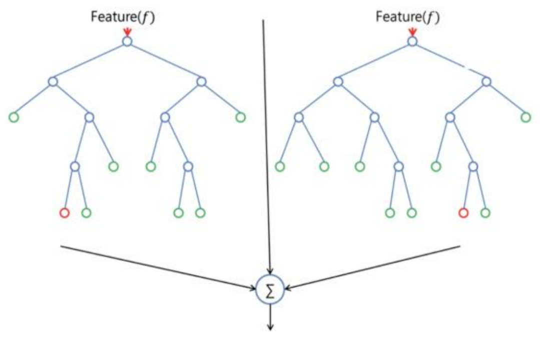
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	0.98	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00
1	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.01	0.01	0.94	0.01	0.00	0.00	0.01	0.01	0.01	0.00
3	0.00	0.00	0.02	0.93	0.00	0.01	0.00	0.01	0.01	0.00
4	0.00	0.00	0.01	0.00	0.96	0.00	0.00	0.00	0.00	0.02
5	0.01	0.00	0.00	0.04	0.01	0.89	0.01	0.00	0.02	0.01
6	0.01	0.00	0.01	0.00	0.01	0.01	0.95	0.00	0.00	0.00
7	0.00	0.01	0.02	0.01	0.01	0.00	0.00	0.93	0.00	0.02
8	0.01	0.00	0.01	0.03	0.01	0.03	0.01	0.01	0.89	0.01
9	0.01	0.01	0.00	0.01	0.03	0.00	0.00	0.02	0.01	0.91



Random Forest:

Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The “forest” it builds, is an ensemble of Decision Trees, most of the time trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:



Trying out different number of estimators:

N_estimators=20

Final accuracy for Random Forest (on MNIST data): .96
Final accuracy for Random Forest (on USPS data): .35

N_estimators=10

Final accuracy for Random Forest (on MNIST data): .95
Final accuracy for Random Forest (on USPS data): .30

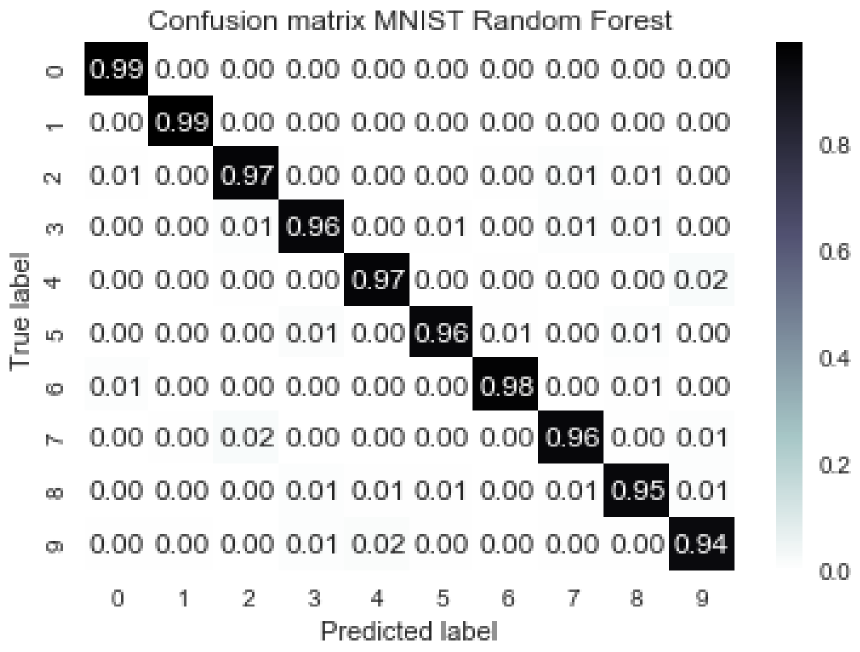
N_estimators=30

Final accuracy for Random Forest (on MNIST data): .96
Final accuracy for Random Forest (on USPS data): .37

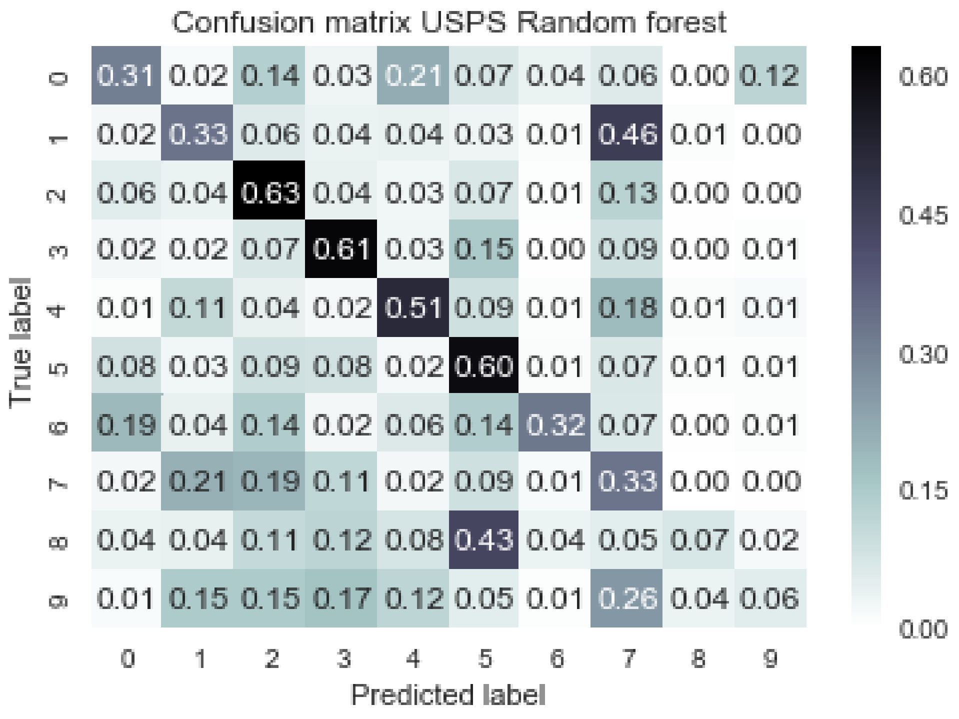
N_estimators=40

Final accuracy for Random Forest (on MNIST data): .97
Final accuracy for Random Forest (on USPS data): .38

Considering the best setting we get the following confusion matrix for MNIST and USPS:



	0	1	2	3	4	5	6	7	8	9
0	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.01	0.00	0.97	0.00	0.00	0.00	0.00	0.01	0.01	0.00
3	0.00	0.00	0.01	0.96	0.00	0.01	0.00	0.01	0.01	0.00
4	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.02
5	0.00	0.00	0.00	0.01	0.00	0.96	0.01	0.00	0.01	0.00
6	0.01	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.01	0.00
7	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.96	0.00	0.01
8	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.01	0.95	0.01
9	0.00	0.00	0.00	0.01	0.02	0.00	0.00	0.00	0.00	0.94



	0	1	2	3	4	5	6	7	8	9
0	0.31	0.02	0.14	0.03	0.21	0.07	0.04	0.06	0.00	0.12
1	0.02	0.33	0.06	0.04	0.04	0.03	0.01	0.46	0.01	0.00
2	0.06	0.04	0.63	0.04	0.03	0.07	0.01	0.13	0.00	0.00
3	0.02	0.02	0.07	0.61	0.03	0.15	0.00	0.09	0.00	0.01
4	0.01	0.11	0.04	0.02	0.51	0.09	0.01	0.18	0.01	0.01
5	0.08	0.03	0.09	0.08	0.02	0.60	0.01	0.07	0.01	0.01
6	0.19	0.04	0.14	0.02	0.06	0.14	0.32	0.07	0.00	0.01
7	0.02	0.21	0.19	0.11	0.02	0.09	0.01	0.33	0.00	0.00
8	0.04	0.04	0.11	0.12	0.08	0.43	0.04	0.05	0.07	0.02
9	0.01	0.15	0.15	0.17	0.12	0.05	0.01	0.26	0.04	0.06

Ensemble classifier:

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

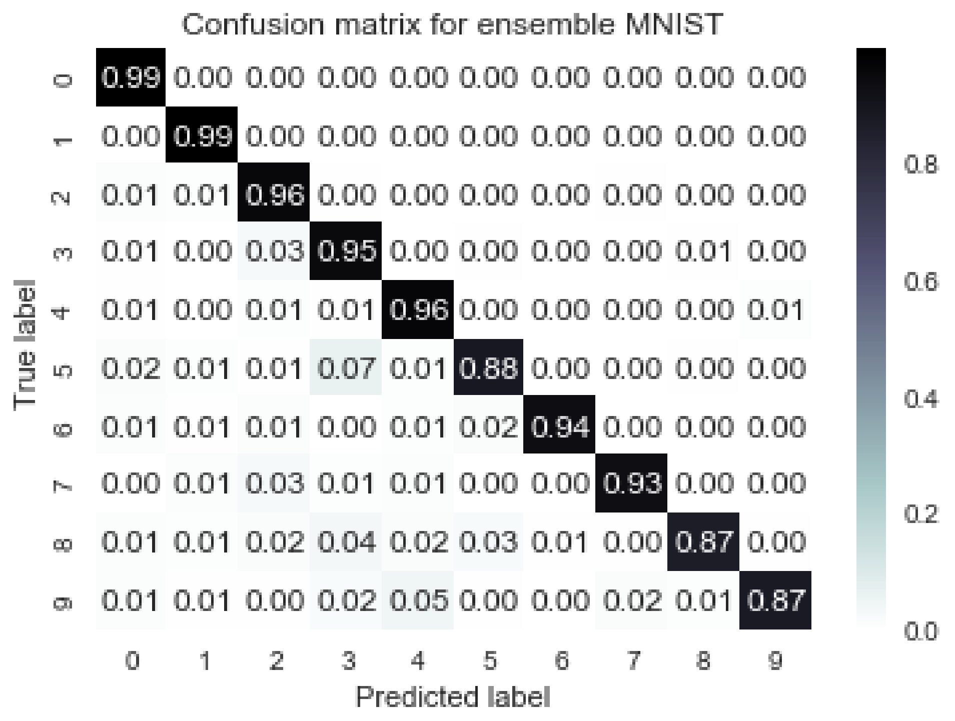
Examples: Bagging methods, Forests of randomized trees, …

By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

Examples: AdaBoost, Gradient Tree Boosting, …

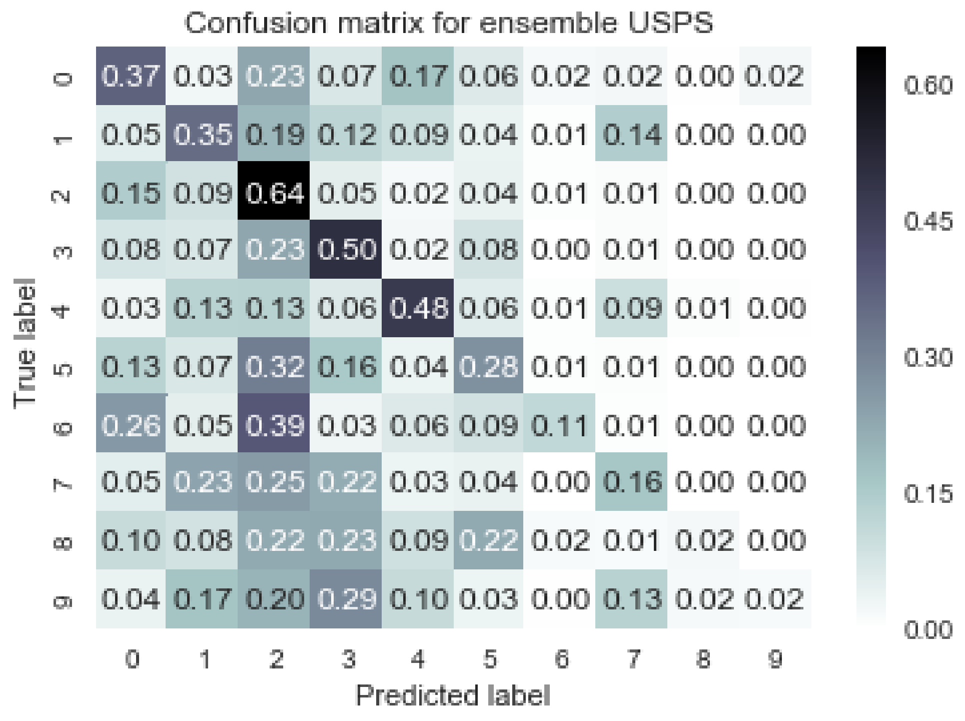
The performance of the ensemble classifier created using Majority voting is as follows:

UBIT: sassikes
Person number: 50291558



	0	1	2	3	4	5	6	7	8	9
0	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.01	0.01	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.01	0.00	0.03	0.95	0.00	0.00	0.00	0.00	0.01	0.00
4	0.01	0.00	0.01	0.01	0.96	0.00	0.00	0.00	0.00	0.01
5	0.02	0.01	0.01	0.07	0.01	0.88	0.00	0.00	0.00	0.00
6	0.01	0.01	0.01	0.00	0.01	0.02	0.94	0.00	0.00	0.00
7	0.00	0.01	0.03	0.01	0.01	0.00	0.00	0.93	0.00	0.00
8	0.01	0.01	0.02	0.04	0.02	0.03	0.01	0.00	0.87	0.00
9	0.01	0.01	0.00	0.02	0.05	0.00	0.00	0.02	0.01	0.87

Final accuracy for ensemble (on MNIST data): 0.94



	0	1	2	3	4	5	6	7	8	9
0	0.37	0.03	0.23	0.07	0.17	0.06	0.02	0.02	0.00	0.02
1	0.05	0.35	0.19	0.12	0.09	0.04	0.01	0.14	0.00	0.00
2	0.15	0.09	0.64	0.05	0.02	0.04	0.01	0.01	0.00	0.00
3	0.08	0.07	0.23	0.50	0.02	0.08	0.00	0.01	0.00	0.00
4	0.03	0.13	0.13	0.06	0.48	0.06	0.01	0.09	0.01	0.00
5	0.13	0.07	0.32	0.16	0.04	0.28	0.01	0.01	0.00	0.00
6	0.26	0.05	0.39	0.03	0.06	0.09	0.11	0.01	0.00	0.00
7	0.05	0.23	0.25	0.22	0.03	0.04	0.00	0.16	0.00	0.00
8	0.10	0.08	0.22	0.23	0.09	0.22	0.02	0.01	0.02	0.00
9	0.04	0.17	0.20	0.29	0.10	0.03	0.00	0.13	0.02	0.02

Final accuracy for Ensemble (on USPS data): 0.29

UBIT: sassikes
Person number: 50291558

Results:

Based on the above implementations we wish to answer the following questions:

1. We test the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS data set. Do our results support the "No Free Lunch" theorem?

Yes, our results support "No Free Lunch" theorem as the accuracy of predictions of USPS data set is much lower than that of the predictions of MNIST on the MNIST trained classifiers.

2. Observe the confusion matrix of each classifier and describe the relative strengths/weaknesses of each classifier. Which classifier has the overall best performance?

The confusion matrices shown above describes the strengths/weaknesses of each classifier. The overall best performance was given by Neural Networks for both MNIST and USPS dataset

3. Combine the results of the individual classifiers using a classifier combination method such as majority voting. Is the overall combined performance better than that of any individual classifier?

The accuracy of the ensemble classifier is similar or better that the accuracies of the individual classifiers for both MNIST and USPS data.

References:

https://machinelearningmastery.com
https://towardsdatascience.com
https://en.wikipedia.org/
https://scikit-learn.org/stable/
https://keras.io/