Handwritten Digit Recognition using various Classification Techniques

Yashankit Shikhar
UBID: yshikhar

November 22, 2018

Abstract

The purpose of this paper is to propose Classification models such as Logistic Regression, Support Vector Machine, Random Forest, etc. to recognize different Handwritten Digits. The feature vectors of the handwritten digits images are obtained from two different sources, MNIST dataset and USPS dataset. The results of various classifiers are observed using Confusion Matrices and are used to ensemble them. The results of individual classifiers are combined using Majority Voting.

1 Introduction

In this project, several classification methods to solve the handwritten digit recognition task is discussed. The feature vectors are obtained for each image from MNIST dataset by using pickle file. The feature vectors are obtained for each image from USPS dataset by re-sizing the image to 28x28 dimension and then flattening it. The target vector takes ten values, 0 to 9 corresponding to each digit. The MNIST dataset is used to train the classifiers and the program is tested on a subset of MNIST dataset and USPS dataset. The datasets contain approximately equal numbers of each class.

2 Experiment

The aim of this project is to train a Classification model to predict which digit does the handwritten digit image resemble. Various Models of the following Classification techniques have been implemented:

1. **Multi-class Logistic Regression**
2. **Neural Network**
3. **Support Vector Machine**
4. **Random Forest**

Various Models of the following Classification ensemble techniques have also been implemented using Scikit-Learn and Mlxtend Library:

1. **Bootstrap aggregating**
2. **Boosting**
3. **Stacking**

The results of all the individual classifiers are combined using Majority Voting.

3 Classification Models

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data [1].

3.1 Multi-class Logistic Regression (Mini-batch Stochastic Gradient Descent)

Softmax regression (or, Multi-class 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. Like all regression analyses, the softmax regression is a predictive analysis [2]. A softmax regression function $\mathbf{y}(\mathbf{x}, \mathbf{w})$ is defined as,

$$y(x, w) = softmax(w^\top X) \tag{1}$$

where,

$$softmax(z) = \frac{e^z}{\sum_{i=1}^{k} e^{z_i}} \tag{2}$$

where $\mathbf{w} = (\mathbf{w}_0, \mathbf{w}_1, .., \mathbf{w}_{m-1})$ is a weight vector to be learned from training samples and X is the data matrix.The Loss Function for Logistic Regression is given as,

$$E_D(w) = \frac{1}{m}(-t^\top log(y(x, w)) - (1 - t)^\top log(1 - y(x, w))) \tag{3}$$

where $\mathbf{t} = (\mathbf{t}_0, \mathbf{t}_1, .., \mathbf{t}_n)$ is the vector of outputs. The derivative of the loss function with respect to weights is defined as,

$$\nabla E_D = \frac{1}{m} X^\top (y(x, w) - t) \tag{4}$$

We modify the solution for each data sample using:

$$w^{\tau+1} = w^\tau + \Delta w^\tau \tag{5}$$

where,

$$\Delta w^\tau = -\eta^\tau \nabla \sum_{i=1}^{m} E_D(w) \tag{6}$$

is called mini-batch weight updates. η^τ is the Learning Rate.

3.2 Neural Network

Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as “feature hierarchy" [3]. A simple Neural Network architecture is shown in Figure 1.

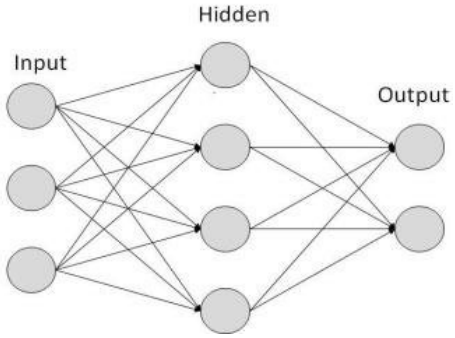

Figure 1: Neural Network architecture

3.3 Support Vector Machine

SVMs are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces [4].

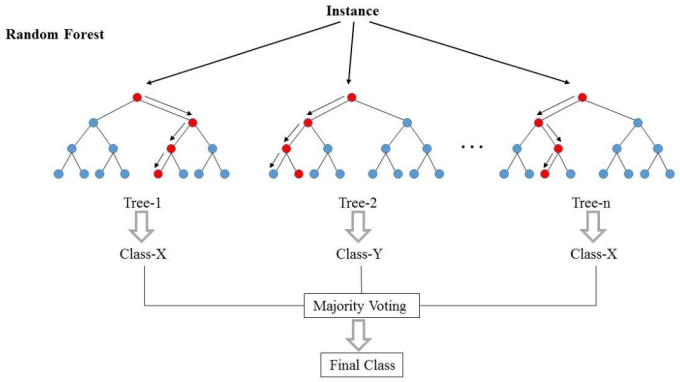

Figure 2: Random Forest Classifier

3.4 Random Forest Classifier

A random forest classifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Figure 2 shows simplified version of Random Forest Classifiers [5].

4 Ensemble Models

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone [6].

4.1 Bootstrap Aggregating

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid over-fitting [7]. Different subsets of data is fed to various classifier models and their results are combined using majority voting.

4.2 Boosting

The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. To correct the previous error, the observations that were incorrectly classified carry more weight than the observations that were correctly classified in the next iteration [8]. Figure 3 shows a simplified image of how bagging and boosting is implemented.

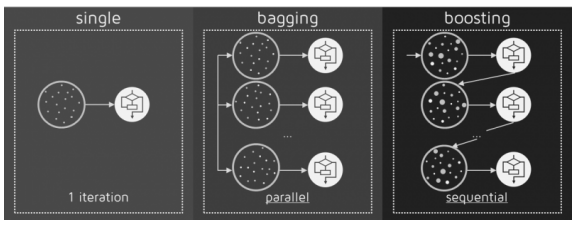

Figure 3: Comparison between Normal, Bagging, and Boosting Technique

4.3 Stacking

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs of the individual classification models in the

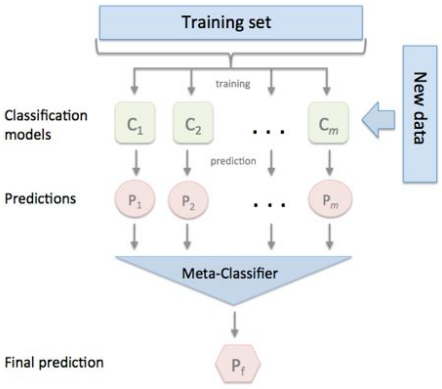

Figure 4: Stacking Technique

ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble [9]. Figure 4 shows an image of how stacking is implemented.

5 Framework

For analysis of the effect of various Classifiers listed below, the models were fitted on the testing data of MNIST dataset and whole USPS dataset to predict the digits and compare it with original labels of testing data to determine the accuracy.

- **Logistic Regression:** Three Models of Logistic Regression have been implemented. The first model implements the mini batch stochastic gradient descent and runs 200 epochs on the data with the batch size of 512. The other two models have been implemented by using the Scikit-Learn library *Logistic Regression* with solver set as *lbfgs* and multi_class is set as *ovr* and *multinomial* respectively.

- **Neural Network:** Four Models of Neural Network have been implemented. The first model is a Deep Neural Network architecture which uses the keras framework and has been implemented from scratch. The next two models have been implemented by using the Scikit-Learn library *MLPClassifier* with solver set as *sgd* and *adam* respectively. The last Model is a Convolutional Neural Network architecture.

- **Support Vector Machine:** Three Models of Support Vector Machine have been implemented using the Scikit-Learn library *SVC* with kernel set as *linear* and *rbf* and gamma parameter set as *0.001* respectively.

- **Random Forest** Three Models of Random Forest have been implemented using the Scikit-Learn library *RandomForestClassifier* with number of trees set as *10*, *100*, and *200* respectively.

- **Bootstrap Aggregating:** Four Models of Bagging have been implemented using the Scikit-Learn library *BaggingClassifier* with base_estimator set as *Logistic Regression*, *Neural Network*, *Support Vector Machine*, and *Random Forest* respectively. The bagging algorithm takes at maximum 80% of features and dataset as input.

- **Boosting:** Two Models of Boosting have been implemented using the Scikit-Learn library *AdaBoostClassifier* with base_estimator set as *Logistic Regression* and *Random Forest* respectively.

- **Stacking:** Three Models of Stacking have been implemented using the library *Mlxtend* with classifier set as *Neural Network*, *Support Vector Machine*, and *Random Forest*. Meta_Classifier is set as *Logistic Regression*. The models don't use probability, uses average probability, and uses stacked probabilities respectively. Sample code of the three models of stacking are shown below.

```
sclf1 = StackingClassifier(classifiers=[clf_SVM, clf_NN, clf_RF],
meta_classifier=clf_LR)

sclf2 = StackingClassifier(classifiers=[clf_SVM, clf_NN, clf_RF],
use_probas=True, average_probas=True, meta_classifier=clf_LR)

sclf3 = StackingClassifier(classifiers=[clf_SVM, clf_NN, clf_RF],
use_probas=True, average_probas=False, meta_classifier=clf_LR)
```

The results of different classifiers were combined using majority voting. Accuracy on both the datasets and confusion matrix were observed for each classifier.

6 Observations

The accuracy of all models on the MNIST testing dataset and USPS dataset is analyzed with the help of gr tables. The observation is listed below model wise for both the datasets.

6.1 Logistic Regression

The accuracy obtained by performing Logistic (Softmax) Regression on MNIST and USPS dataset were observed and compared to other models. The Accuracy obtained from different models of Logistic Regression are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Mini-batch SGD Logistic Regression	87.07	34.10
Scikit-Learn (multi_class='ovr')	92.06	**34.99**
Scikit-Learn (multi_class='multinomial')	**92.54**	34.51

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Scikit-Learn library with the multi_class parameter set as *multinomial*. However, the best performance on USPS dataset was obtained by the model with the multi_class parameter set as 'one versus rest'. The confusion matrices for these models are shown in Figure 5. From the figure, it can be observed that the model mostly performs quite accurately for each digit of MNIST dataset. However, it fails in some cases where it wrongly classifies some digits which look similar. An example can be seen for digit 5 getting misclassified as 3 and digit 4 and 9 getting misclassified. In USPS dataset, we mostly observe incorrect classification except for digit 5 where the model performs satisfactorily. This can be explained using the "No Free Lunch Theorem".

```
[ 947    0    3    2    0    3   16    1    8    0][ 627    2  151  137  217  264   70  187   80  265]
[   0 1096    5    3    1    2    4    0   24    0][  62  550   41  209  501  135   25  355   96   26]
[  16   22  845   26   19    0   28   22   47    7][  93   50 1198  146   77  211   73   81   41   29]
[   5    3   22  881    1   29    8   20   27   14][  43   10  301  897   19  581    9   58   51   31]
[   3    9    5    0  857    1   17    2   11   77][  46   42   47   77  880  146   21  476  179   86]
[  26   15    6   82   24  643   29    9   41   17][  74   15   98  199   48 1375   70   75   32   14]
[  19    5   13    2   13   18  882    0    6    0][  90    8  513  117  102  531  601   14   14   10]
[   4   39   26    1   13    0    4  889   10   42][ 100   83  191  545  119  214   17  573  114   44]
[   9   17   14   39   12   22   18   14  809   20][ 192   20  107  508  135  580  116  111  181   50]
[  14   13   11   12   52   10    1   27   11  858][  36   40   95  526  133  109   11  642  292  116]
```

Figure 5: Confusion Matrices for LR of MNIST and USPS dataset respectively

6.2 Neural Network

The accuracy obtained by performing classification using Neural Network on MNIST dataset were not great when compared to other models. The Accuracy obtained from different models of Ne are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Deep Neural Network	94.06	39.42
Scikit-Learn (solver='adam')	93.4	37.51
Scikit-Learn (solver='sgd')	84.33	31.81
Convolutional Neural Network	**98.91**	**70.8**

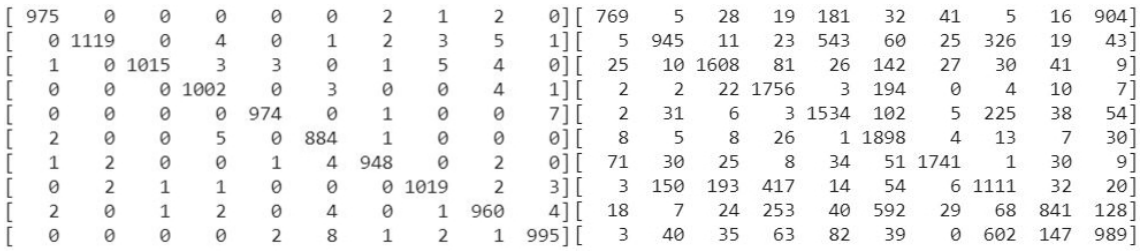

Figure 6: Confusion Matrices for CNN of MNIST and USPS dataset respectively

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Convolutional Neural Network Architecture. Similarly, the best performance on USPS dataset was obtained by the same model. We observe that Convolutional Neural network gives the highest accuracy for both datasets among all the classification models implemented. The confusion matrices for these models are shown in Figure 6. From the figure, it can be observed that the model performs very accurately for each digit of MNIST dataset. In USPS dataset, we observe satisfactory results except for digit 1,8, and 9 which gets misclassified as 4,5, and 7.

6.3 Support Vector Machine

The accuracy obtained by performing classification using Support vector Machine on MNIST dataset were observed and compared to other models. The Accuracy obtained from different models of Support Vector Machine are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Scikit-Learn (kernel='linear')	93.9	33.97
Scikit-Learn (kernel='rbf')	**94.35**	**40.85**
Scikit-Learn (gamma='0.001')	94.01	40.79

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Scikit-Learn library with kernel parameter set as 'radial basis function'. Similarly, the best performance on USPS dataset was obtained by the same model. The confusion matrix for these models are shown in Figure 7. From the figure, it can be observed that the model performs quite accurately for each digit of MNIST dataset. In USPS dataset, we mostly observe wrong classification except for digit 2 and 5 where the model performs satisfactorily. This can be explained using the "No Free Lunch Theorem".

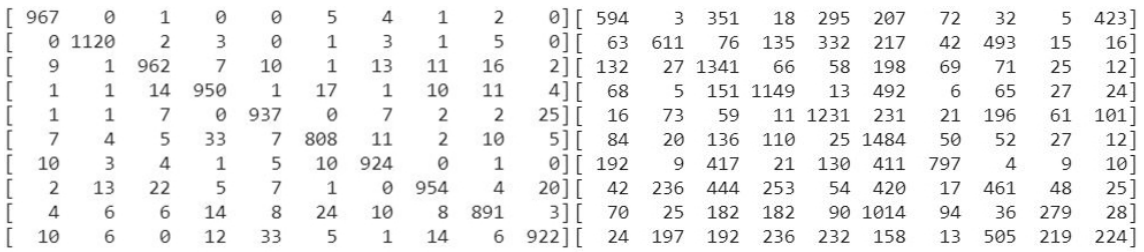

Figure 7: Confusion Matrices for SVM of MNIST and USPS dataset respectively

6.4 Random Forest

The accuracy obtained by performing calssification using Random Forest on MNIST dataset were observed and compared to other models. The Accuracy obtained from different models of Random Forest are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Scikit-Learn (n_estimator='10')	94.45	37.18
Scikit-Learn (n_estimator='100')	96.86	44.84
Scikit-Learn (n_estimator='200')	**96.87**	**45.34**

```
[ 969    0    0    0    0    1    4    1    4    1][ 671    6  258   67  440   90   56   40    4  368]
[   0 1121    3    4    1    2    2    0    1    1][   9  676   54  125   38  123   36  932    6    1]
[   5    0 1000    6    3    0    2    9    7    0][ 100   21 1413   63   53  143   15  182    6    3]
[   0    0    9  976    0    6    0    8    8    3][  42    3   78 1418   46  261    2  126    5   19]
[   1    0    0    0  955    0    5    0    3   18][   7  154   46   26 1209  142   15  312   57   32]
[   4    0    1   16    3  852    6    3    6    1][ 106   16   85   70   11 1625   12   61    5    9]
[   6    3    0    1    3    5  937    0    3    0][ 311   28  216   30  101  260  991   44   10    9]
[   1    4   19    1    1    0    0  989    1   12][  48  230  528  275   30  233   37  604    4   11]
[   5    0    5    7    7    5    3    4  928   10][  66   22  182  226   97  965   68   61  274   39]
[   5    5    1   15   10    2    1    4    6  960][  23  186  310  344  235   87   14  473  141  187]
```

Figure 8: Confusion Matrices for RF of MNIST and USPS dataset respectively

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Scikit-Learn library with the number of trees set as '200'. Similarly, the best performance on USPS dataset was obtained by the same model. The confusion matrices for these models are shown in Figure 8. From the figure, it can be observed that the model performs quite accurately for each digit of MNIST dataset except 9 where it misclassifies some data as 4. In USPS dataset, we mostly observe incorrect classification except for digit 2,4, and 5 where the model performs satisfactorily.

6.5 Bootstrap Aggregating

The accuracy was obtained by performing bagging on different classifiers and then testing on MNIST and USPS dataset. The results were observed and compared to other models. The Accuracy obtained from different bagging models are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Scikit-Learn (base_estimator='LR')	92.45	34.69
Scikit-Learn (base_estimator='NN')	**96.72**	**46.23**
Scikit-Learn (base_estimator='SVM')	96.11	42.27
Scikit-Learn (base_estimator='RF')	94.22	35.87

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Scikit-Learn library with base classifier set as 'Neural Network'. Similarly, the best performance on USPS dataset was obtained by the same model. The confusion matrices for these models are shown in Figure 9. From the figure, it can be observed that the bagging model performs very accurately for each digit of MNIST dataset. In USPS dataset, we mostly observe incorrect classification except for digit 2 and 3 where the model performs satisfactorily.

```
[ 967    0    0    1    3    1    3    2    2    1][ 563   17  102   56  197   27   48   27  367  596]
[   0 1121    2    0    1    3    1    1    6    0][  28  541  303  123  204   25   18  233  232  293]
[   5    4  995    3    8    0    0   12    2    3][  33   56 1405  176   59   19    1   40  157   53]
[   0    0    4  974    0    1    0    6   24    1][  21   13  174 1446    3   64    2   33  185   59]
[   1    0    1    0  957    0    1    0    7   15][  28   69   32   23 1026   22   26  140  284  350]
[   1    2    0    3    1  846    5    0   26    8][  48   22   34   94   20 1282   23   17  323  137]
[   6    8    0    1    6    1  930    0    6    0][ 169  117   61   81  222   68 1023    7  199   53]
[   0    4   11    5    0    0    0  984    6   18][  19   85  528  433   35    6    2  546  241  105]
[   3    5    2   10    5    5    0    2  932   10][ 188   39   80  274  107  110   51   41  963  147]
[   2    2    0    5    7    0    0    3   24  966][  16   77  156  260   77   13    6  463  481  451]
```

Figure 9: Confusion Matrices for Bagging of MNIST and USPS dataset respectively

6.6 Boosting

The accuracy was obtained by performing boosting on different classifiers and then testing on MNIST and USPS dataset. The results were observed and compared to other models. The Accuracy obtained from different boosting are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Scikit-Learn (base_estimator='LR')	85.31	32.06
Scikit-Learn (base_estimator='RF')	**95.59**	**39.66**

```
[ 964    0    1    1    1    3    5    2    3    0][ 519    7  326   50  447  126   93   58   53  321]
[   0 1119    3    2    1    1    3    0    6    0][  22  606  153  111  321  120   35  568   47   17]
[   7    2  981    6    3    2    2    8   18    3][  95   37 1262   92   71  159   39  155   73   16]
[   0    0   15  961    0   10    2    8   10    4][  50    8  154 1158   39  365   18   85   79   44]
[   1    2    4    0  936    0    9    0    6   24][  11  128   77   35 1150  144   25  231  130   69]
[   3    1    3   27    2  833   10    1    9    3][  87   30  123  151   37 1345   22  100   81   24]
[   5    3    1    1    3    6  933    0    5    1][ 242   19  260   70  174  297  814   38   67   19]
[   2    8   18    3    4    1    0  967    5   20][  74  284  419  231  131  211   32  472  125   21]
[   2    1    9   10    3   10    5    5  919   10][  64   28  202  250  144  716   68   73  407   48]
[   5    4    3   12   17    2    2    7   11  946][  24  224  186  212  354  119    9  446  226  200]
```

Figure 10: Confusion Matrices for Boosting of MNIST and USPS dataset respectively

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Scikit-Learn library with base classifier set as 'Random Forest'. Similarly, the best performance on USPS dataset was obtained by the same model. The confusion matrices for these models are shown in Figure 10. From the figure, it can be observed that the boosting model performs very accurately for each digit of MNIST dataset. In USPS dataset, we mostly observe incorrect classification. This can be explained from the fact that over-fitting can occur while carrying out boosting of a classifier and the classifier performance might decrease on another dataset. This can be the reason that such low accuracy on USPS data was seen compared to other ensemble techniques even though the classifier performs well on MNIST dataset.

6.7 Stacking (Combining Classifiers)

The accuracy was obtained by performing Stacking different classifiers and then testing on MNIST and USPS dataset. The results were observed and compared to other models. The Accuracy obtained from different Stacking models are shown in the table given below.

Models	Accuracy (MNIST)	Accuracy (USPS)
Mlxtend (probability='False')	90.85	26.69
Mlxtend (probability='Stacked')	**96.47**	**42.95**
Mlxtend (probability='Average')	96.46	41.25

As it can be seen in the table, the best performance on MNIST dataset was obtained by the model constructed using the Mlxtend library in which the probability of each classifier is stacked and not averaged. Similarly, the best performance on USPS dataset was obtained by the same model. The confusion matrices for these models are shown in Figure 11. From the figure, it can be observed that the stacking model performs very accurately for each digit of MNIST dataset. In USPS dataset, we mostly observe incorrect classification.

```
[ 971    0    1    0    0    2    1    2    3    0][ 528    4  144  307   89   34   67  266  132  429]
[   0 1124    2    1    0    1    2    0    5    0][  20  478   55  205   13   55   29  717  200  228]
[   7    1  991    6    1    5    2    6   13    0][ 145   38 1246  158    7   99   26   80  189   11]
[   3    0    7  980    0    2    0    8    9    1][ 133   11  239 1306    1  139    4   52  103   12]
[   0    0    1    1  947    0    7    3    3   20][  10   56   21   79  829   20  215  244  131  395]
[   2    3   22   10    0  833    5    2   13    2][  62   37  262   66    5 1451   17   28   65    7]
[   6    3    1    1    4    7  929    0    7    0][ 114  115  196   69   53  157  995   27  234   40]
[   0    2    7    7    0    0    0  993    5   14][ 113   33  314  580    3   51   10  589  237   70]
[   4    0    5    9    3    7   11    5  923    7][ 114   31  273  295   11  288  140   89  698   61]
[   3    5    1   15   10    0    1   13    5  956][  15   47   68  324  104   15   38  573  346  470]
```

Figure 11: Confusion Matrices for Stacking of MNIST and USPS dataset respectively

6.8 Majority Voting (Combining Classifiers)

Once the analysis of each classifier was done, the predicted values of each model are put in an algorithm to get the most voted label for each data. The accuracy of **96.51%** was obtained for MNIST dataset and **44.69%** for USPS dataset. The confusion matrix obtained after carrying out the majority voting analysis is shown in Figure 12. From the figure, it can be observed that after majority voting the predictions are very accurate for each digit of MNIST dataset. However, some misclassifications are observed such as for digit 4 and 5 as most of the classifier classified 4 as 9,

```
[ 970    0    1    1    0    2    4    1    1    0][ 628    1  289   48  306  154   40   97   22  415]
[   0 1123    2    2    0    1    3    1    3    0][  28  622   53  179  169  172   25  703   42    7]
[   6    2  994    4    3    1    7    8    6    1][ 104   30 1453   73   46  164   29   75   17    8]
[   0    0   12  971    0    9    0   11    7    0][  45    4  154 1326    8  372    2   56   26    7]
[   1    1    4    0  955    0    6    0    2   13][  11   91   42   19 1210  136   11  305  101   74]
[   9    3    1   24    3  829    7    1   12    3][  66   14  121  133   18 1540   20   58   25    5]
[   8    3    3    1    5    6  930    0    2    0][ 193   18  429   34  104  335  870    8    4    5]
[   2    7   20    3    3    0    0  982    1   10][  44  165  318  454   24  288   10  591   91   15]
[   5    3    5   15    8   18    8    6  902    4][  93   15  167  281   83  893   67   69  309   23]
[   8    6    1   14   15    3    1   13    4  944][   9  142  153  366  146  101    6  625  262  190]
```

Figure 12: Confusion Matrices for Majority voting of MNIST and USPS dataset respectively

etc. In USPS dataset, we mostly observe incorrect classification except for digit 2,3, and 5 where the results obtained are satisfactory because many classifiers performed well for these digits.

7 Conclusion

In this paper, various Machine Learning Classification methods are proposed to solve the handwritten digit classification problem. Hyper-parameters are tuned for each model and the results obtained was observed. The best results for each model is shown in the Figure 13.

- **Which classifier has the overall best performance?:**
  The best accuracy of 98.91% and 70.8% was achieved by using Convolutional Neural Network for classification on both MNIST and USPS dataset respectively.

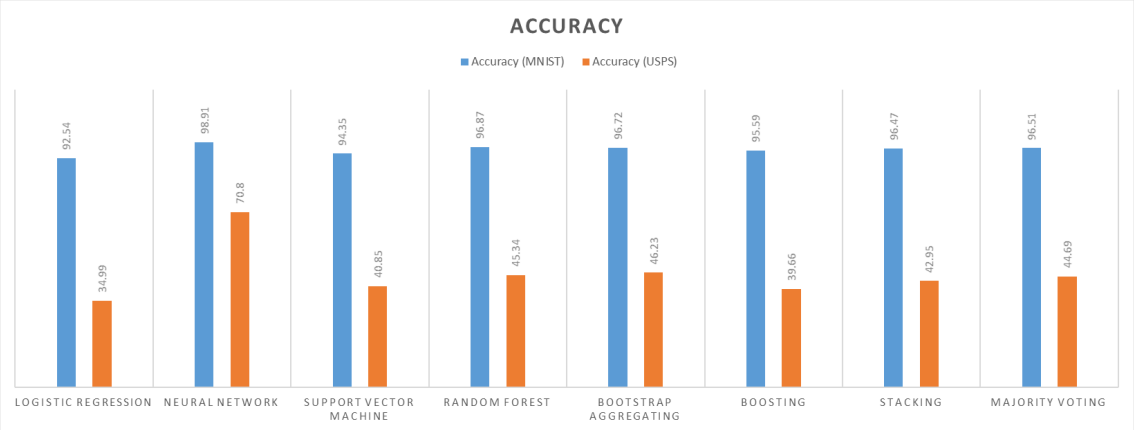

Figure 13: Comparison between different Classification Techniques

- **Does the results obtained support the "No Free Lunch" theorem?:**
  Hume (1739–1740) pointed out that 'even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience'. This can be explained as following in the context of this project: No model is universal. Though every classification model after training on MNIST dataset gives good performance on its testing data, it can't be expected to perform similarly for a different dataset even when the dataset is similar for example, in this case, handwritten digits. If a model is trained on one dataset then it should only be expected to run properly on that dataset and not any other dataset. This is why we see that even though both datasets contain handwritten digit images, the models perform properly only on one it's trained on or 'had experience with'.

- **Is the overall combined performance better than that of any individual classifier?:**
  The Accuracy obtained by implementing Majority Voting on MNIST dataset is higher than all models except Convolutional Neural Network, Random Forest with 100 and 200 decision trees, and Bagging model with Neural Network as the base classifier. Similar results were obtained for USPS dataset.

References

[1] https://medium.com/fuzz/machine-learning-classification-models-3040f71e2529

[2] https://medium.com/@martinpella/logistic-regression-from-scratch-in-python-124c5636b8ac

[3] https://en.wikipedia.org/wiki/Artificial_neural_network

[4] https://en.wikipedia.org/wiki/Support_vector_machine

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.-RandomForestClassifier.html

[6] https://en.wikipedia.org/wiki/Ensemble_learning

[7] https://en.wikipedia.org/wiki/Bootstrap_aggregating

[8] https://en.wikipedia.org/wiki/Boosting_(machine_learning)

[9] http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/