A Report on Handwritten Digit Recognition Using Classification Techniques

Miki Padhiary
UBID: mikipadh Person Number: 50286289

November 22, 2018

Abstract

In this report, We studied how to apply machine learning to solve the classification task. Our objective was to recognize a 2828 grayscale handwritten digit image and identify it as a digit among 0, 1, 2, ... , 9. Classifiers such as Multiclass Logistic Regression, Neural Networks, Support Vector Machine and Random Forest are used for our purpose.The feature vectors are obtained from MNIST dataset and testing is done on both MNIST and USPS dataset. Experiments on various hyper-parameters were performed for their eect on the accuracy of the program. The results of various classifiers were observed using Confusion Matrices. Majority Voting ensembling method is carried out for all he four classifiers. The best accuracy was achieved using ***convolutional neural networks*** *and* ***majority voting ensembler****.*

Keywords: MNIST, USPS, Softmax Regression, Neural Network, Convolution Neural Network, Support Vector Machine, Random Forest, Ensembler, Majority Voting

1 Introduction

For the experiment, features and label were obtained from MNIST dataset. The testing was carried out on the following datasets:

1. MNIST test Dataset
2. USPS Dataset

The MNIST database was constructed from NIST's Special Database. It is a large database of handwritten digits that is commonly used for training various image processing systems. This database contains 60,000 training images and 10,000 testing images. Furthermore, the MNIST dataset is normalized to fit into a 28x28 pixel bounding box.
The USPS is another testing data for this project and is used to test whether the models can be used to generalize a new population of data. USPS images are segmented images scanned at a resolution of 100 ppi and cropped.
The following four classifiers were implemented:

1. **Multiclass Logistic Regression**
2. **Neural Networks**
3. **Support Vector Machine and**
4. **Random Forest**

A number of different models by changing hyperparameters are also modeled and combined using **Bootstrap Aggregating(Majority Voting)**.

2 Steps Performed

Classication is a data mining function that assigns items in a collection to target categories or classes. The goal of classication is to accurately predict the target class for each case in the data.

1. Extract features and target from the MNIST dataset.
2. Read USPS dataset and resize the images to 28*28 pixels.
3. Patch the MNIST training dataset into Softmax Regression and test on MNIST and USPS test dataset.
4. Patch the MNIST training dataset into Neural Network and test on MNIST and USPS test dataset.
5. Patch the MNIST training dataset into SVM and test on MNIST and USPS test dataset.
6. Patch the MNIST training dataset into Random Forest and test on MNIST and USPS test dataset.
7. Perform ensembling using majority voting.

3 Multiclass Logistic Regression

Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes. The whole model, including the activation function can be written as:

$$\hat{y} = softmax(xW + b)$$

This model is sometimes called multiclass logistic regression. Other common names for it include **softmax regression and multinomial regression**. The model actually performs two tasks:
First: It transforms the input vector x of size input_features_number to the new vector z of size number_of_classes.

Second: It applies softmax over the output vector z to make its components look like probabilities.
The softmax operation computes the following:

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

If we use 1-of-K coding scheme, $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_k]$, our problem can be written as:

$$p(C_K|X) = y_K(x) = \frac{exp(a_k)}{\sum_j exp(a_j)} \tag{2}$$

where activation $\mathbf{a_k} = \mathbf{W^T}X + b_K$.The cross entropy function is given as

$$E(x) = -\sum_{k=1}^{K} t_k ln y_k \tag{3}$$

where $\mathbf{y_k} = y_k(\mathbf{x})$. The gradient of error function is,

$$\nabla_{wj} E(x) = (y_j) - (t_j)(x) \tag{4}$$

The weight is calculated using the following:

$$\boldsymbol{w}_j^{(t+1)} = \boldsymbol{w}_j^{(t)} - \eta \nabla \boldsymbol{w_j} E(x) \tag{5}$$

where η is the learning rate.
Following softmax regression models with hyperparameter changes were implemented:

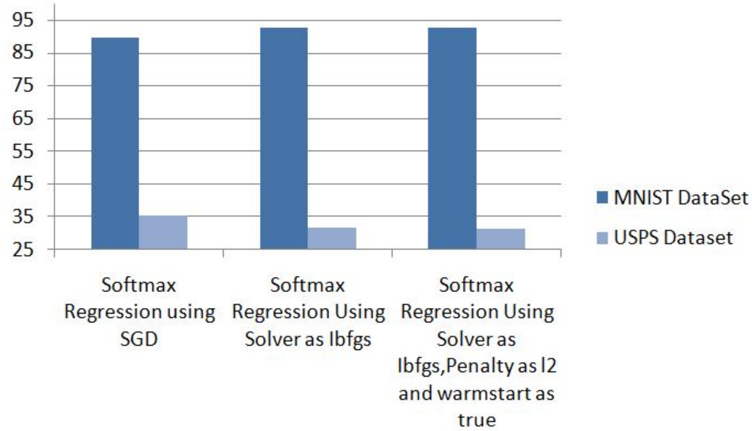
1. Softmax Regression using SGD
2. Softmax Regression Using Solver as Ibfgs
3. Softmax Regression Using Solver as Ibfgs,Penalty as l2 and warm_start as true

3.1 Analysis

The first model runs 256 epochs on the data with batch size of 512. The other two models have been implemented by using the Scikit-Learn library Logistic Regression.The second model has solver set as lbfgs and multi_class set as multinomial. The third model has solver set as lbfgs and multi_class set as multinomial, Penalty as ’l2’ and warm_start as ’true’.

3.2 Results Based on Accuracy

The reason for choosing mini-batch SGD is with same computing time, it updates the weights much more often than batch gradient descent, and therefore has faster converging speed. For this reason, minibatch SGD performs better on USPS dataset.Various accuracy for different models is shown in graph as below:



3.3 Results Based on Confusion Matrix

For MNIST dataset, **Softmax Regression Using Solver as Ibfgs and multi_class as multinomial** performed well and for USPS dataset, **Softmax Regression using mini batch SGD** performed better. The confusion matrix for both the models are depicted below:

```
[[ 958    0    0    3    1    8    5    4    1    0]
 [   0 1111    4    2    0    2    3    2   11    0]
 [   5    7  932   15   10    3   14    7   35    4]
 [   4    1   18  917    1   22    4   11   24    8]
 [   1    1    7    3  917    0   10    4    8   31]
 [  11    2    2   34   11  780   14    5   29    4]
 [   9    3    8    2    8   14  911    2    1    0]
 [   1    7   26    4    7    1    0  950    3   29]
 [   9   12    8   22    7   25   12    6  859   14]
 [  10    8    1    9   23    6    0   19    7  926]]
```

```
[[ 390    1  200  149   70  301   31  595   84  179]
 [  37  264  400  134  266  239   18  499  116   27]
 [  46   22 1253  114   23  369   68   41   41   22]
 [  29    6  281  861   10  671    8   74   42   18]
 [  29    9   84   40  660  184   20  727  143  104]
 [  45    5  364  210   19 1191   24   86   44   12]
 [  78    5  728   82   43  427  558   24    9   46]
 [  74   45  108  536   42  188    8  789  161   49]
 [ 172    7  191  501   63  565   76  182  196   47]
 [  15   10  105  501   70   87    6  825  190  191]]
```

Figure 1: Confusion matrix on MNIST Dataset(left) and USPS Dataset(right) for Softmax Regression

It is observed that the model mostly predicts accurately for each digit of MNIST dataset. It fails in some cases where it wrongly classifies some digits which look similar. In USPS dataset, the model mostly classifies incorrectly.

1. **Strengths**
   - For MNIST, the best classification was for digit 2 and then for 1.
   - For USPS, the best classification was for digit 3 and then for 6.
2. **Weakness**
   - For MNIST, 3 and 5 were misclassifed as 8 and 9 for 35 and 31 times respectively.
   - For USPS, 9 and 6 were misclassified as 7 and 2 for 825 and 728 times respectively.

4 Neural Networks

Neural Networks are machine learning framework that attempts to mimic the learning pattern of natural biological neural networks. Neural Network is created by adding layers of perceptrons together, creating a multi-layer perceptron model of a neural network. It has an input layer which directly takes the data and an output layer which creates the resulting outputs. Any layers in between are known as hidden layers because they dont directly see the feature inputs within the data feeded in or the outputs. A simple architecture is described below:

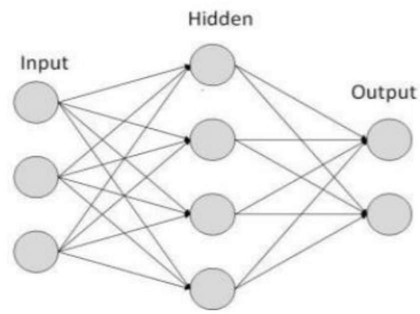


Figure 1: Neural Network architecture

Convolutional neural network is a deep learning neural network used for image classification. It is associated with the idea of a moving filter which passes through the image. Following neural network models with hyperparameters were implemented:

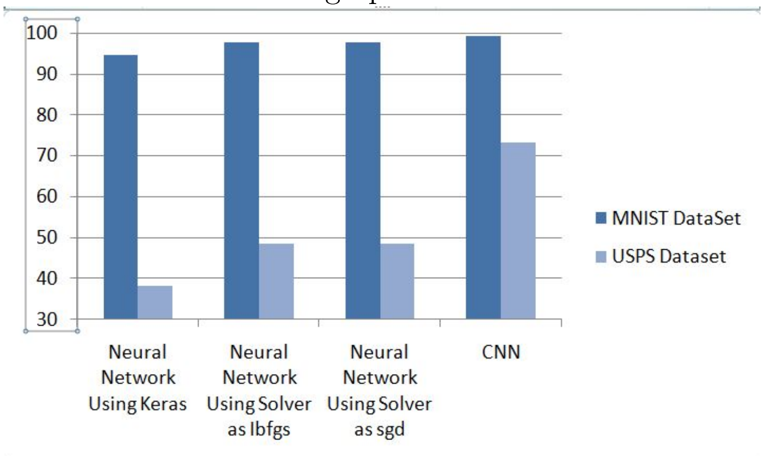
1. Neural Network Using Keras
2. Neural Network Using Solver as 'Ibfgs', alpha as 1e-5, hidden_layer_sizes as '256', random_state as '1'
3. Neural Network Using Solver as 'sgd' ,alpha as 1e-5,hidden_layer_sizes as '200', random_state as '1', max_iter as '50'
4. Convolution Neural Network

4.1 Analysis

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 MLP-Classifier with solver set as sgd and lbfgs respectively. The last Model is a Convolutional Neural Network architecture created from scratch.

4.2 Results Based on Accuracy

Convolutional neural network with **batch_size =128 and epochs = 10** outperformed the accuracy in comparison to other models for both MNIST and USPS dataset. Various accuracy for different models is shown in graph as below:



4.3 Results Based on Confusion Matrix

For both the dataset, **Convolutional Neural Network** performed better. Neural Network using **Ibfgs and sgd** predicted the same accuracy. The confusion matrix for **Solver as Ibfgs** for both the dataset is depicted below:

```
[[ 971    0    1    0    0    1    3    1    3    0]
 [   0 1124    2    3    0    1    3    0    1    1]
 [   6    0  999    7    2    0    4    9    5    0]
 [   0    0   11  972    0    5    0    9   10    3]
 [   1    0    0    0  960    0    5    0    3   13]
 [   3    0    1   12    4  859    4    2    4    3]
 [   8    3    0    0    2    1  941    0    3    0]
 [   1    3   22    0    0    0    0  991    2    9]
 [   5    0    4    7    5    6    3    3  930   11]
 [   4    5    2    8   11    3    1    4    8  963]]
```

```
[[ 619    9  277   52  461  164   74   95    2  247]
 [  30  570  124  119   62   91   22  964   17    1]
 [  78   28 1277   79   66  209   14  243    4    1]
 [  34    9   87 1293   58  316    5  175    4   19]
 [  11  209   49   25 1090  166   21  380   22   27]
 [ 117   27  136   79   25 1478   20  102   10    6]
 [ 281   47  239   21   98  378  797  121    5   13]
 [  33  323  396  252   37  241   29  678    6    5]
 [  46   49  151  215  112 1087   59   95  161   25]
 [  17  279  255  299  250  134    8  571   85  102]]
```

Figure 2: Confusion matrix on MNIST Dataset(left) and USPS Dataset(right) for Neural Network

The confusion matrix for CNN is as below:

```
[[ 974    0    2    0    0    1    1    0    1    2]
 [   1 1131    3    0    0    0    2    1    0    0]
 [   1    0 1016    1    0    0    0    2    0    0]
 [   0    1    2 1007    0    7    0    0    0    0]
 [   0    0    0    0  979    0    1    1    0   10]
 [   0    0    0    1    0  878    1    0    0    2]
 [   2    1    4    0    1    4  951    0    2    0]
 [   1    2    3    0    0    0    0 1018    0    0]
 [   1    0    2    1    0    1    2    1  970    0]
 [   0    0    0    0    2    1    0    5    1  995]]
```

```
[[1009   25   11    1    1    6   66    6   32    2]
 [   4 1094    1    2   53    4   17  154   13   71]
 [  24   96 1774   20   14   29   53  200   31   30]
 [  11   56  115 1863    3   27    3  347   49   69]
 [ 180  480    9    4 1711   17   32   50   74  230]
 [   2    5   28   96   25 1835    9   37  253   19]
 [  36   51   12    2    3    3 1775    2   40    1]
 [   3  156   20    2   56   24    0 1092   13  232]
 [  15   23   23    5   97    9   41   74 1372  203]
 [ 716   14    6    5   37   46    4   38  123 1143]]
```

Figure 3: Confusion matrix on MNIST Dataset(left) and USPS Dataset(right) for CNN

The model predicted quite well for digits of MNIST dataset. However, for some of the digits it failed to classify where the images were similar. For USPS the accuracy is lower but is still better than the prediction from Softmax Regression.

1. **Strengths**
   - For MNIST, using CNN the classification was accurate for all digits.
   - For USPS, the best classification was for digit 4 and then for 6.
2. **Weakness**
   - For USPS, 4 and 3 were incorrectly classified as 1 and 7 for 480 and 347 times.

5 Support Vector Machine

SVM are supervised learning algorithms that can be used for both regression and classification tasks. The objective of the SVM algorithm is to find a hyperplane that has maximum distance between data points of both classes. 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. Following SVM models with various hyperparameters were implemented:

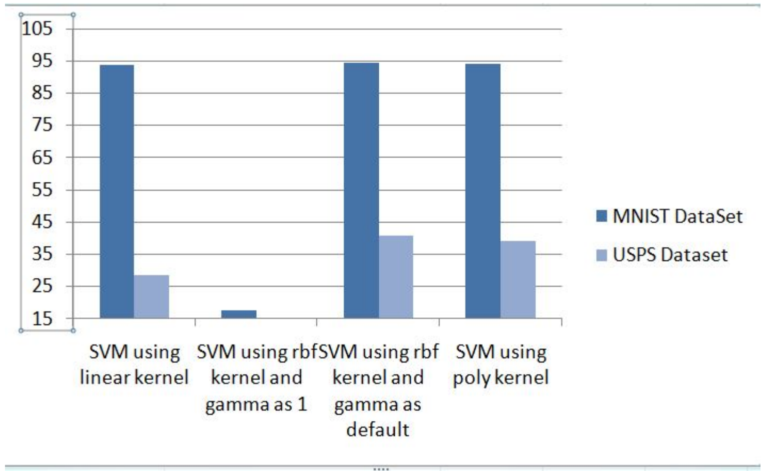
1. SVM using kernel as 'linear'
2. SVM using kernel as 'rbf', gamma as default and max_iter as '1000'
3. SVM using kernel as 'rbf', gamma as '1' and max_iter as '1000'
4. SVM using kernel as 'poly'

5.1 Analysis

Four Models of Support Vector Machine have been implemented using the Scikit-Learn library SVC with kernel set as linear,rbf with default 'gamma' and 'gamma as 1'and 'poly' respectively.

5.2 Results Based on Accuracy

SVM using rbf kernel and gamma as default outperformed the accuracy in comparison to other models for both MNIST and USPS dataset. Various accuracy for different models is shown in graph as below:



5.3 Results Based on Confusion Matrix

It is observed that the model mostly predicts accurately for each digit of MNIST dataset. In USPS dataset, almost all the digits were misclassified with either 2 or 5.

967	0	1	0	0	5	4	1	2	0
0	1120	2	3	0	1	3	1	5	0
9	1	962	7	10	1	13	11	16	2
1	1	14	950	1	17	1	10	11	4
1	1	7	0	937	0	7	2	2	25
7	4	5	33	7	808	11	2	10	5
10	3	4	1	5	10	924	0	1	0
2	13	22	5	7	1	0	954	4	20
4	6	6	14	8	24	10	8	891	3
10	6	0	12	33	5	1	14	6	922

594	3	351	18	295	207	72	32	5	423
63	611	76	135	332	217	42	493	15	16
132	27	1341	66	58	198	69	71	25	12
68	5	151	1149	13	492	6	65	27	24
16	73	59	11	1231	231	21	196	61	101
84	20	136	110	25	1484	50	52	27	12
192	9	417	21	130	411	797	4	9	10
42	236	444	253	54	420	17	461	48	25
70	25	182	182	90	1014	94	36	279	28
24	197	192	236	232	158	13	505	219	224

Figure 4: Confusion matrix on MNIST Dataset(left) and USPS Dataset(right) for SVM

1. **Strengths**
   - For MNIST, the best classification was for digit 0 and then for 1.
   - For USPS, the best classification was for digit 4 and then for 5.
2. **Weakness**
   - For USPS, almost all the digits were misclassified except for 2 and 5.

6 Random Forest Classifier

Random forests are used to build predictive models for both classification and regression problems. In the case of a random forest, the model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer. A simplified version of

Random Forest Classifiers is depicted below:

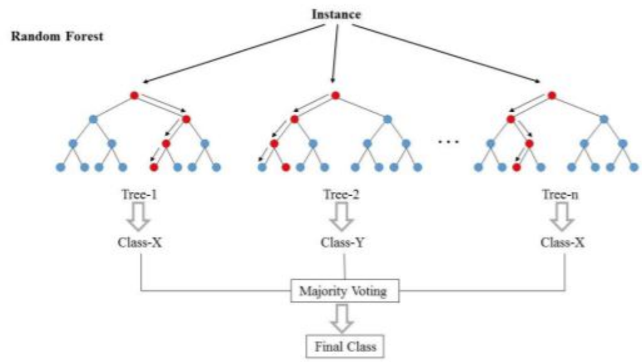


Figure 2: Random Forest Classifier

Following Random Forest classifiers with changes in hyperparameters were implemented:

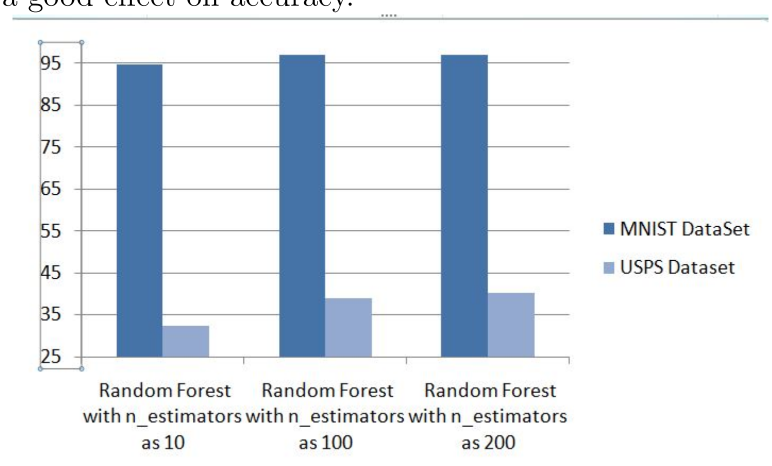
1. Random Forest with n_estimators set to 10
2. Random Forest with n_estimators set to 100 and criterion as 'entropy'
3. Random Forest with n_estimators set to 200

6.1 Analysis

All the models were implemented by using the Scikit-Learn ensembler **RandomForestClassifier**. The models have n_estimators set to 10, 100 and 200 respectively.

6.2 Results Based on Accuracy

The best accuracy was obtained when the value of n_estimators was set to 200. Below is a graph showing the accuracy obtained for the used models. It is observed that increasing n_estimators has a good effect on accuracy.



6.3 Results Based on Confusion Matrix

It is observed that the model mostly predicts accurately for each digit of MNIST dataset. In USPS dataset, almost all the digits were misclassified with either 2 or 5.

```
[[ 971    0    1    0    0    1    3    1    3    0]
 [   0 1124    2    3    0    1    3    0    1    1]
 [   6    0  999    7    2    0    4    9    5    0]
 [   0    0   11  972    0    5    0    9   10    3]
 [   1    0    0    0  960    0    5    0    3   13]
 [   3    0    1   12    4  859    4    2    4    3]
 [   8    3    0    0    2    1  941    0    3    0]
 [   1    3   22    0    0    0    0  991    2    9]
 [   5    0    4    7    5    6    3    3  930   11]
 [   4    5    2    8   11    3    1    4    8  963]]
```

```
[[ 619    9  277   52  461  164   74   95    2  247]
 [  30  570  124  119   62   91   22  964   17    1]
 [  78   28 1277   79   66  209   14  243    4    1]
 [  34    9   87 1293   58  316    5  175    4   19]
 [  11  209   49   25 1090  166   21  380   22   27]
 [ 117   27  136   79   25 1478   20  102   10    6]
 [ 281   47  239   21   98  378  797  121    5   13]
 [  33  323  396  252   37  241   29  678    6    5]
 [  46   49  151  215  112 1087   59   95  161   25]
 [  17  279  255  299  250  134    8  571   85  102]]
```

Figure 5: Confusion matrix on MNIST Dataset(left) and USPS Dataset(right) for Random Forest

1. **Strengths**
    - For MNIST, the best classification was for digit 1 and then for 2.
    - For USPS, the best classification was for digit 3 and then for 5.
2. **Weakness**
    - For USPS dataset, most of the digits were misclassified except for 2, 3 and 5.

7 Ensembling Methods

The main causes of error in learning are due to noise, bias and variance. Ensemble helps to minimize these factors. These methods are designed to improve the stability and the accuracy of Machine Learning algorithms. Combinations of multiple classifiers decrease variance, especially in the case of unstable classifiers, and produce a more reliable classification than a single classifier.

7.1 Bootstrap Aggregating

Every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes. If none of the predictions get more than half of the votes,then the ensemble method could not make a stable prediction for this instance. Majority Voting is a kind of Bagging techniques. Bagging decreases the models variance.

7.2 Boosting

In Boosting algorithms each classifier is trained on data, taking into account the previous classifiers success. After each training step, the weights are redistributed. Misclassified data increases its weights to emphasise the most difficult cases. In this way, subsequent learners learn while training. Boosting decreases the model's bias.

7.3 Stacking

Stacking is another ensemble model, where a new model is trained from the combined predictions of two (or more) previous model. The predictions from the models are used as inputs for each sequential layer, and combined to form a new set of predictions.

7.4 Analysis

Different bagging techniques were used in addition to **Majority Voting**. The following table depicts the accuracy obtained for different bagging models:

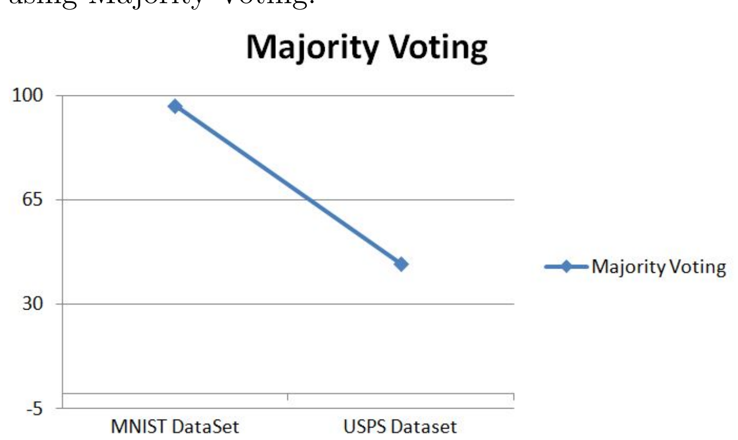
	Bagging	
Models	**Accuracy - MNIST**	**Accuracy - USPS**
base_estimator='LR'	89.76	33.96
base_estimator='NN'	96.72	46.23
base_estimator='SVM'	96.11	42.27
base_estimator='RF'	93.11	35.87

Below is the table which shows the accuracy obtained for different boosting models:

	Boosting	
Models	**Accuracy - MNIST**	**Accuracy - USPS**
base_estimator='LR'	84.31	32.06
base_estimator='RF'	96.46	39.66

7.5 Results Based on Accuracy

Majority voting was implemented from scratch. Using Majority Voting, the best accuracy was obtained for MNIST dataset. Below is a graph showing the accuracy obtained MNIST and USPS dataset using Majority Voting.



From above tables, we have also observed that:

1. The best performance for bagging on MNIST dataset is obtained by the model constructed using the Scikit-Learn library with base classifier set as Neural Network. Similarly, the best performance on USPS dataset is obtained by the same model.

2. The best performance for boosting on MNIST dataset is obtained by the model constructed using the Scikit-Learn library with base classifier set as Random Forest. Similarly, the best performance on USPS dataset is obtained by the same model.
3. Majority Voting gives an accuracy of 96.41% and 43.57% on MNIST and USPS dataset respectively.
4. Implemented **Voting Classifier**, and the accuracy obtained was about **94.17%**, which is better than most of the predicted classifiers.

7.6 Results Based on Confusion Matrix

It is observed that the model mostly predicts accurately for each digit of MNIST dataset using **Majority Voting**. In USPS dataset, most of the digits were misclassified except for 2,3 and 5.

```
Confusion Matrix for Majority Voting  Using MNIST Dataset:
[[ 971    0    1    1    0    2    2    1    2    0]
 [   0 1123    2    1    0    1    3    1    4    0]
 [   8    2  990    6    3    0    6    8    8    1]
 [   0    0   10  976    0    8    0    6    8    2]
 [   1    0    3    0  955    0    6    0    2   15]
 [   6    1    0   22    4  838    7    1   11    2]
 [   9    3    2    1    4    7  930    1    1    0]
 [   2    5   19    2    2    0    0  986    1   11]
 [   5    2    3   13    6    8    7    5  917    8]
 [   7    6    1   11   16    2    1    6    4  955]]
```

```
Confusion Matrix for Majority Voting  Using USPS Dataset:
[[ 683    4  315   65  330  147   35  123   20  278]
 [  60  521  280  158  170  128   19  615   43    6]
 [  87   15 1516   80   30  154   28   68   16    5]
 [  37    2  160 1432    8  279    4   47   23    8]
 [  14  104   58   27 1178  151   11  310  108   39]
 [  74   12  243  131   19 1430   11   54   21    5]
 [ 211   21  509   54   67  302  785   26    5   20]
 [  63  220  274  417   30  191    7  689   93   16]
 [ 148   19  175  335   83  739   70   89  317   25]
 [  17  161  158  425  135   81    4  648  208  163]]
```

Figure 6: Confusion matrix on MNIST Dataset(left) and USPS Dataset(right) for Majority Voting

1. **Strengths**
   - For MNIST, all of the digits were correctly classified.The best classification was for digit 0 and then for 1.
   - For USPS, the best classification was for digit 2,3 and 5.
2. **Weakness**
   - For USPS dataset, most of the digits were misclassified except for 2, 3 and 5.

8 Conclusions

Replies to questions asked in project pdf:
Question: *Do your results support the No Free Lunch theorem?*
Answer: Yes. The No Free Lunch theorem states that there is no one model that works best for every problem. The assumptions of a great model for one problem may not hold for another problem, so it is common in machine learning to try multiple models and find one that works best for a particular problem. For our problem, though every classification

model after training on MNIST dataset gives good performance on its testing data, it didn't 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. Hence, we had seen the performance was poor in USPS dataset.We have further found out that combining the results of various classifiers using Majority Voting or any other ensembling techniques provides better result than individual testing on USPS dataset.

Question: *Observe confusion matrix of each classifier and describe the relative strengths/weaknesses of each classifier. Which classifier has the overall best performance?*
Answer: Strengths and weaknesses of confusion matrices is described in individual sections. **Convolutional Neural Network classifier** has the overall best performance. The accuracy obtained was **99.19% for MNIST and 73.34% for USPS**.

Question: *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?*
Answer: Yes. Using the dataset provided, the accuracy obtained were as follow:

Models	Accuracy - MNIST	Accuracy - USPS
Neural Network Using Keras	94.61	38.08
Neural Network Using Solver as lbfgs	97.85	48.56
Neural Network Using Solver as sgd	97.85	48.56
CNN	99.19	73.34
SVM using linear kernel	93.81	28.45
SVM using rbf kernel and gamma as 1	93.9	33.97
SVM using rbf kernel and gamma as default	94.43	40.85
SVM using poly kernel	94.01	39.01
Random Forest with n_estimators as 10	94.63	32.3
Random Forest with n_estimators as 100	96.98	38.9
Random Forest with n_estimators as 200	97.1	40.32
Majority Voting	96.41	43.57

Combinations of multiple classifiers decrease variance, especially in the case of unstable classifiers. From the table above, we can clearly see that the results of **Majority Voting** is much better than most of the individual classifers.

References

[1] Ensemble
https://scikit-learn.org/stable/modules/ensemble.html#totally-random-trees-embedding

[2] Neural Networks
http://neuralnetworksanddeeplearning.com/chap1.html/

[3] Project Description
https://ublearns.buffalo.edu/bbcswebdav/pid-4759309-dt-content-rid-20737326_1/courses/2189_24904_COMB/Project3.pdf

[4] Multinomial Logistic Regression
https://gist.github.com/cyrilq/2db97c440f0e559a9d5f4fb30567dbf5

[5] Support Vector Machine
https://scikit-learn.org/stable/modules/svm.html

[6] Random Forest
https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

[7] Convolutional Neural Network
http://adventuresinmachinelearning.com/keras-tutorial-cnn-11-lines/