Project 3: Classification

Srivenkata Krishnan Sesharamanujam
CSE574 Introduction to Machine Learning
UB Person no: 50288730
srivenka@buffalo.edu

1 Introduction

The following report is about the different machine learning approaches that can be applied for a classification problem. The classification problem here is to classify the given $28 * 28$ image as digits from $0, 1, 2....., 9$. Four classification approaches are used to train the MNIST data: First, Mutlinomial logistic Regression. Second, Multi layer Perceptron Neural Network. Third, Random Forest classifier and finally, Support Vector Machine classifier. After the MNIST data are trained using the above models, the results of all the models are compared to see which method works better for which scenario. The rest of the report is structured as follows: section 2 explains each of the classifier method used in the project while section 3 talks about the implementation of the each classifier on MNIST data, its advantages and disadvantages based on the results obtained. Section 4 explains the classification combination methods more specifically major voting method. Finally, we conclude based on the results of all classifier models.

2 Classifier Methods

2.1 Multinomial Logistic Regression

2.1.1 Logistic Regression and its types

Logistic Regression is a classification algorithm, that is used where the response variable is categorical. The idea of Logistic Regression is to find a relationship between features and probability of particular outcome. Example, When we have to predict if a student passes or fails in an exam when the previous marks scored in the exam are given as features, the response variable has two values, pass and fail. This type of a problem is referred to as Binomial Logistic Regression, where the response variable has two values 0 and 1 or pass and fail or true and false. Multinomial Logistic Regression deals with situations where the response variable can have three or more possible values. Example of a multinomial logistic regression is predict the type of flower where there are more than three flower species

2.1.2 What is Multinomial Logistic Regression ?

In simple words, multinomial logistic regression is a classification technique which is used to predict more than one target class. In our MNIST data set, the idea is to predict the different digits $0, 1, 2, ..., 9$(target classes in this example).

2.1.3 How to implement multinomial Logistic Regression ?

The underlining technique will be the same like the logistic regression for binary classification until calculating the probabilities for each target. Once the probabilities are calculated we need to transfer them into one hot encoding and use the cross entropy methods in the training process for calculating the optimized weights.

The output layer should be discrete probability distribution over the no of target classes. To be a valid output probability distribution, the output should contain

- Non negative values
- Sum should be equal to 1

We can accomplish this using a soft max function.

2.1.4 Softmax function

Given an input vector z, softmax does the following two things

- It exponentiates e^z making all values a non negative number
- Then, it normalizes so that the sum of all values is equal to 1

The softmax function for a input vector z is given as,

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

Softmax function is extensively used for binary classification while sigmoid function is used for binary logistic classification. In the next sections we will understand what one hot encoding, cross entropy and loss functions mean with respect to multinomial logistic regression.

2.1.5 One Hot Encoding

One hot encoding is a method to represent the target values or categorical attributes into a binary representation. To explain in simple words, lets take an example from the training data set which contains values say $x_1, x_2 and x_3$. Now, the one hot encoding matrix will have 1 for the target class and 0 for the other classes.

2.1.6 Cross Entropy

The cross entropy is a distance function which takes the calculated probabilities from the softmax function and the created on hot encoding matrix to calculate the distance. For the next target class, distance will be less and distance values will be larger for wrong target class.

2.1.7 Loss function

The expected output after training the multinomial logistic regression classifier is the calculated weights. Later the calculated weights will be used for the prediction task.This Parameters optimization is an iteration process where the calculated weights for each observation used to calculate the cost function which is also known as the Loss function.The input parameters for the loss function is the calculated weights and all the training observations. The function calculates the distance between the predicted class using the calculated weights for all the features in the training observation and the actual target class. If the loss function value is fewer, it means with the estimated weights, the target classes for the test data can be predicted. In the case of high loss function value, the process of calculating the weights will start again with derived weights of the previously calculated weights.The process will continue until the loss function value is less.

2.2 Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological

systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

The type of neural networks which are going to be implemented in this project are

- Multilayer Deep Neural Network (DNN)
- Conventional Neural Network (CNN)

In the following sections we will see how these neural network models are implemented.

2.2.1 Multilayer Deep Neural Network

A deep neural network is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modeling to process data in complex ways.

Deep neural networks are 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. One of the key uses of these sophisticated neural networks is dealing with unlabeled or unstructured data. The phrase deep learning is also used to describe these deep neural networks, as deep learning represents a specific form of machine learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

Multi Layer Perceptron is a subset of DNN. While DNN can have loops and Multi Layer Perceptron are always feed-forward, i.e. A Multilayer Perceptron is a finite acyclic graph.

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. MLP uses back propagation for training the network. MLP is a deep learning method.

In the next sections, we will learn about the different terminologies used with respect to multilayer perceptron.

2.2.2 Feed Forward Artificial Neural Network

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs.

2.2.3 Network Layers

The ideal artificial neural network consists of three layers, namely

- **Input Layer:** Represents the raw information that is fed into the network
- **Hidden Layer:** Determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- **Output Layer:** Depends on the activity of the hidden units and the weights between the hidden and output units.

2.2.4 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNNs have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. CNNs apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which the model can then use for classification.

Convolutional Neural Networks contains 3 components,namely

- **Convolutional layers**, For each sub region, It performs a set of mathematical operations to produce a single value in the output feature map. Convolutional layers then typically apply a ReLU activation function to the output to introduce non linearities into the model.
- **Pooling layers**, which downsample the image data extracted by the convolutional layers to reduce the dimensionality of the feature map in order to decrease processing time. A commonly used pooling algorithm is max pooling, which extracts subregions of the feature map (e.g., 2x2-pixel tiles), keeps their maximum value, and discards all other values.
- **Dense (fully connected) layers**, which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers. In a dense layer, every node in the layer is connected to every node in the preceding layer.

Typically, a CNN is composed of a stack of convolutional modules that perform feature extraction. Each module consists of a convolutional layer followed by a pooling layer. The last convolutional module is followed by one or more dense layers that perform classification. The final dense layer in a CNN contains a single node for each target class in the model (all the possible classes the model may predict), with a softmax activation function to generate a value between 01 for each node (the sum of all these softmax values is equal to 1).

2.3 Random Forest Classifier

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because its simplicity and the fact that it can be used for both classification and regression tasks.

In simple words, Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is grown as follows:

- If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
- If there are M input variables, a number $m << M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

2.4 Support Vector Machines

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 dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

The algorithm can be seen as three step process,

- Define an optimal hyperplane: maximize margin
- Extend the above definition for non-linearly separable problems that is to have a penalty term for misclassifications.
- Map data to high dimensional space where it is easier to classify with linear decision surfaces that is to reformulate problem so that data is mapped implicitly to this space.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane.

2.4.1 SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, you can say that it converts non separable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

Types of kernels are:

- **Linear Kernel:** A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.
- **Polynomial Kernel:** A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.
- **Radial Basis Function Kernel:** The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

3 Implementation and Results of Each Classifier

3.1 Metrics Used

The following are the metrics against which each model is differentiated

3.1.1 Classification Accuracy

Classification accuracy is ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{NoofCorrectPredictions}{NoofTotalPredictions} \tag{2}$$

3.1.2 Confusion Matrix

The confusion matrix is a handy presentation of the accuracy of a model with two or more classes. The table presents predictions on the x-axis and accuracy outcomes on the y-axis. The cells of the table are the number of predictions made by a machine learning algorithm.

3.2 Multinomial Logistic Regression

The code implementation of Multinomial Logistic Regression is given in figure 1.

The confusion matrix and accuracy from the Mutlinomial Logistic Regression is given by figure 2, 3

Figure 1: Code Snippet of Multinomial Logistic Regression

```python
def softmax(x):

    x -= np.max(x)
    return (np.exp(x)).T / (np.sum(np.exp(x),axis= 1)).T

def oneHotEncoding(y):

    val = scipy.sparse.csr_matrix((np.ones(y.shape[0]),(y,np.array(range(y.shape[0])))))
    return np.array(val.todense()).T

def computeLoss(softmax_x, oneHotEncoding_y ,weight, n):

    return ((-1/n) * np.sum(np.dot(oneHotEncoding_y,np.log(softmax_x)))) + (1/2) * np.sum(weight*weight)

def computeGradient(x,softmax_x , oneHotEncoding_y,weight, n):

    difference = oneHotEncoding_y - softmax_x.T

    return (-1/n) * np.dot(x.T,difference ) + weight

def mulitvariateLogisticRegression(x,y):

    x = np.asarray(x)
    y = np.asarray(y)
    weight = np.zeros([x.shape[1],len(np.unique(y))])
    no_Of_Iterations = 300
    learningRate = 0.1
    losses = []

    for i in range(no_Of_Iterations):

        softmax_x = softmax((np.dot(x,weight)))
        onehotcoding_y = oneHotEncoding(y)
        loss = computeLoss(softmax_x,onehotcoding_y,weight,x.shape[0])
        gradient = computeGradient(x,softmax_x,onehotcoding_y,weight,x.shape[0])
        losses.append(loss)
        weight = weight - (learningRate * gradient)
```

3.3 Multi layer Classifier

The code is implemented using the sklearn MLPClassifier method. The confusion matrix is given by figures 4, 5

The advantages of Multi Layer perceptron are,

- Capability to learn non-linear models
- Capability to learn models in real-time

The disadvantages are,

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

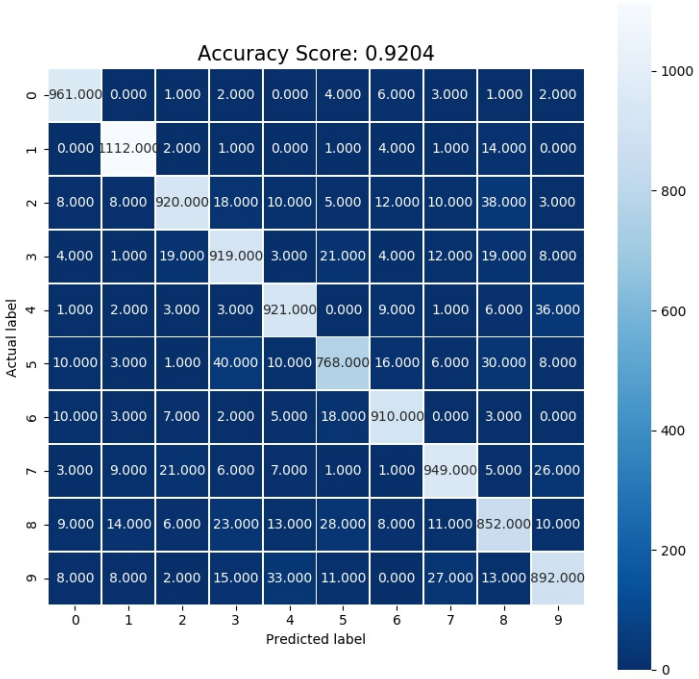
3.4 Random Forest Classifier

The algorithm is implemented using sklearn RandomForestClassifier method. The confusion matrix for the MNIST and USPS data is given by 6, 7

The Hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster.

- Increasing the Predictive Power: There is the ”n_estimators” hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and

Figure 2: Confusion Matrix of Multinomial Logistic Regression for MNIST Data



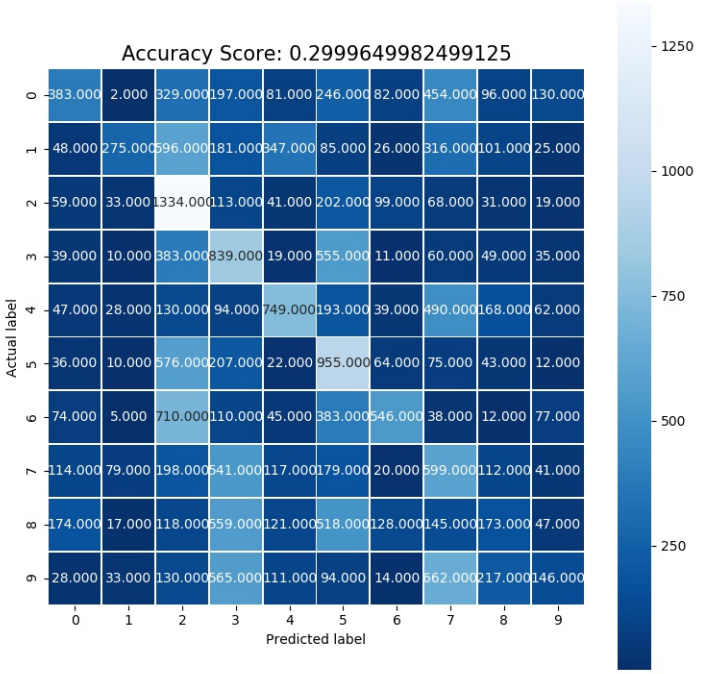
makes the predictions more stable, but it also slows down the computation.Another important hyperparameter is "max_features", which is the maximum number of features Random Forest is allowed to try in an individual tree.The last important hyper-parameter we will talk about in terms of speed, is "min_sample_leaf". This determines, like its name already says, the minimum number of leafs that are required to split an internal node.

- Increasing the Models Speed: The "n_jobs hyperparameter tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of -1 means that there is no limit."random_state makes the models output replicable. The model will always produce the same results when it has a definite value of "random_state" and if it has been given the same hyperparameters and the same training data. Lastly, there is the "oob _score (also called oob sampling), which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out of bag samples. It is very similar to the leave-one-out cross-validation method, but almost no additional computational burden goes along with it.

The advantages of this classifier are : First, it can be used for both regression and classification tasks. Second, its easy to view the relative importance it assigns to the input features. Third, it is also considered as a very handy and easy to use algorithm, because its default hyperparameters often produce a good prediction result. Fourth, the number of hyperparameters is also not that high and they are straightforward to understand.

The disadvantages if this classifier are: First, a large number of trees can make the algorithm to slow and ineffective for real-time predictions. Second, A more accurate prediction requires more trees, which results in a slower model. Third, it is a predictive modeling tool and not a descriptive tool.

Figure 3: Confusion Matrix of Multinomial Logistic Regression for USPS Data



That means, if you are looking for a description of the relationships in your data, other approaches would be preferred.

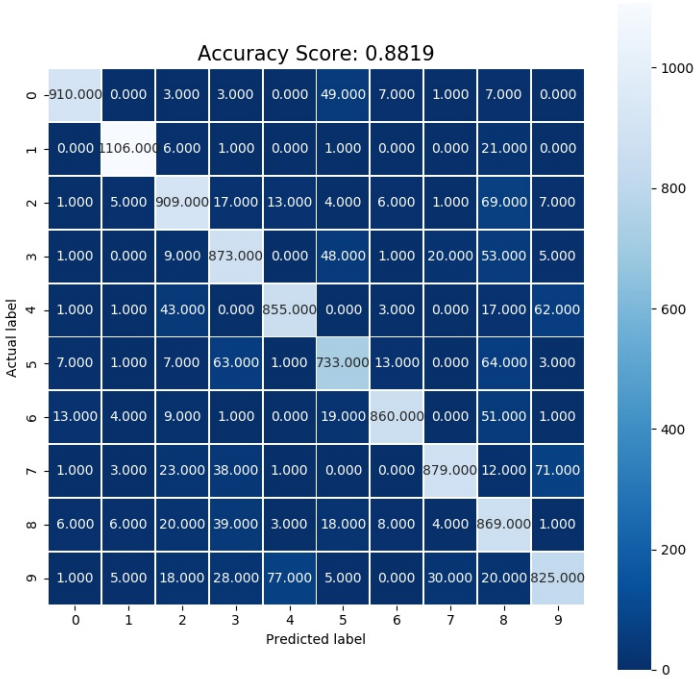
3.5 Support Vector Machines

The algorithm is implemented using sklearn SVM classifier. The confusion matrix for the MNIST and USPS data by the default SVM Classifier is given by 8, 9. The SVM classifier with RBF as input confusion matrix is given by, 10 , 11

The advantages of Support Vector Machines are: First, Effective in high dimensional spaces. Second , effective in cases where number of dimensions is greater than the number of samples. Third, Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. Fourth, it is Versatile that is different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of Support Vector Machines are: First, if the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial. Second, SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Figure 4: Confusion Matrix of MLP for MNIST Data



4 Classifier Combination Methods

4.1 Hard Voting aka Majority voting

Majority Voting is an idea where conceptually similar or different classifier methods are combined. Here, we predict the final class label as the class label that has been predicted most frequently by the classification models.

Hard voting is the simplest case of majority voting. Here, we predict the class label y via majority voting of each classifier C_j

$$y = mode[C_1(x), C_2(x), ..., C_m(x)] \tag{3}$$

The confusion matrix for both the data sets are given by figures **??**, **??**

5 Conclusion

The report details each of the classifier method with its advantages and disadvantages. Now, we have a clear idea as to which model applies best for a certain scenario.

Figure 5: Confusion Matrix of MLP for USPS Data

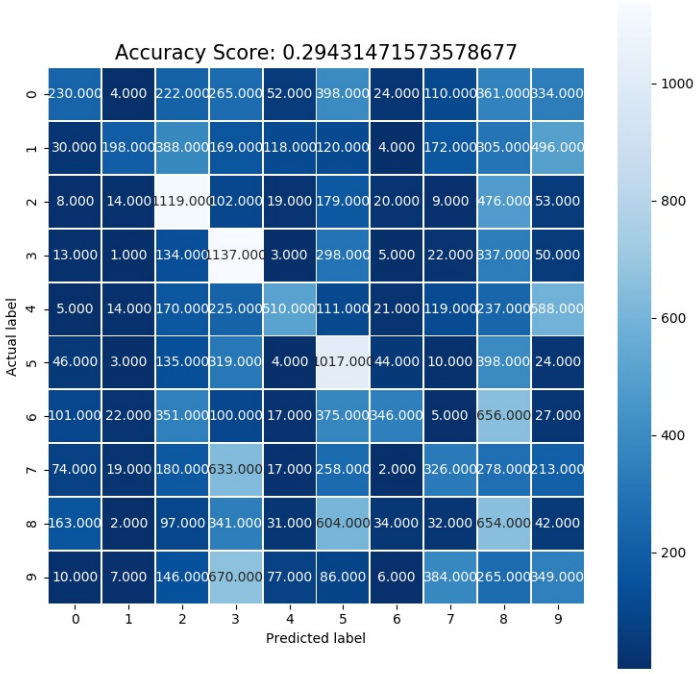


Figure 6: Confusion Matrix of Random Forest for MNIST Data

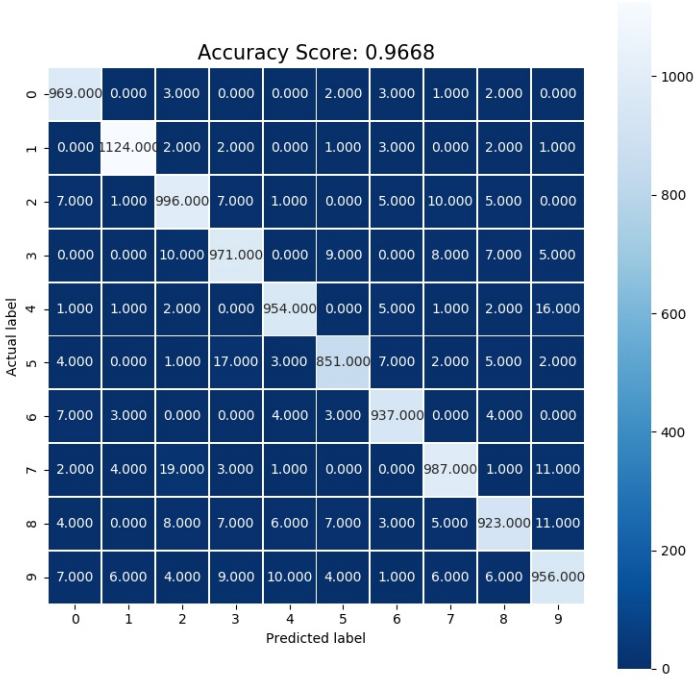


Figure 7: Confusion Matrix of Random Forest for USPS Data

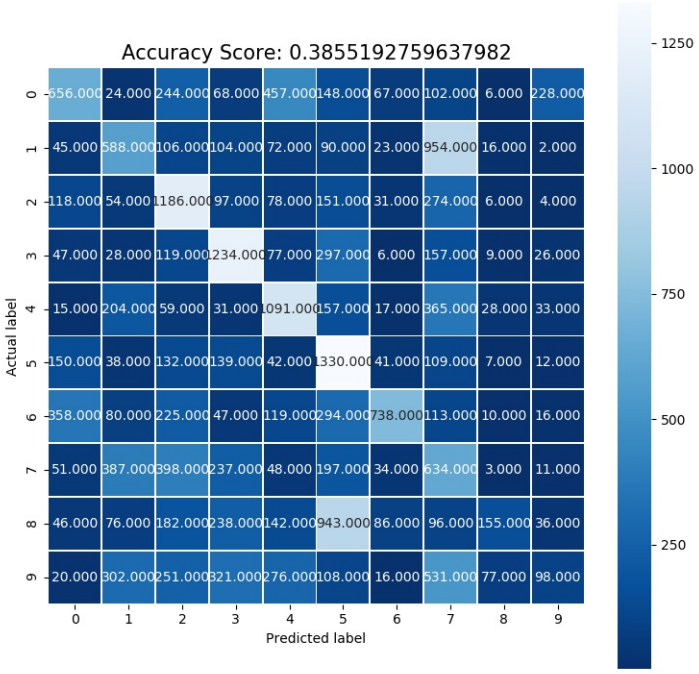


Figure 8: Confusion Matrix of SVM(Default) for MNIST Data

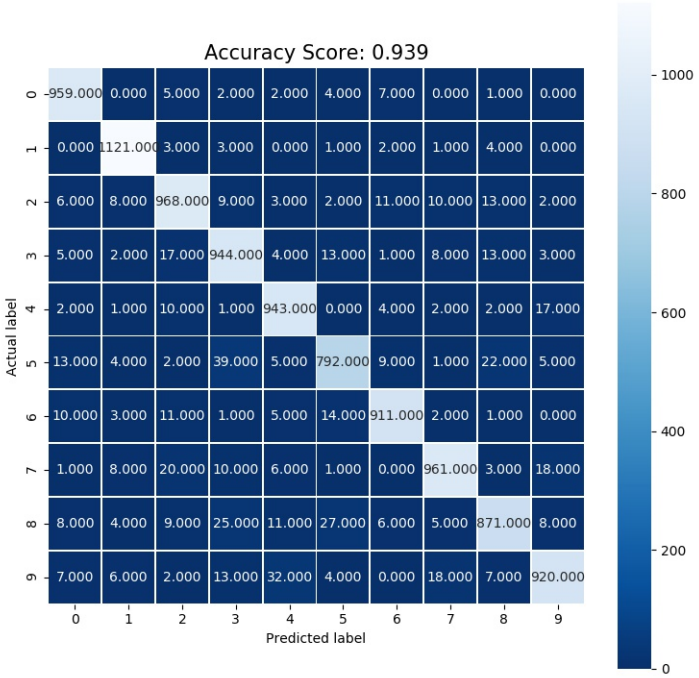


Figure 9: Confusion Matrix of SVM(Default) for USPS Data

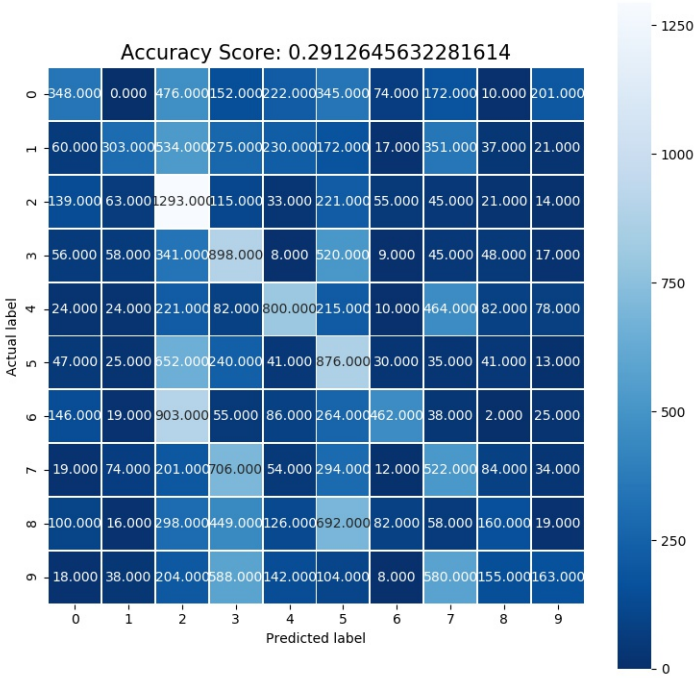


Figure 10: Confusion Matrix of SVM(RBF) for MNIST Data

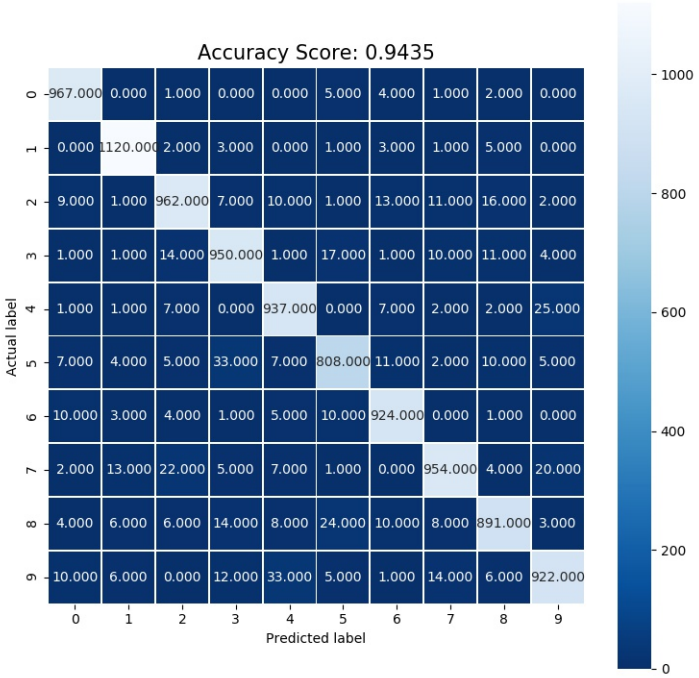


Figure 11: Confusion Matrix of SVM(RBF) for USPS Data

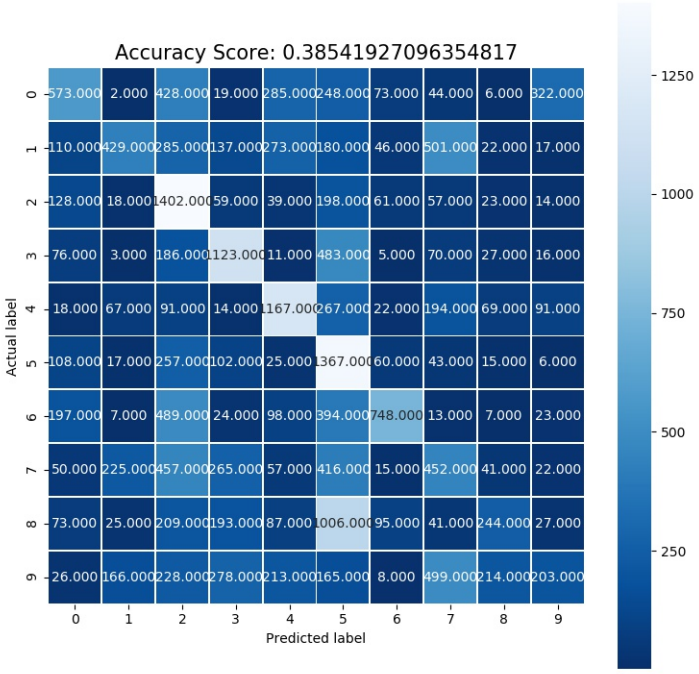


Figure 12: Confusion Matrix of Majority Voting for MNIST Data