17:610:596

Machine Learning for Data Science

Special Topic Course

Fall 2017

December 17, 2017

FINAL PROJECT
DIGITAL RECOGNIZER

AARTHI MURALI

ANANTH SANKARASUBRAMANIAN

KRISHNAMOHAN PATHICHERIKOLLAMPARAMBIL

Table of Contents:

List of Figures:

1. Introduction:

The data is collected from MNIST (Modified National Institute of Standards and Technology) database which has a large number of hand-written digits which is used for image processing processes. This hand written binary digits is a subset from a larger database – NIST Database. The data is split into two file formats – training and testing sets. This is the most commonly resourced dataset by researchers when a new algorithm is developed. Learners can start out with this data to learn pattern recognizing methods on real world data.

The hand-written digits have been processed from an image format down to strings of pixel values and the image is in 28*28pixel form. The original image from NIST is normalized such that it fits in a 20*20pixel box. The images were then centered in a 28*28 scale by computing the center of mass of the pixels. Considering that it is a 28*28pixel form, there are totally 784 features containing the image's associated pixel values in the training set. The training set also has a label column which identifies the digit, hence there are totally 785 columns in the training set. The testing set is similar to the training except for there is no label column in the testing set. The training set has a total of 42000 observation while the training set has 8398 observations. Each pixel has a value associated with it and ranges from 0 to 255 indicating the lightness or darkness of the pixel on a gray scale.

Our goal is to correctly identify the handwritten digits from the huge collection of dataset using various appropriate machine learning techniques. Since this is a classification based dataset, we use the different classification algorithms to predict the handwritten digits correctly. Once the cleaning and pre-processing is over, we go into modelling the dataset where the classification algorithms are used. We start with the traditionally used algorithms – KNN, Naïve Baye's to model the dataset and predict the accuracy. Since this is an image classification problem, it is natural to explore the neural network and Convolution neural network algorithms to classify the handwritten digits. In this course of action, we try to find the model which most accurately predicts the digits.

2. Data Analysis and Preprocessing:

2.1 Exploratory Data Analysis:

To model and predict the correct digits from the dataset, we are using the R platform to conduct the analysis. We are performing the models on R because it's an open source software tool which provides excellent visualizations and is also strong computationally. We begin by loading the training and testing sets of the MNIST dataset into R.

The dataset is visualized with respect to the label to know the frequency of occurrence for each digit in the label column and understand the distribution of the data. The below graph is a representation showing the frequency of the digits in the dataset.

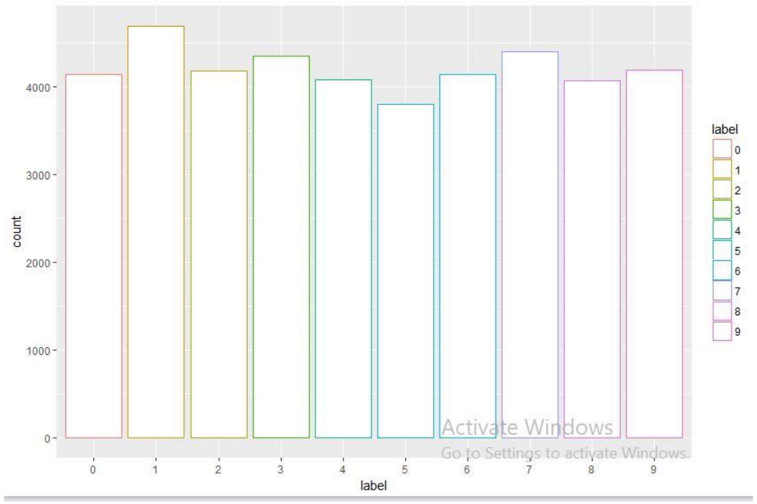


Fig 1- Observing the frequency of the digits in the data

The label column is then converted into a factor before modelling the algorithms. Since the data is in the form of pixel values, we are converting into its original image form to view the digits.

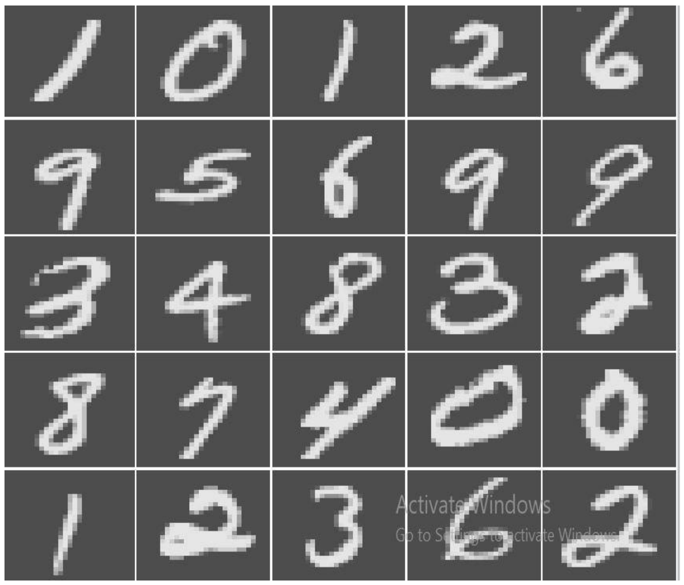


Fig 2 – Image of the handwritten digits from the pixel values

2.2 Image Preprocessing:

As explained previously, the images of the hand - written digits are converted in to a data frame with each column representing the intensity of a pixel in the image. While considering all the pixel values for development, the model might give us a good accuracy. The entire process is computationally heavy and resource intensive, not to forget the entire process could potentially take a long time to execute.

In the light of the of the above circumstances, there must be a process to reduce the number of features under consideration. This technique is more commonly known as dimensionality reduction where we try to eliminate a few features that do not unduly effect the performance of the model while making predictions.

In the Given Dataset, there are about 784 features and running classifiers on such a large number of predictors would negatively affect the accuracy. We pre-process the dataset in the following manner:

1. Identification of Near Zero Variance:

   In this process, we diagnose the predictors that have only one unique value or ones that have that meet the following conditions:

   - They have very few unique values relative to the number of samples
   - The ratio of the most common value to the second most common value is large.

In the above dataset, we remove those columns or the predictors that have near zero variance. This way we can reduce the number of predictors from 784 to about 581.The ones that were removed only added to the computational cost of the program.

2. Principal Component Analysis:

   In this procedure, we use orthogonal transformations to covert possibly correlated observations into a set of values of linearly uncorrelated variables called principal components. The advantage being that it helps reduce the storage space, computational time and redundant features.

We first obtain the plot of the variance explained by the first 100 principal components as below:

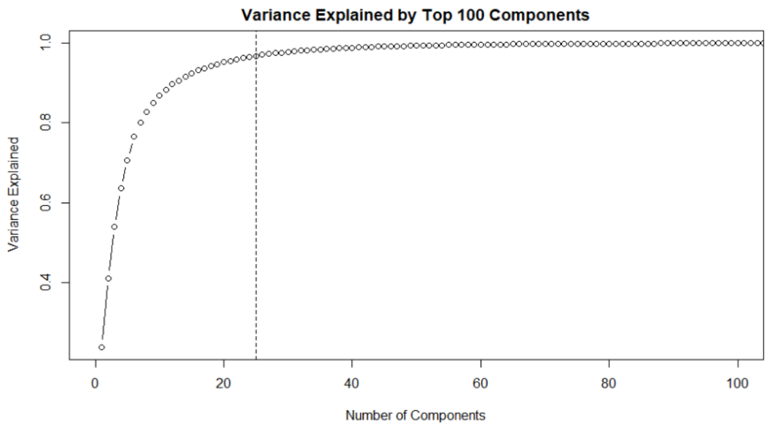


Fig 3 – Variance explained by top 100 principal components

We observe that about 25 components are sufficient to explain the variance explained in the dataset. Consequently, we only consider the features that are present in the 25 components. This would lead to slightly blurred images but it is sufficiently good to make predictions. The figure below is illustrative of the first 2 principal components of the dataset.

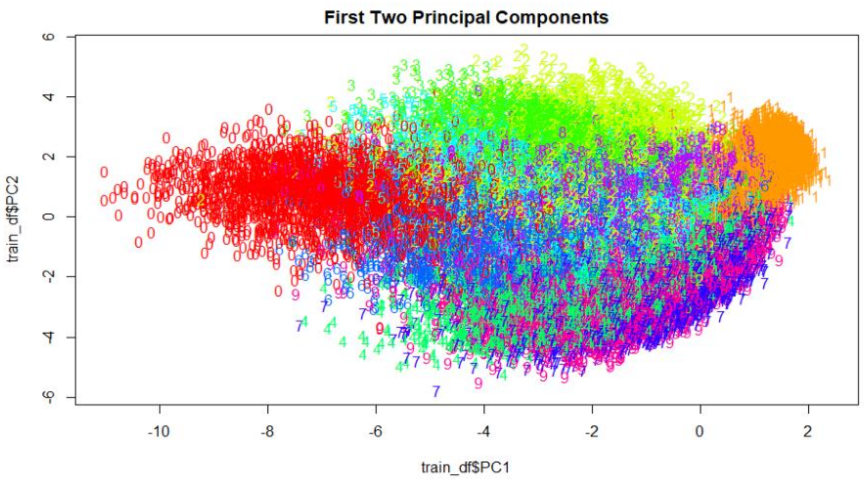


Fig 4 – Illustration of the first two principal components in the dataset

To get an idea of what Dimensionality reduction does to the image, we shall reconstruct the images after considering only the features that are a part of the selected principal components. The following images are the result after handling the near zero variance columns and Principal component Analysis.

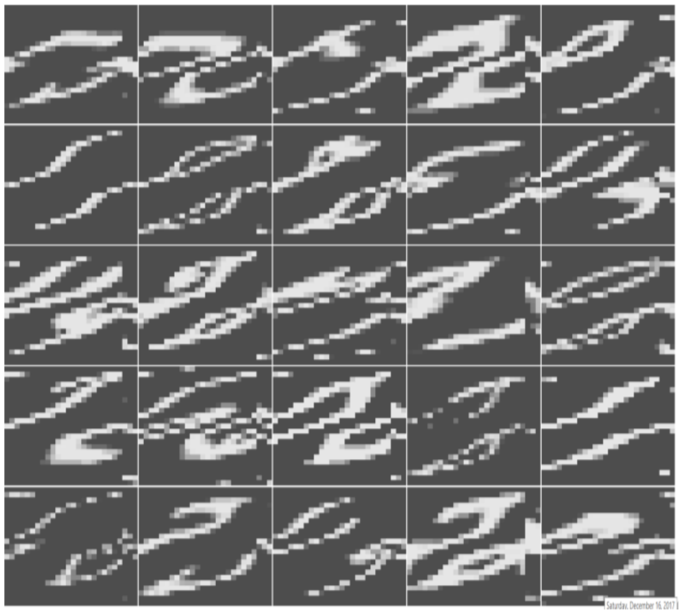


Fig 5 – Representation of the image after principal component analysis

In the coming sections we talk about the different classifiers available for classification, their applicability to the dataset at hand and the results obtained while using them.

3.Model Generation:

In this section, we experiment with different classifiers. We begin with the basic ones and incrementally move to more specialized classifiers for the purpose of comparison.

3.1 Decision Trees

Decision Trees are Rule based classification Algorithms. Given a dataset, the algorithm must choose an ideal next split for the dataset on one of the features. This is done by considering the Information Gain which is the decrease in entropy after the dataset is split on an attribute. Long story short, constructing a decision tree is about finding the attribute that returns the highest information gain.

Although, not entirely, suitable to the problem at hand, it would be good way to judge the effectiveness of the algorithm on a dataset such as ours. The following is the Decision Tree.

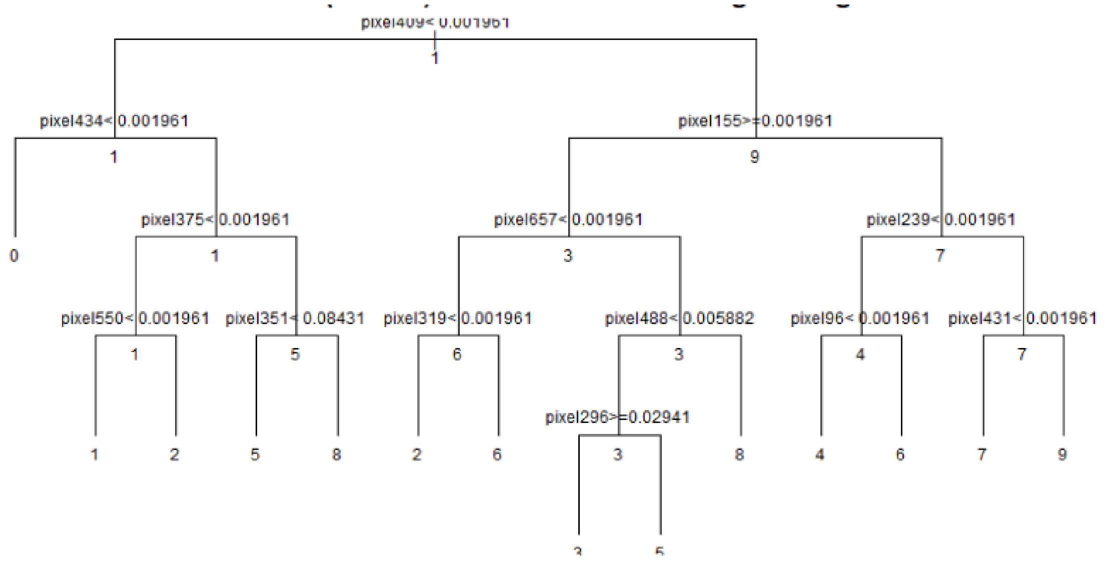


Fig 6 – Decision Tree of the dataset

What we are try to do here is, lay down rules such as "If this pixel is dark, it could most probably be this particular digit". Let us examine the performance by looking at the Confusion Matrix:

	Predicted Class										
		0	1	2	3	4	5	6	7	8	9
	0	695	1	4	32	4	45	18	9	20	24
	1	3	760	19	10	34	2	2	22	76	17
	2	60	52	447	5	16	22	51	21	107	40
Actual Class	3	67	12	62	377	13	85	39	45	97	79
	4	16	7	36	4	524	8	42	9	36	103
	5	93	16	28	35	13	313	30	83	45	94
	6	57	3	85	1	38	34	514	1	46	55
	7	62	35	7	1	67	0	0	625	8	55
	8	8	34	22	27	12	36	26	24	509	102
	9	13	4	10	15	51	6	11	136	38	511

The Accuracy obtained is only about 63.41%. As we suspected a rule based classification approach to the given problem would not yield good results. While Decision Trees do help in generalization, the same does not do well for the given problem.

3.2 Random Forests:
While Decision Trees did not return a good accuracy, is it possible to better the accuracy by considering several decision trees? Well this is what we tried to explore using the Random Forest Algorithm. The Algorithm works by growing many classification trees. The input vector is sent to each tree in the forest and each tree gives a classification. The output is the class with maximum votes.

Random Forest is known to be one of the best performing algorithms and is capable of running on large datasets such as ours. Also, the more important advantage being that it does not overfit the dataset. The following table is the confusion matrix obtained by considering 100 trees.

		Predicted Value									
		0	1	2	3	4	5	6	7	8	9
Actual Value	0	778	0	5	0	0	2	9	0	5	4
	1	0	916	0	0	2	0	0	3	6	3
	2	3	2	808	10	6	1	1	10	8	2
	3	1	10	3	818	1	13	0	2	17	12
	4	2	1	5	0	763	9	1	11	5	23
	5	5	3	1	17	1	720	12	0	15	4
	6	11	3	6	2	5	4	837	1	5	1
	7	1	1	12	9	4	0	0	825	5	18
	8	6	3	14	21	1	7	2	2	751	6
	9	1	0	1	12	14	5	0	13	3	732

The Accuracy obtained is about: 94.58 %. This is a huge increase over the accuracy obtained from Decision Trees, but, is it possible to obtain a higher accuracy considering more number of trees in the forest. The Following Graph illustrates the Accuracies obtained for different number of trees.

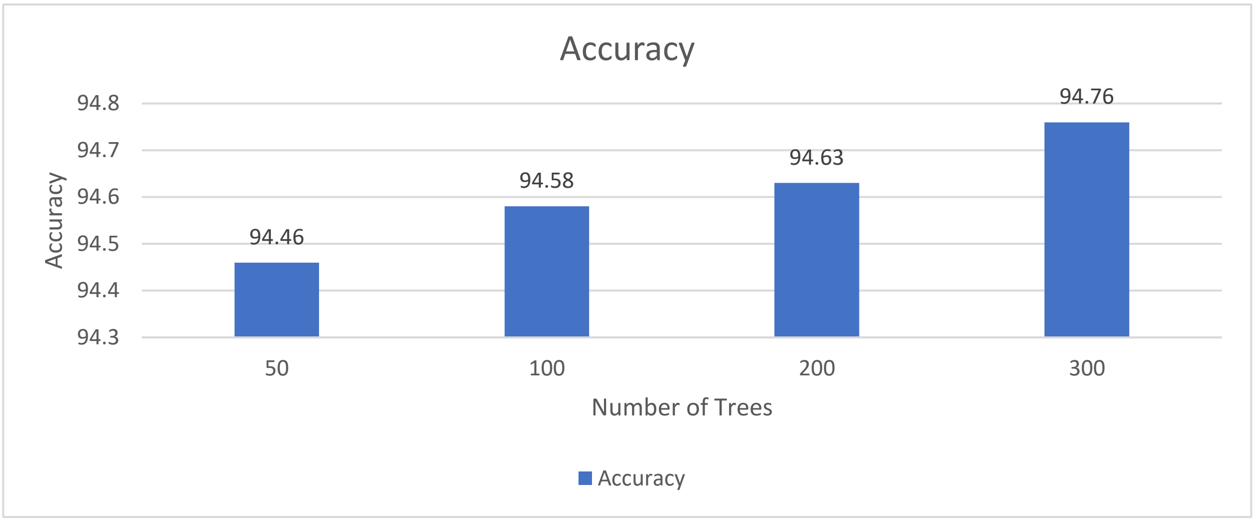


Fig 7 – Accuracies for different number of trees in random forests modelling

3.3 K-Nearest Neighbors:

The KNN Algorithm classifies the new case based on the similarity measure, (i.e.) distance function. A case is classified by majority vote of its neighbors, with the case being assigned to the class that is most common among its K nearest neighbors.

KNN is a very simple algorithm that makes no assumptions about the data. Although it is computationally a bit heavy, it would be exciting to see how it performs on our dataset. Intuitively, it seems that digits that have similar pixel values would be closer together. So we can reasonably expect the new point to be classified as the digit/digits it is closer to.

The following is the confusion matrix obtained by considering 3 Nearest Neighbors.

	Predicted Values										
		0	1	2	3	4	5	6	7	8	9
Actual Values	0	797	1	1	0	0	1	5	1	2	0
	1	0	928	3	2	2	0	2	1	0	1
	2	0	2	832	1	1	0	0	17	0	2
	3	1	2	5	845	0	14	1	6	9	6
	4	0	2	1	0	770	0	3	1	2	18
	5	1	1	0	12	0	732	9	0	3	3
	6	6	0	0	0	1	4	851	0	0	0
	7	1	8	4	2	7	0	1	836	0	8
	8	1	4	2	7	1	7	6	2	787	3
	9	2	1	1	1	9	4	0	8	1	778

The Accuracy obtained is 97.06%. To be honest we did not expect it to perform so well. KNN actually performs better than both Decision Trees as well as Random Forest for our task. But does this accuracy depend on the number of Nearest Neighbors we consider? The following graph shows the accuracy obtained for different values of K.

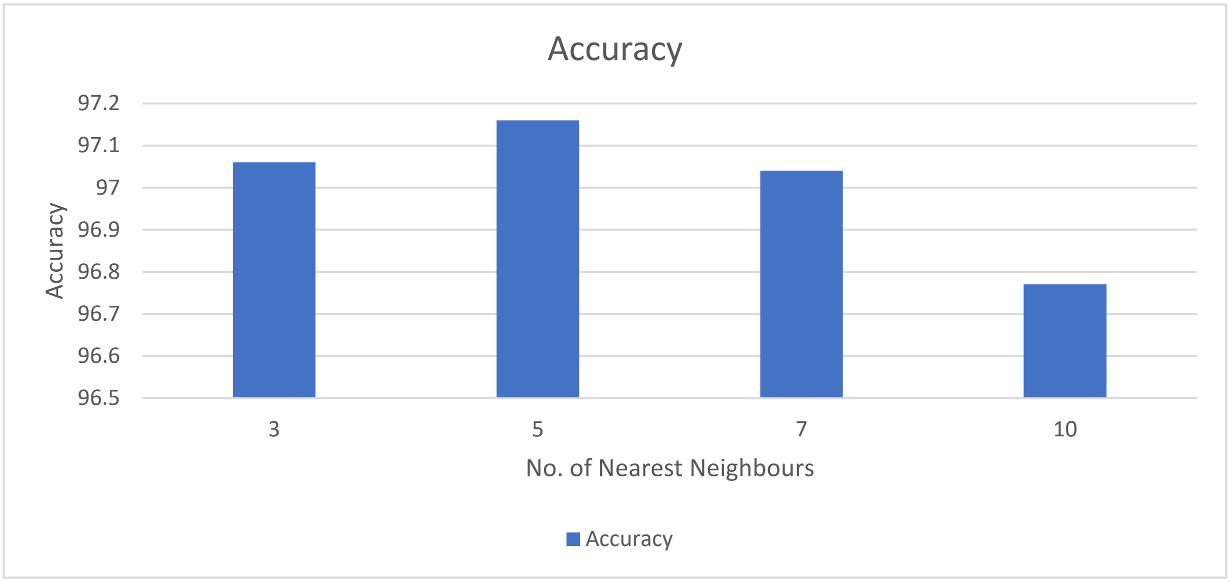


Fig 8 – Accuracies for different values of K in KNN modelling

3.4 Neural Network

A neural network consists of a collection of artificial neurons much like the neurons on the brain. Each neuron has a set of inputs and associated weights, a bias input and a single output. The neuron might fire for a combination of inputs, based on the activation function which serves as the input for another neuron within the neural network.

Neural network follows a different problem solving approach than the conventional machine learning algorithms like KNN or logistic regression. Conventional machine learning algorithms executes a set of instructions (usually iteratively) to solve a problem. Neural networks learn the problem at hand like a human brain. They learn from the training data by searching for patterns and evolve as they see more complex data. Many modern networks even have the ability to save the predictions on the test set and learn from them to make future predictions.

The neural network (NN) discussed here has been built from scratch inorder to fully understand their main components and their purpose. The NN model presented here is 2 layer fully connected network having 784 inputs (corresponding to each pixel in the 28 x 28 image) and 10 outputs (corresponding to each label from 0 to 9). The activation function chosen here is ReLU, and softmax is selected as the cost function. Ideally the best initial values for the network are chosen by several trial and errors but here we are initializing then same with random values. The network converges when the reported loss goes below the specified threshold. The main algorithm used here is called the back propagation algorithm which uses the error between the label predicted by the model and the actual label to adjust the input weight of the neurons. This process of adjusting the weights of the neurons over several iterations is the process we call as learning.

One possible problem that the network might come across is that the network might over train on the data and learns the data with high accuracy but perform poorly on the data to be predicted. This indicates that the network has memorized the learning data but has not been able to come up with a generalized model. To overcome this we split the data into training and testing set. The accuracy measured on the test set is calculated as the actual model accuracy.

The model was run on the training set and the testing set and the following accuracies were obtained.

Number of Iterations	**Training Accuracy**	**Testing Accuracy**
300	89.21	88.6
500	90.0	89.52
1000	91.22	90.54
2000	92.63	91.89
3000	93.32	91.99

3.5 Naïve Bayes

Naïve Bayes is a simple yet powerful classification algorithm found on top of the Bayes theorem. It gives you the probability of a particular set of data with different classes in it. So considering a dataset having 10 different classes and given a row of data, the naïve Bayes algorithm gives you 10 different membership probability values; one for each class. The data point shall be finally assigned to the class having the highest probability. The Naïve Bayes algorithm assumes that all the features are unrelated to each other and that the presence/absence of a feature does not influence any other feature.

Our model has the following random variables:

$C \varepsilon \{0,1,2\ldots9\}$: the digit label.

$x \varepsilon \{0,255\}$: the data map

$\theta \varepsilon [0,1]$: the activation probability. It represents the probability that bit is turned on in a data map which is labeled c.

$\pi \varepsilon [0,1]$: the label probability. It represents that probability that a given bitmap has label c.

Since the features are independent in the naïve Bayes approach we get the following joint distribution:

$p(c,x,\theta,\pi)=p(c,x|\theta,\pi)p(\theta,\pi)=p(x|c,\theta,\pi)p(c|\theta,\pi)p(\theta,\pi)=p(\theta,\pi)\pi c\prod ip(xi|c,\theta)$

The MNIST database is,

$D=\{c(n),x(n)\}n=1,\ldots,ND=\{c(n),x(n)\}n=1,\ldots,N$ and we are interested in computing the distribution $p(c|x,D)p(c|x,D)$.

The joint posterior distribution is,

$p(c,x,\theta,\pi|D)=p(c,x|\theta,\pi,D)p(\theta,\pi|D)=p(c,x|\theta,\pi)p(\theta,\pi|D)$

The confusion matrix obtained for the model is given below:

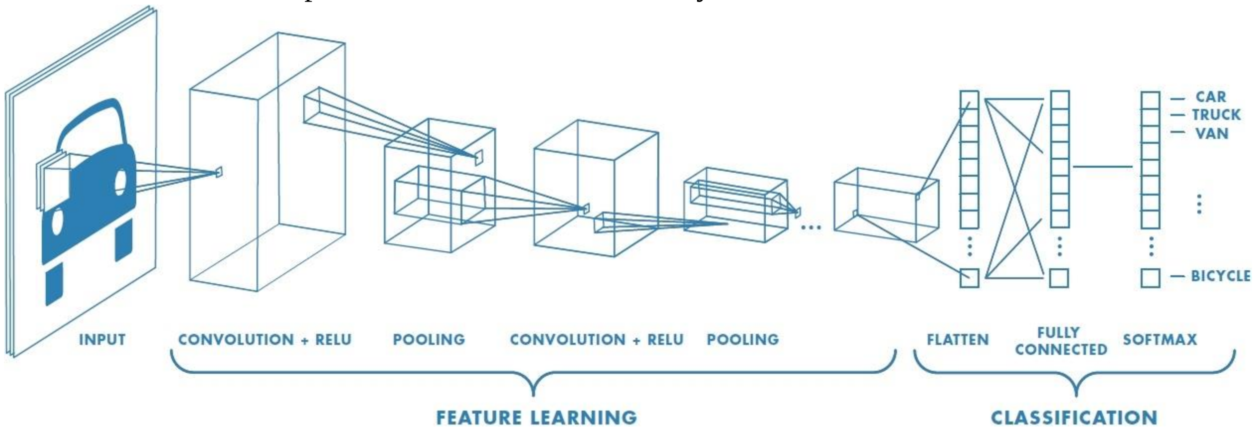
		Predicted Values									
Actual Values		0	1	2	3	4	5	6	7	8	9
	0	592	1	2	0	1	2	41	0	19	9
	1	0	728	1	1	0	1	6	0	14	6
	2	82	26	110	78	2	4	222	2	155	10
	3	57	55	2	209	0	1	48	1	255	65
	4	30	10	2	2	68	2	73	2	122	372
	5	74	16	2	10	2	17	46	0	342	57
	6	6	11	0	0	0	2	611	1	8	3
	7	7	11	0	3	3	2	10	201	24	442
	8	14	110	1	9	2	1	19	1	402	92
	9	2	22	0	1	4	0	2	2	17	636

We obtained an accuracy of 53.03% from our model. The naïve Bayes algorithm is more suitable for text prediction applications which relies heavily on conditional probability and not so much for image data.

3.6 Convolutional Neural Network (CNN)

CNN models currently regarded as the best models for classification of image data because of their powerful feature learning abilities. CNN consists of a number of convolutional layers, drop out layers, pooling layers and fully connected layers which together help in extracting and learning a large variety of features from the image.

2D convolution layers are used for images and they look at the image (MNIST digit in this case) through as small window of 5 by 5 pixel. This window slides over the image until the whole image has been covered. The next layers look at a combination of features extracted from the previous layers and learn even more features. Thus, CNNs has the advantage of looking a small part of image at a time and finding patterns within patterns. So even if the image is skewed, rotated etc., CNNs could still make predictions with much accuracy.



Source: https://www.mathworks.com/discovery/convolutional-neural-network.html

Fig 9 – Convolution Neural Network Process flow

For the given MNIST dataset we are initially scaling the pixel values and converting the data frames to matrices. Then we reorient the matrix as a 28x28x1 matrix which forms the input for the first layer of our CNN. The model we used for the MNIST dataset consists of 2 convolutional layers with a sliding window size of 5x5. The first convolve layer has got 20 filters and the second convolve layer has got 30 layers to learn the features from the image. The pooling and dropout layers are used for data regularization which helps in minimizing the number of variables in the network. These layers choose the most relevant features from the extracted features and pass it on to the next layer.

The last two layers of the CNN is forms the fully connected layer (fc) layer which is connected to the output of all neurons in the previous layer. The fc layers combine the knowledge gained from the previous layers to finally give the probabilities for each class for a given image. Since we have 10 different classes for the MNIST data set the final fc layer will have 10 nodes. Each node will output the probability for the classes from 0 to 9.

	Predicted Value										
Actual Value	0	1	2	3	4	5	6	7	8	9	
	0	1126	0	1	2	0	0	1	0	0	1
	1	0	1672	0	0	2	0	0	10	1	0
	2	1	0	1165	1	0	0	0	9	1	0
	3	0	0	2	1343	0	3	0	0	1	2
	4	0	1	0	0	1062	0	1	0	1	7
	5	0	0	0	7	0	781	2	0	4	1
	6	2	0	0	0	1	2	1133	0	0	0
	7	0	4	1	1	0	0	0	1391	1	4
	8	1	3	1	0	0	0	1	4	1043	11
	9	1	0	0	0	1	1	0	0	2	1184

The model was run for several different epochs and the accuracies obtained on the training and testing data is given below.

Number of epochs	**Training Accuracy**	**Testing Accuracy**
10	97.61%	98.80%
20	98.33%	99.04%
30	98.723%	99.13%

4.Result:

The following Table Summarizes the Accuracies obtained by using the various classifiers on the MNIST dataset:

Models	**Maximum Accuracy**
CNN	99.13%
KNN	97.06%
Random Forests	94.58 %
Neural Network	93.32%
Decision Tree	63.41%
Naïve Bayes	53.03%

5. Conclusion:

The project was a very good window into understanding the applicability of Machine Learning Algorithms for Image Classification. Traditional Machine Learning Algorithms such as Decision Trees and Naïve Bayes performed poorly on the given dataset, probably because Decision Trees were meant for classification based on rules and Naïve Bayes for datasets that assume independence of variables.

Random Forests and KNN performed well on the given dataset. In fact the accuracies did improve while considering more number of Trees in the case of Random Forests and more number of neighbors in the case of KNN. It is therefore reasonable to expect such algorithms to perform well on other image classification problems too.

By far the best performing model was the CNN. This actually gave us an insight into deep learning and the accuracy obtained gave us a concrete reason as to why Convolutional Neural Networks perform so well on Image classification problems. Upon a little bit of research, we realize most of the real world image classification tasks use models developed using CNN.
We believe we could use our model that used the Convolutional Neural Network on tasks such as recognizing digits in forms and saving them digitally. We also believe, we could extend our project to recognize not only single digit numbers but also multiple digits which has a lot of applications in the industry.

6. References:

[1] http://yann.lecun.com/exdb/mnist/

[2]http://danielbarter.github.io/posts/statistics/2016-06-08-naive-bayes-classification-and-mnist-database.html

[3] https://www.tensorflow.org/get_started/mnist/beginners

[4] https://steven.codes/blog/ml/how-to-get-97-percent-on-MNIST-with-KNN/

[5] https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/

[6] https://www.kaggle.com/cnjn22/digit-recognizer