A project report on

HANDWRITTEN DIGIT CLASSIFICATION USING MACHINE LEARNING

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology (IT)

Under the supervision of

Ms. Vandana Choudhary
Assistant Professor

Submitted by

VIDUSHI GARG

(07914803116)

DEPARTMENT OF INFORMATION TECHNOLOGY

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

SECTOR-22, ROHINI, DELHI –110086

December, 2019

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of the people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

I am grateful to my mentor **Ms. Vandana Choudhary** for the guidance, inspiration and constructive suggestion that helped me in the preparation of this report.

I would like to thank all my friends for helping me in all measures of life and for their guidance, cooperation and moral support.

I would like to express my gratitude to all those who directly or indirectly helped me throughout the training for the successful completion.

Last but not the least; I would like to thanks all the staff at **MAIT** for being so helpful during this project.

Vidushi Garg

(07914803116)

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this project report titled, **"HANDWRITTEN DIGIT CLASSIFICATION USING MACHINE LEARNING"** submitted by me in the partial fulfillment of the requirement of the award of the degree of **Bachelor of Technology (B.Tech.)** Submitted in the Department of **Information Technology,** Maharaja Agrasen Institute of Technology is an authentic record of my project work carried out under the guidance of Ms. Vandana Choudhary, Assistant Professor.

The matter presented in this project report has not been submitted either in part or full to any university or Institute for award of any degree.

Date: 20th Nov,19 **(Vidushi Garg)**

Place: Delhi Roll No.: 07914803116

SUPERVISOR'S CERTIFICATE

It is to certify that the Project entitled **"HANDWRITTEN DIGIT CLASSIFICATION USING MACHINE LEARNING** which is being submitted by **Ms. Vidushi Garg** to the Maharaja Agrasen Institute of Technology, Rohini in the fulfillment of the requirement for the award of the degree of **Bachelor of Technology (B.Tech.)**, is a record of bonafide project work carried out by her under my/ our guidance and supervision.

(Ms. Vandana Choudhary)
Assistant Professor
Department of Information Technology

Prof. (Dr.) M.L. Sharma
H.O.D.
Department of Information Technology

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The rapid growth of new documents and multimedia news has created new challenges in pattern recognition and machine learning. The problem of handwritten digit recognition has long been an open problem in the field of pattern classification. Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. The main objective of this project is to provide efficient and reliable techniques for recognition of handwritten numerals by comparing various classification models. MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. This project performs the analysis of accuracies and performance measures of algorithms Support Vector Classification model (SVC), Logistic Regression, Decision Tree Classification model and Random Forest Classification model.

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

Handwritten digit recognition is an important problem in optical character recognition, and it can be used as a test case for theories of pattern recognition and machine learning algorithms. To promote research of machine learning and pattern recognition, several standard databases have emerged. Handwriting recognition is one of the compelling and fascinating works because every individual in this world has their own style of writing. The main difficulty of handwritten numerals recognition is the serious variance in size, translation, stroke thickness, rotation and deformation of the numeral image because of handwritten digits are written by different users and their writing style is different from one user to another user. In real-time applications like the conversion of handwritten information into digital format, postal code recognition, bank check processing, verification of signatures, this recognition is required.

This research aims to recognize the handwritten digits by using tools from Machine Learning to train the classifiers, so it produces a high recognition performance. The MNIST data set is widely used for this recognition process. The MNIST data set has 70,000 handwritten digits. Each image in this data set is represented as an array of 28x28. The array has 784 pixels having values ranging from 0 to 255. If the pixel value is ‘0’ it indicates that the background is black and if it is ‘1’ the background is white.

This study focuses on feature extraction and classification. The performance of a classifier can rely as much on the quality of the features as on the classifier itself. In this study, we compare the performance of four different machine learning classifiers for recognition of digits. The four classifiers namely Support Vector Classification model (SVC), Logistic Regression, Decision Tree Classification model and Random Forest Classification model. The main purpose of this research is to build a reliable method for the recognition of handwritten digit strings. The main contribution in this work is that Support Vector Classification model gives the highest accuracy while compared to the other classification models. Yet 100% accuracy is something that is to be achieved and the research is still actively going on in order to reduce the error rate. The accuracy and

correctness are very crucial in handwritten digit recognition applications. Even 1% error may lead to inappropriate results in real-time applications.

Handwritten character recognition is one of the practically important issues in pattern recognition applications. The applications of digit recognition includes in postal mail sorting, bank check processing, form data entry, etc. The heart of the problem lies within the ability to develop an efficient algorithm that can recognize hand written digits and which is submitted by users by the way of a scanner, tablet, and other digital devices. This project presents an approach to off-line handwritten digit recognition based on different machine learning technique. The main objective of this project is to ensure effective and reliable approaches for recognition of handwritten digits.

Handwritten digits recognition is a well-researched subarea within the field that is concerned with learning models to distinguish pre-segmented handwritten digits. It is one of the most important issues in data mining, machine learning, pattern recognition along with many other disciplines of artificial intelligence. The main application of machine learning methods over the last decade has determined efficacious in conforming decisive systems which are competing to human performance and which accomplish far improved than manually written classical artificial intelligence systems used in the beginnings of optical character recognition technology. However, not all features of those specific models have been previously inspected.

The handwritten digits are not always of the same size, width, orientation and justified to margins as they differ from writing of person to person, so the general problem would be while classifying the digits due to the similarity between digits such as 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc. This problem is faced more when many people write a single digit with a variety of different handwritings. Lastly, the uniqueness and variety in the handwriting of different individuals also influence the formation and appearance of the digits.

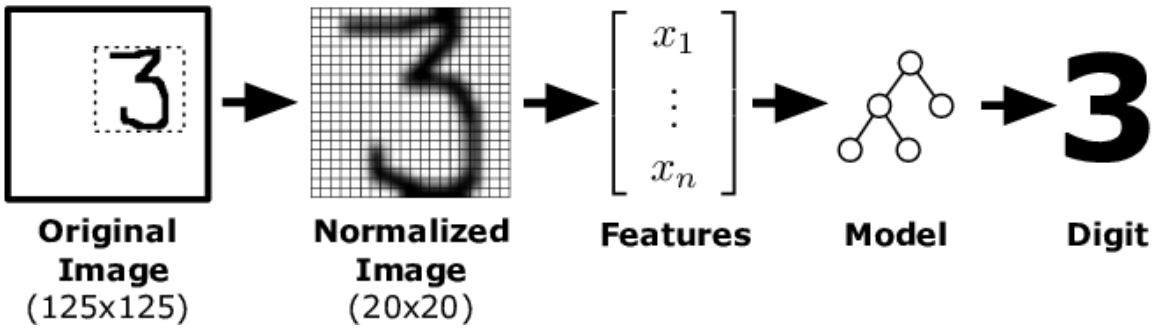


Figure 1.1: Recognition process of handwritten digits

CHAPTER-2

HARDWARE AND SOFTWARE REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

Item	Minimum	Recommended
Operating System	Windows 7/8	Windows 10
RAM	2 GB	4 GB
Processor	64 Bit i3 Processor or Equivalent	64 Bit i5 Processor or Equivalent
Storage	1 GB	3 GB

Fig 2.1 Hardware Requirements

2.2 SOFTWARE REQUIREMENTS

1. Programming Language – Python 3
2. Programming Environment – Spyder
3. Libraries Required –
   - Scikit-Learn
   - Numpy
   - Matplotlib

CHAPTER-3

LITERATURE SURVEY

An early notable attempt in the area of character recognition research is by Grimsdale in 1959. The origin of a great deal of research work in the early sixties was based on an approach known as analysis-by-synthesis method suggested by Eden in 1968. The great importance of Eden's work was that he formally proved that all handwritten characters are formed by a finite number of schematic features, a point that was implicitly included in previous works. This notion was later used in all methods in syntactic (structural) approaches of character recognition.

K. Gaurav, Bhatia P. K. Et al, this paper deals with the various pre-processing techniques involved in the character recognition with different kind of images ranges from a simple handwritten form based documents and documents containing colored and complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques of contrast stretching, binarization, noise removal techniques, normalization and segmentation, morphological processing techniques are discussed. It was concluded that using a single technique for preprocessing, we can't completely process the image. However, even after applying all the said techniques might not possible to achieve the full accuracy in a preprocessing system

Parveen Kumar, Nitin Sharma and Arun Rana made an attempt to recognize a handwritten character using SVM classifier and MLP Neural Network. Different kernel-based SVM like the linear kernel, polynomial kernel, and quadratic kernel-based SVM classifiers are used. In the SVM classifier model, there are two phases of training and testing. From each character, about 25 features are extracted with the help of which SVM is trained. Amongst the three kernels used the linear kernel gives an accuracy of 94.8%.

Reena Bajaj, Lipika Dey and Santanu Chaudhury in 2002 decided to present well built and systematic methodology for handwritten numeral recognition. In this paper, connectionist networks are used for building recognition architecture.

Patrice Y. Simard, Dave Steinkraus, John C. Platt narrated that the convolutional neural networks are better for the visual document analysis like handwriting recognition task. The main essential practices for this task are the expansion of the data sets using elastic distortions and the use of convolutional neural networks.

T.Siva Ajay also proposed that the higher rate of accuracy in handwritten digit recognition task can be achieved by the use of convolutional neural networks. The implementation of CNN is made easy and simple by the use of LeNet engineering. As a result of this accuracy greater than 98% is obtained in this project.

Ming Wu and Zhen Zhang in 2010 made a comparison between different classifiers to conclude which gives better performance in the recognition task. The comparison is done between the six classifiers namely LDA (Linear Discriminant Analysis), GMM (Gaussian Mixture Models), QDA(Quadratic Discriminant Analysis), SVML (SVM with linear kernel function), SVMR (SVM with radial basis kernel function) and k-NN. Out of all the classifiers,k-NN (k=3) gives the lowest error rate.

Haider A.Alwzwazy, Haider M.Albehadili, Younes S.Alwan, and NazE. Islam started a challenging task of recognition task of Arabic handwritten digits. For this, they decided to carry on the research using the Deep Convolutional Neural Networks. The accuracy of 95.7% is achieved as a result of this work.

In 2015, Saeed AL Mansoori applied the Multi-Layer Perceptron model to identify handwritten digits. The samples from the data set are trained by employing gradient descent back propagation algorithm and later feed forward algorithm. From the obtained results it can be observed that the digit 5 has the highest accuracy of 99.8% whereas digit 2 has the lowest accuracy of 99.04%. And the proposed system achieved an overall accuracy of 99.32%.

Xuan Yang and Jing Up focused on the handwritten multi-digit recognition on mobile using Convolutional Neural Network. In this paper, a mobile application MDig

is explained which is used to identify multiple digits using CNN. A narrow CNN is designed on mobile which gives a 99.07% accuracy using MNIST data set.

Shobhit Srivastava, SanjanaKalani, Umme Hani and SayakChakraborty illustrated the handwritten recognition using Template Matching, Support Vector Machine and Artificial Neural Networks. Among the methods used, Artificial Neural Networks turned out to give more accurate results.

CHAPTER-4

SOFTWARE REQUIREMENT ANALYSIS

4.1 PROBLEM STATEMENT

The problem of handwritten digit recognition has long been an open problem in the field of pattern classification. The main objective of this project is to provide efficient and reliable techniques for recognition of handwritten numerals by comparing various classification models. MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. This project performs the analysis of accuracies and performance measures of algorithms Support Vector Classification model (SVC), Logistic Regression, Decision Tree Classification model and Random Forest Classification model.

4.2 Description of the dataset

MNIST ("Modified National Institute of Standards and Technology") is the de facto “Hello World” dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike

The MNIST dataset, a subset of a larger set NIST, is a database of 70,000 handwritten digits, divided into 60,000 training examples and 10,000 testing samples. The images in the MNIST dataset are present in form of an array consisting of 28x28 values representing an image along with their labels. This is also the same in case of the testing images. The graylevel values of each pixel are coded in this work in the [0,255] interval, using a 0 value for white pixels and 255 for black ones.

Digits	# Training	# Testing	Subtotal
9	5949	1009	6958
8	5851	974	6825
7	6265	1028	7293
6	5918	958	6876
5	5421	892	6313
4	5842	982	6824
3	6131	1010	7141
2	5958	1032	6990
1	6742	1135	7877
0	5923	980	6903
Total	60,000	10,000	70,000

Table 4.1: Dataset

Yann Lecun, Corinna Cortes, and Christopher Burges developed this MNIST dataset for evaluating and improving machine learning models on the handwritten digit classification problem. The MNIST dataset was developed from the special dataset from NIST with special database 3 (United States Census Bureau employees) and special database 1 (high school students) which consist with the binary images of handwritten digits. Earlier SD-3 (special database -3) was considered as training and SD-1 (special database -1) as testing set with easier recognizing level of SD-3. Therefore to keep it challenging, disjoint and fair among different learning classifiers, NIST dataset was mixed up. Division of the MNIST took place by 30,000 samples from SD-3 and 30,000 samples from SD- 1 with 250 writers approx. and 5,000 samples from SD-3 and remaining 5,000 samples from SD-1 to form a different test set. Images of digits were taken from various scanned digits, normalized in size and justify as centered. This makes it an excellent dataset for evaluating models and allowing the machine learning aspirant to focus on deep learning and machine learning with very little data cleaning.

Following points are same in training and testing set along with the set of the images and labels files –

- Pixels are arranged row-wise, ranging from 0 to 255, as from RGB color code.
- Background as white (0 value from RGB) and foreground as black (255 value from RGB).
- Labels of digits classified from 0 to 9.

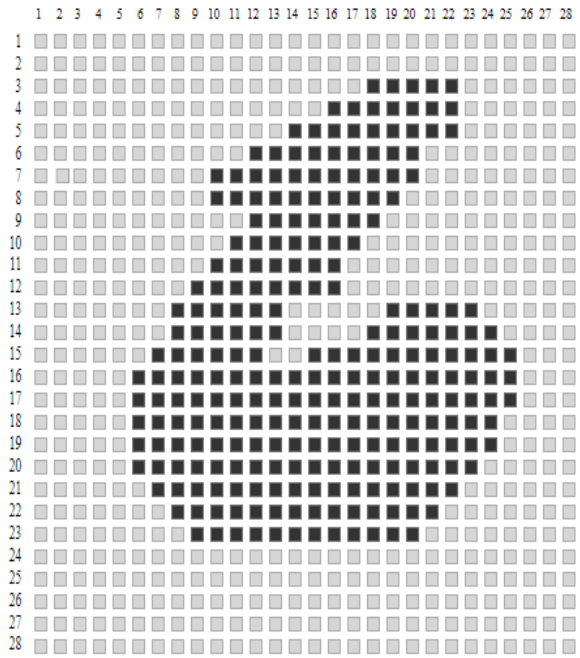


Figure 4.1: Digit 6 in 28x28 format in MNIST

4.3 Data Preprocessing

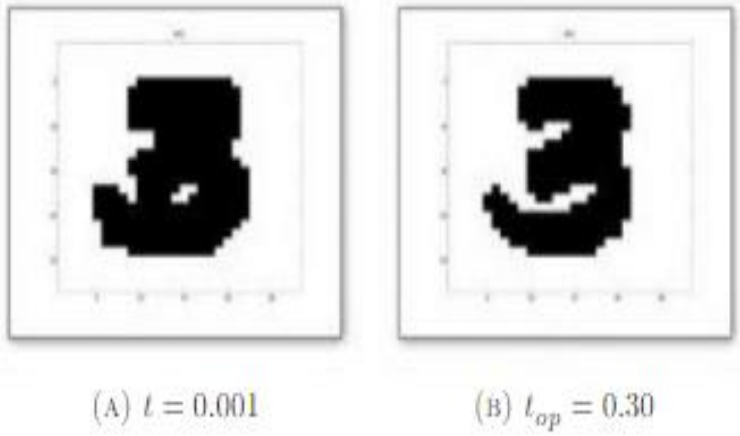
An important point for managing a high performance in the learning process is the construction of a useful training set. The 60000 different patterns contained in the MNIST database can be seen as a rather generous set, but evidence shows that the usual learning algorithms run into serious trouble for about one hundred or more of the test set samples. Therefore, some strategy is needed in order to increase the training set cardinality and variability. Usual actions comprise geometric transformations such as displacements, rotation, scaling and other distortions.

The variables proposed in this paper to make handwritten digit classification require images in binary binary level. The binarization process assumes that images contain two classes of pixel: the foreground (or white pixels, with maximum intensity, i.e., equal to 1) and the background (or black pixels with minimum intensity, i.e., equal to 0). The goal of the method is to classify all pixels with values above of the given threshold as white, and all other pixels as black. That is, given a threshold value t and an image X with pixels denoted as $x(i, j)$, the binarized image X_b with elements $x_b(i, j)$ is obtained as follows:

$$\text{If } x(i, j) > t \qquad x_b(i, j) = 1$$

$$\text{Else} \qquad x_b(i, j) = 0$$

Then, the key problem in the binarization is how to select the correct threshold, t, for a given image. We observe that the shape of any object in the image is sensitive to variations in the threshold value, and even more sensitive in the case of handwritten digit. Therefore, we consider that a binary handwritten number is better recognized computationally if its trace is complete and continuous, this is the criterion that we use to the threshold, being its choice of crucial importance.



(A) $t = 0.001$ (B) $t_{op} = 0.30$

Figure 4.2: The same digit with different threshold value

4.4 Implementation

4.4.1 Logistic Regression Model for digit recognition

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

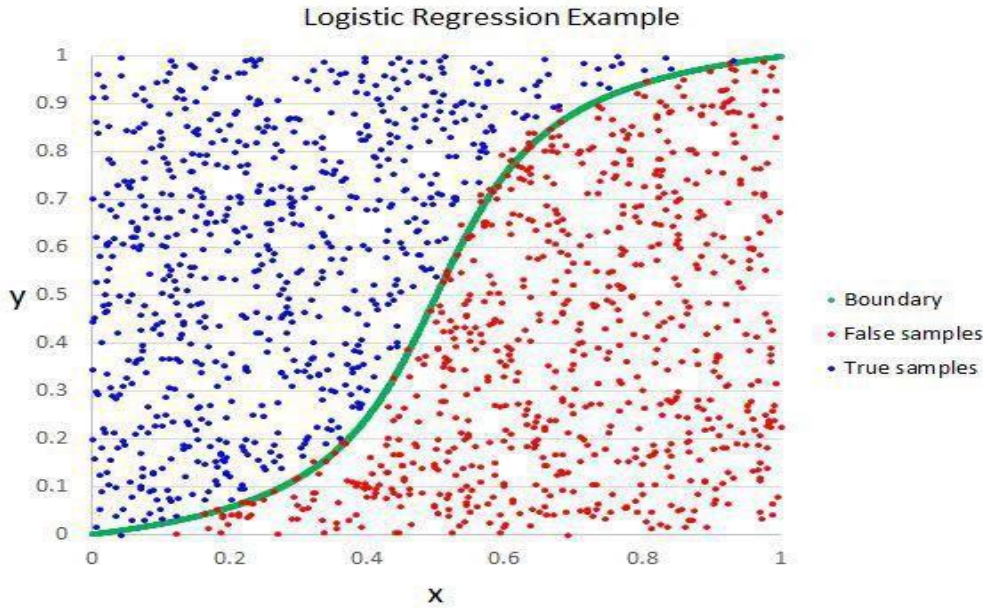


Figure 4.3: Logistic Regression Model

After that training set data is fed as input to the Logistic Regression model so that the classifier gets trained. The guessed label is matched with the original to get the accuracy of the trained classifier. Once the training is done, the testing data is given to the classifier to predict the labels and testing accuracy is obtained. The confusion matrix is generated which gives the probability between the actual data and the predicted data. Using the confusion matrix, the performance measures like precision, recall and f1 score can be calculated. Using Logistic Regression, an accuracy of 89.37% is obtained on test data.

Precision = TP/(TP+FP)

where TP = True Positive, FP = False Positive

Recall = TP/(TP+FN)

where FN = False Negative

F1score = 2*Precision+Recall/(Precission+Recall)

The confusion matrix obtained here is in the form of matrix M 10x10 because it is a multi-class classification (0-9).

The below Table 4.2 is the confusion matrix obtained for the trained data set using Logistic Regression Model.

Digit	0	1	2	3	4	5	6	7	8	9
0	985	0	3	0	3	19	11	1	2	1
1	0	1115	2	0	1	5	1	3	19	0
2	12	20	916	19	32	7	22	16	25	3
3	9	10	35	954	2	57	10	18	26	30
4	2	5	3	1	944	2	12	5	6	44
5	17	9	11	31	30	737	25	9	17	12
6	15	6	7	2	18	10	944	0	7	1
7	1	15	24	2	25	4	0	1014	3	47
8	6	34	11	32	10	31	10	10	840	21
9	8	6	7	15	44	11	2	42	6	893

Table 4.2: Confusion matric using Logistic Regression model

The below table 4.3 shows the precision, recall and f1 score values obtained for the trained data set using the Logistic Regression Model.

Digit	Precision	Recall	f1-score
0	0.93	0.96	0.95
1	0.91	0.97	0.94
2	0.9	0.85	0.88
3	0.9	0.83	0.86
4	0.85	0.92	0.89
5	0.83	0.82	0.83
6	0.91	0.93	0.92
7	0.91	0.89	0.9
8	0.88	0.84	0.86
9	0.85	0.86	0.86

Table 4.3: Precision, Recall and F1 score for Logistic Regression on trained dataset

4.4.2 Random Forest Classifier for digit recognition

Random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forest as is an ensemble of un-pruned regression or classification trees, activated from bootstrap samples of the training data, adopting random feature selection in the tree imitation process. The prediction is made by accumulating the predictions of the ensemble by superiority voting for classification. It returns generalization error rate and is more potent to noise. Still, similar to most classifiers, RF may also suffer from the curse of learning from an intensely imbalanced training data set. Since it is constructed to mitigate the overall error rate, it will tend to focus more on the prediction efficiency of the majority class, which repeatedly results in poor accuracy for the minority class.

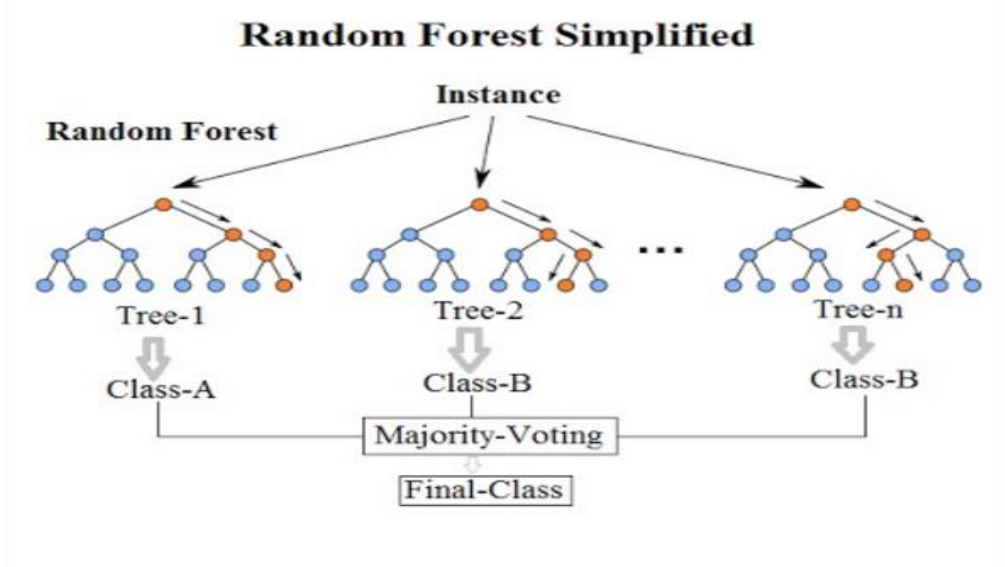


Figure 4.4: Random Forest Classifier

Here also, the confusion matrix is obtained and precision and recall values are computed.

Digit	0	1	2	3	4	5	6	7	8	9
0	1011	0	2	1	2	1	4	0	4	0
1	0	1135	5	1	1	1	2	1	0	0
2	4	5	1028	5	10	1	6	5	7	1
3	3	1	10	1090	2	11	2	11	11	10
4	2	0	1	0	995	0	7	1	0	18
5	3	2	2	12	0	851	9	2	9	8
6	8	2	0	0	3	8	986	0	3	0
7	1	7	16	2	4	1	0	1080	1	23
8	1	5	2	9	4	10	5	2	964	3
9	5	4	2	15	14	4	2	8	5	975

Table 4.4: Confusion matrix using Random Forest Classifier

The above Table 4.4 shows the confusion matrix obtained for the trained data set using the Random Forest Classifier.

The below table 4.5 shows the precision, recall and f1 score values obtained for the trained data set using the Random Forest Classifier.

Digit	Precision	Recall	f1-score
0	0.97	0.99	0.98
1	0.98	0.99	0.98
2	0.96	0.96	0.96
3	0.96	0.95	0.95
4	0.96	0.97	0.97
5	0.96	0.95	0.95
6	0.96	0.98	0.97
7	0.97	0.95	0.96
8	0.96	0.96	0.96
9	0.94	0.94	0.94

Table 4.5: Precision, Recall and F1 score for Random Forest Classifier on trained dataset

The Test Data Set obtained accuracy of 96.76% using the Random Forest Classifier on MNIST data set.

4.4.3 Support Vector Machine (SVM) Classifier for digit recognition

Support Vector Machine is a supervised machine learning technique which is applied for classification and regression. It is nothing but the representation of the input data into points in the space and mapped thus classifying them into classes. The SVM classifies or separates the classes using the hyper-plane concept. The separation margin should be equidistant from the classes.

SVM or Support Vector Machine is a specific type of supervised ML method that intents to classify the data points by maximizing the margin among classes in a high-dimensional space. SVM is a representation of examples as points in space, mapped due to the examples of the separate classes are divided by a fair gap that is as extensive as possible. After that new examples are mapped into that same space and anticipated to reside to a category based on which side of the gap they fall on. The optimum algorithm is developed through a "training" phase in which training data are adopted to develop an algorithm capable to discriminate between groups earlier defined by the operator (e.g. patients vs. controls), and the "testing" phase in which the algorithm is adopted to blind-predict the group to which a new perception belongs. It also provides a very accurate classification performance over the training records and produces enough search space for the accurate classification of future data parameters. Hence it always ensures a series of parameter combinations no less than on a sensible subset of the data. In SVM it's better to scale the data always; because it will extremely improve the results. Therefore be cautious with big dataset, as it may leads to the increase in the training time.

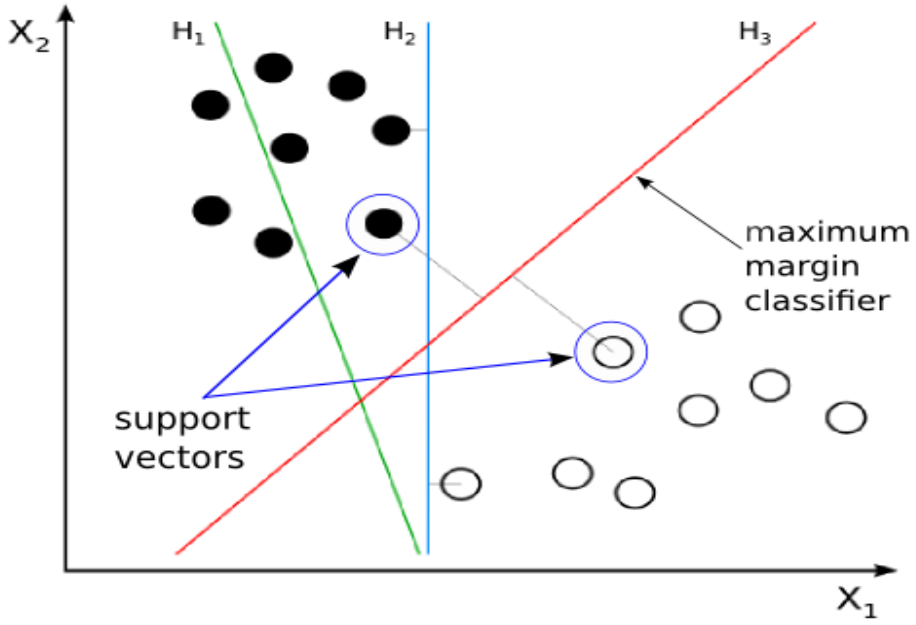


Figure 4.5: Support Vector Machine Classifier

Here also, the confusion matrix is obtained and precision and recall values are computed.

Digit	0	1	2	3	4	5	6	7	8	9
0	1018	0	2	0	0	0	3	0	0	2
1	1	1140	1	0	1	1	1	1	0	0
2	2	2	1054	3	1	1	0	6	3	0
3	2	1	4	1122	0	8	0	2	8	4
4	3	0	1	0	1000	0	5	3	1	11
5	1	1	0	7	0	879	6	0	1	3
6	3	1	0	0	0	2	1001	0	3	0
7	0	7	6	0	2	1	0	1109	1	9
8	0	2	2	4	3	3	1	1	988	1
9	4	1	1	6	8	6	2	7	4	995

Table 4.6: Confusion matrix using Support Vector Machine Classifier

The above Table 4.6 shows the confusion matrix obtained for the trained data set using the Support Vector Machine Classifier.

The below table 4.7 shows the precision, recall and f1 score values obtained for the trained data set using the Support Vector Machine Classifier.

Digit	Precision	Recall	f1-score
0	0.98	0.99	0.99
1	0.99	0.99	0.99
2	0.98	0.98	0.98
3	0.98	0.97	0.98
4	0.99	0.98	0.98
5	0.98	0.98	0.98
6	0.98	0.99	0.99
7	0.98	0.98	0.98
8	0.98	0.98	0.98
9	0.97	0.96	0.97

Table 4.7: Precision, Recall and F1 score for Support Vector Machine Classifier

The Test Data Set obtained accuracy of 98.15% using the Support Vector Machine Classifier on MNIST data set.

4.4.4 Decision Tree Classifier for digit recognition

A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning.

The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures (ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
   - All the tuples belong to the same attribute value.
   - There are no more remaining attributes.
   - There are no more instances.

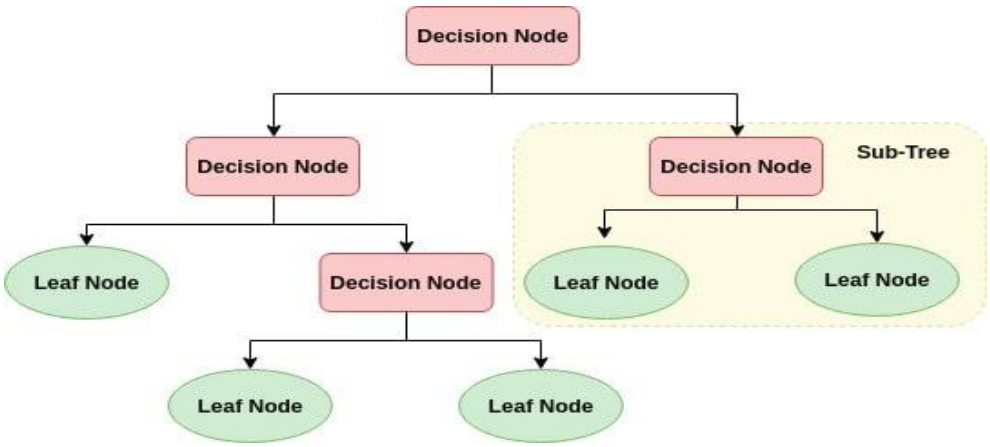


Figure 4.6: Decision Tree Classifier

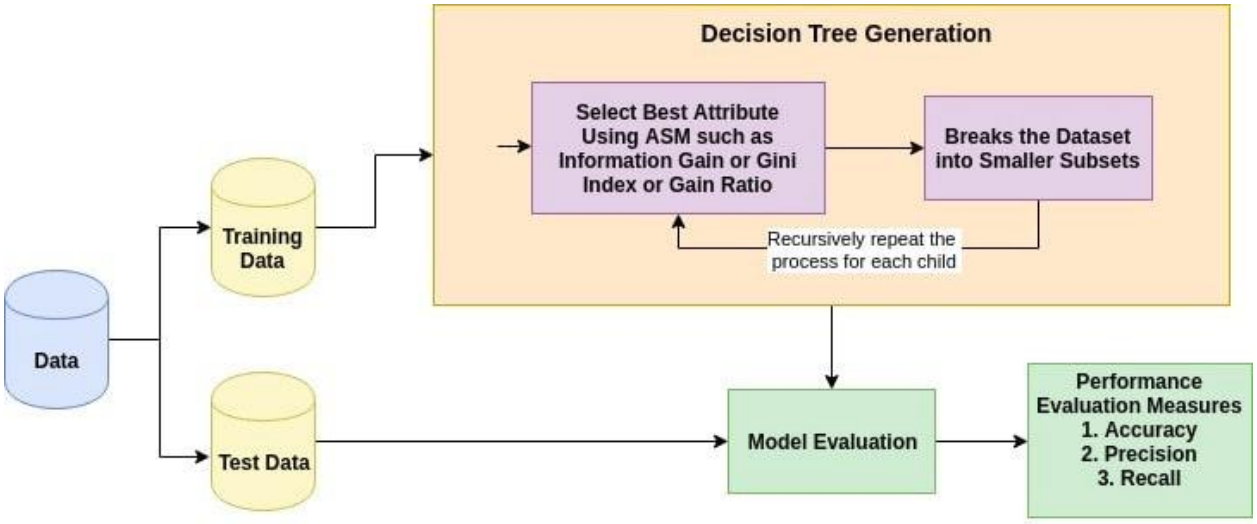


Figure 4.7: Implementation of Decision Tree Classifier

Here also, the confusion matrix is obtained and precision and recall values are computed.

Digit	0	1	2	3	4	5	6	7	8	9
0	914	1	14	14	5	25	21	4	18	9
1	1	1090	6	3	10	7	5	6	11	7
2	11	13	879	32	17	19	33	26	31	11
3	10	8	34	880	12	92	7	20	62	26
4	4	2	17	10	813	12	17	25	20	104
5	25	7	18	57	22	676	15	8	48	22
6	33	4	19	9	19	27	884	3	8	4
7	2	8	23	11	29	9	4	984	12	53
8	16	15	22	66	24	49	13	11	756	33
9	6	2	11	16	90	22	9	55	28	795

Table 4.8: Confusion matrix using Decision Tree Classifier

The above Table 4.8 shows the confusion matrix obtained for the trained data set using the Decision Tree Classifier.

The below table 4.9 shows the precision, recall and f1 score values obtained for the trained data set using the Decision Tree Classifier.

Digit	Precision	Recall	f1-score
0	0.89	0.89	0.89
1	0.95	0.95	0.95
2	0.84	0.82	0.83
3	0.8	0.76	0.78
4	0.78	0.79	0.79
5	0.72	0.75	0.74
6	0.88	0.88	0.88
7	0.86	0.87	0.86
8	0.76	0.75	0.76
9	0.75	0.77	0.76

Table 4.9: Precision, Recall and F1 score for Decision Tree Classifier on trained dataset

The Test Data Set obtained accuracy of 82.58% using the Decision Tree Classifier on MNIST data set.

CHAPTER-5

SOFTWARE DESIGN

5.1. Data Flow Diagram of Logistic Regression Model

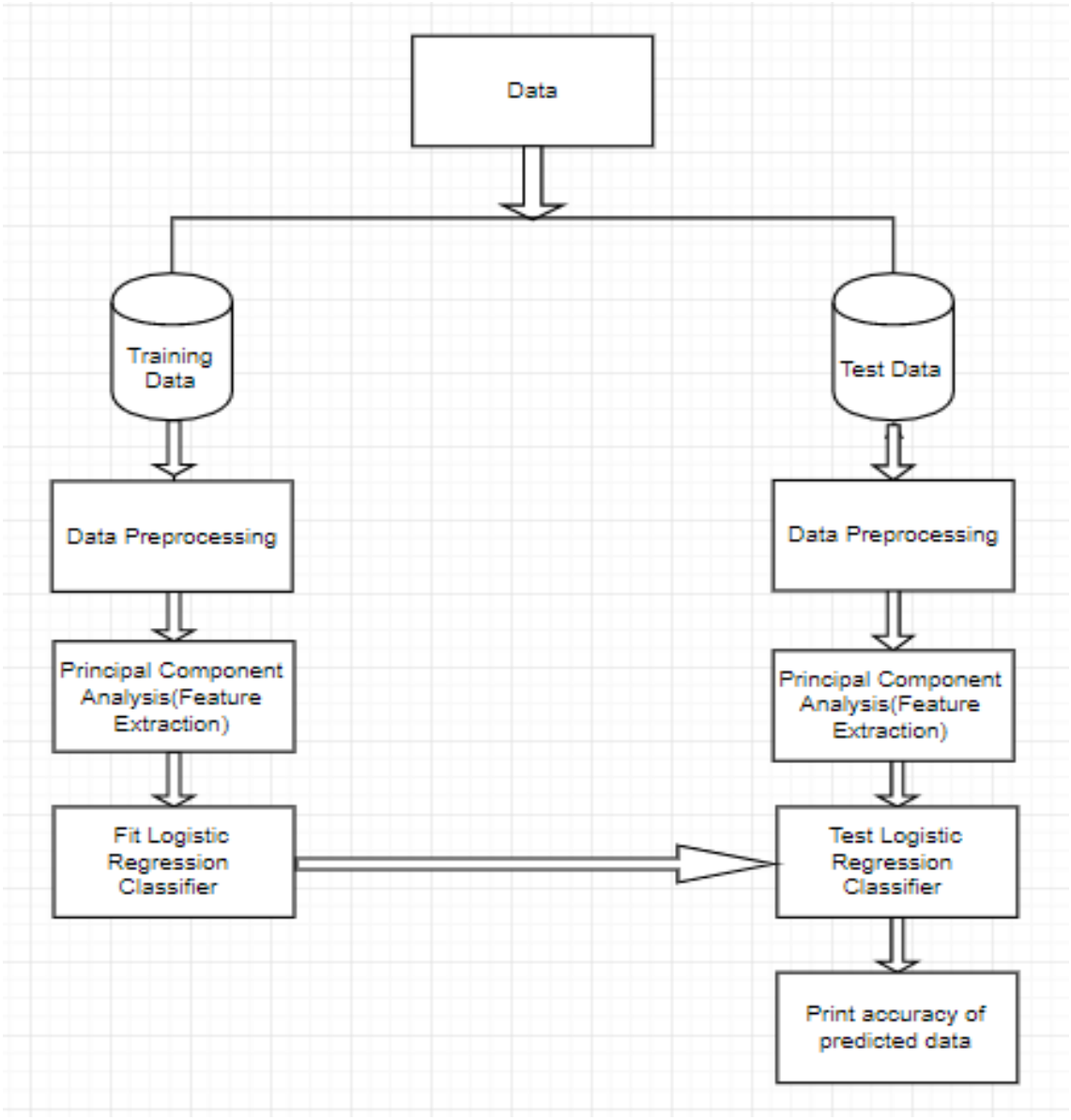


Figure 5.1: Data Flow Diagram of Logistic Regression Model

5.2. Data Flow Diagram of Random Forest Classifier Model

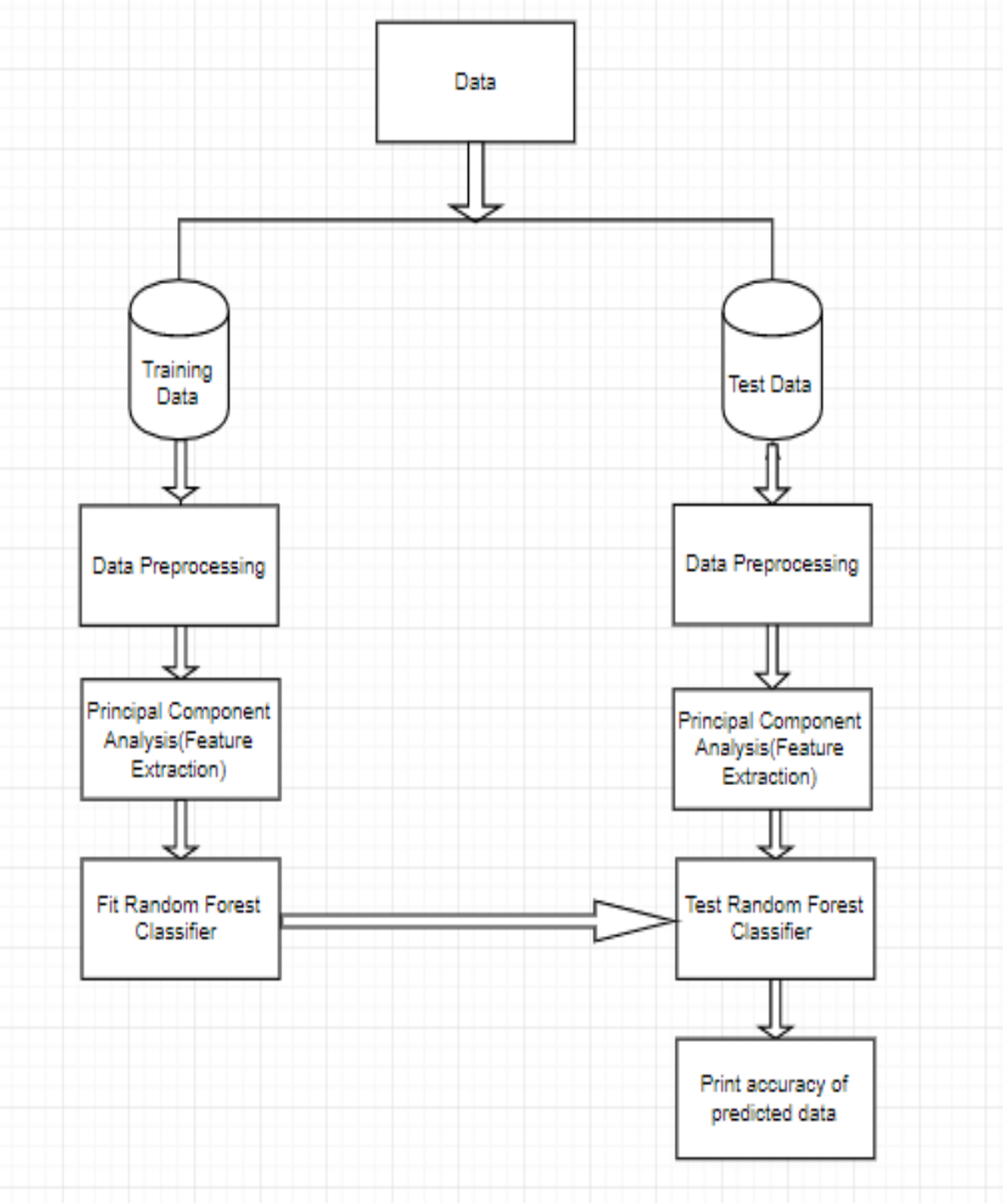


Figure 5.2: Data Flow Diagram of Random Forest Classifier Model

5.3. Data Flow Diagram of Support Vector Machine Classifier Model

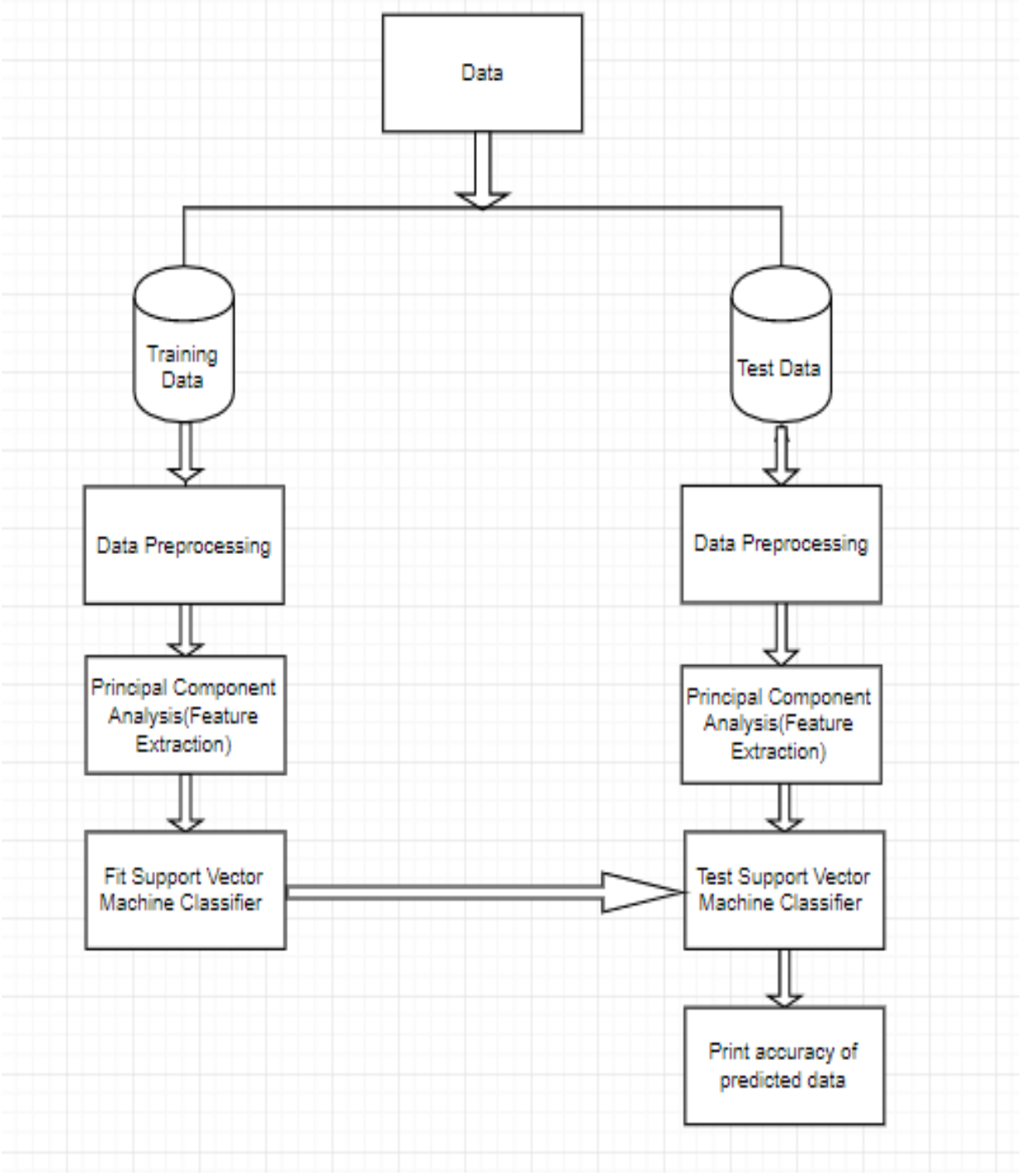


Figure 5.3: Data Flow Diagram of Support Vector Machine Model

5.4. Data Flow Diagram of Decision Tree Classifier Model

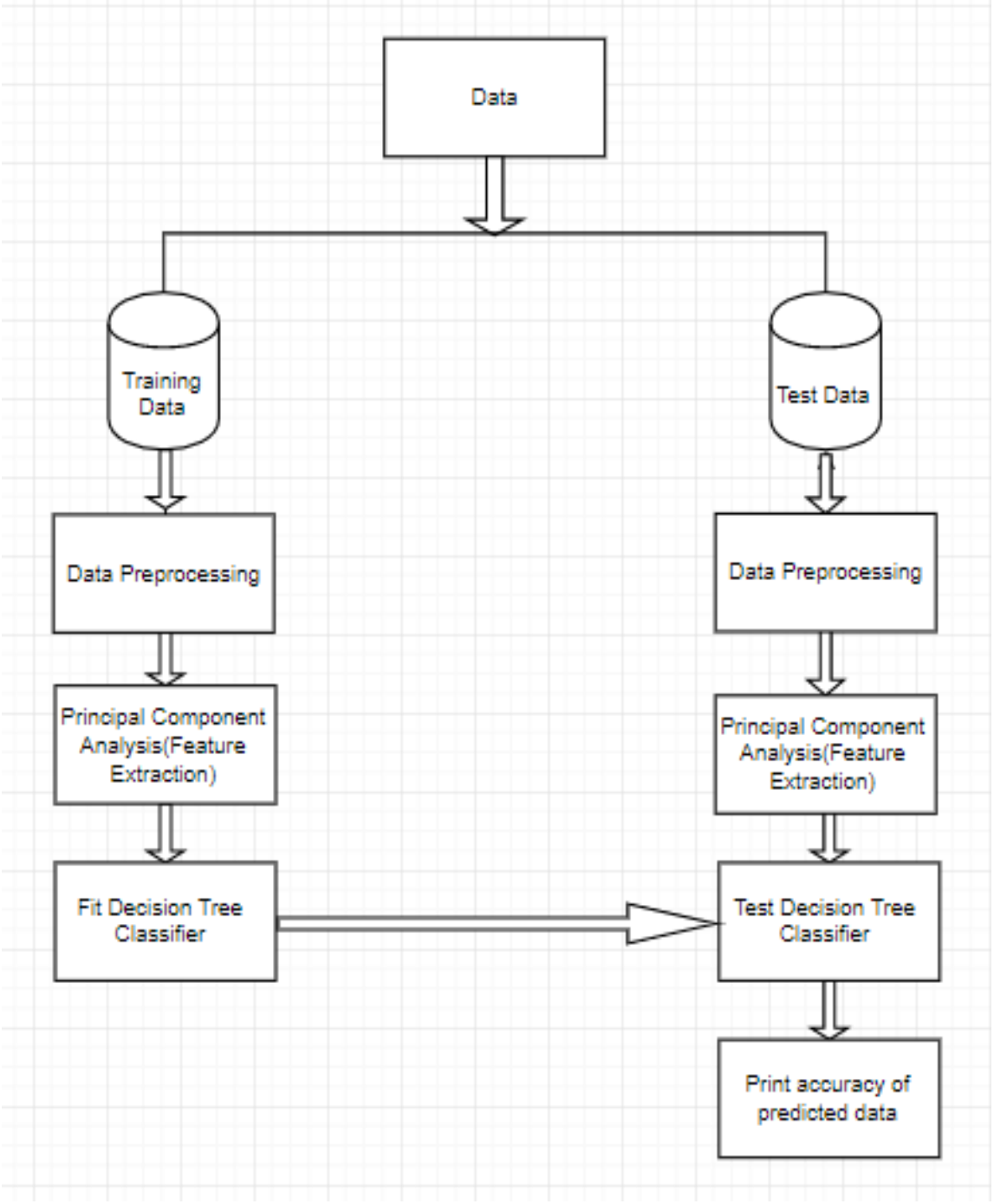


Figure 5.4: Data Flow Diagram of Decision Tree Classification Model

CHAPTER-6

CODE TEMPLATES

- Import the libraries

  Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Moreover Numpy forms the foundation of the Machine Learning stack.

  Pandas is hands down one of the best libraries of python. It supports reading and writing excel spreadsheets, CVS's and a whole lot of manipulation.

  Matplotib is a very powerful plotting library useful for those working with Python and NumPy. The most used module of Matplotib is Pyplot which provides an interface like MATLAB but instead, it uses Python and it is open source.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- Import the dataset named ‘train.csv’

```
# Loading the dataset
dataset = pd.read_csv('train.csv')
```

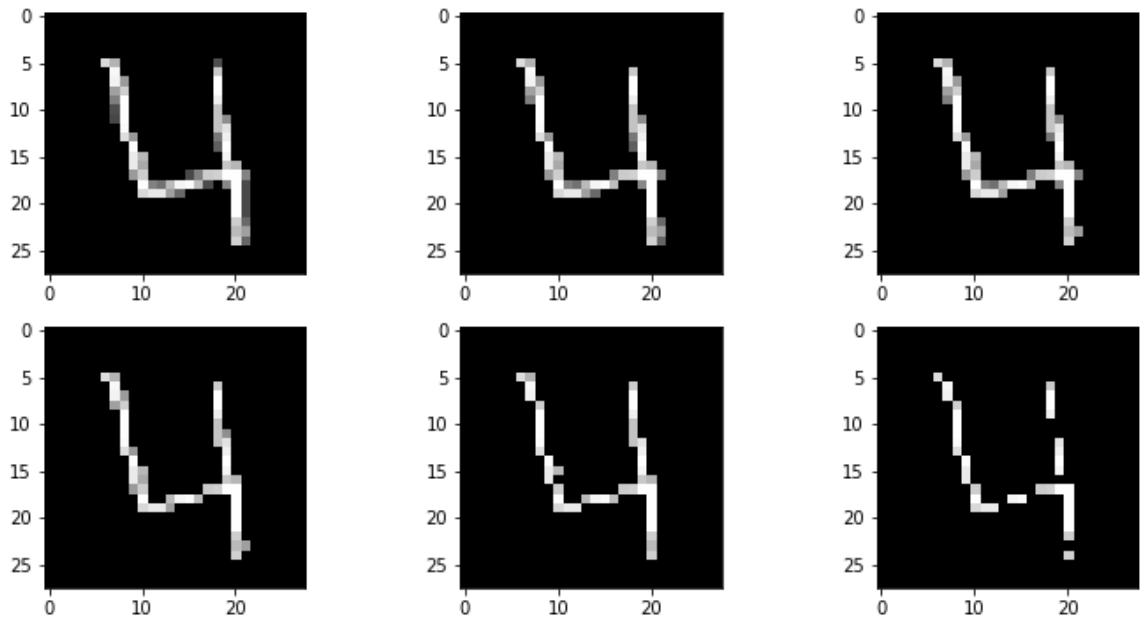
- ‘X’ contains the independent variables and ‘Y’ contains the dependent variable

```
X = dataset.iloc[:,1:]
y = dataset['label']
```

- In data preprocessing part, binarization approach is used. The binarization process assumes that images contain two classes of pixel: the foreground (or white pixels, with maximum intensity, i.e., equal to 1) and the background (or

black pixels with minimum intensity, i.e., equal to 0). The goal of the method is to classify all pixels with values above of the given threshold as white, and all other pixels as black. The different threshold values are tested to maximize the accuracy.

```
# Test done for differnt digits to find appropriate threshold
threshold = [70,90,110,140,170,200]
fig = plt.figure(figsize=(12,6))
row = 2
col = 3
plot_num = 1
for val in threshold:
    num = []
    digit = X.iloc[3,:]
    for pixel in digit:
        if pixel < val:
            pixel = 0
        num.append(pixel)
    img = np.array(num)
    img = np.reshape(img,newshape=(28,28))
    fig.add_subplot(row, col, plot_num)
    plot_num = plot_num + 1
    plt.imshow(img,cmap='gray')
plt.show()
```



According to the above output, the appropriate threshold is in between 90 and 110. Hence, the mean of 90 and 110 is set the threshold value i.e. 100

- The values above 100 are set to '1' and rest of the values are set to '0'

```
X = (np.array(X)[:,1:] >= 100).astype(int)
```

- The dataset is split into two parts. One is used to train the classifier and the other one is used to test the trained classifier

```
# Splittng the dataset
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.25,random_state = 42)
```

- Since the dataset contains 784 columns so to improve the efficiency of the classifier, Principle Component Analysis (PCA) method is used.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 33)
pca_train = pca.fit_transform(x_train)
pca_test = pca.transform(x_test)
```

- Logistic Regression model is trained using the train dataset and its accuracy is calculated using the test dataset

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(solver='saga',multi_class='multinomial')

classifier.fit(pca_train,y_train)
y_pred = classifier.predict(pca_test)
```

```
from sklearn.metrics import accuracy_score
print("Accuracy :",accuracy_score(y_test,y_pred))
```

- Random Forest Classifier is trained using the train dataset. Since, Random forest classifier contains many parameters hence Gridsearch CV is used to calculate the appropriate parameters to improve the efficiency of the classifier

```
from sklearn.model_selection import GridSearchCV
params =  {
            "n_estimators" : [10,50,100],
            "max_features" : ["auto", "log2", "sqrt"],
            "bootstrap"    : [True, False]
        }
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier()

clas = GridSearchCV(classifier, params, n_jobs=-1, cv=5, verbose=1, scoring='accuracy')
clas.fit(pca_train, y_train)

clas.best_estimator_

classifier = RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini'
            max_depth=None, max_features='log2', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)

classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)

from sklearn.metrics import accuracy_score
print("Accuracy :",accuracy_score(y_test,y_pred))
```

- Support Vector Machine Classifier is trained using the train dataset. Since, Support Vector Machine Classifier classifier contains many parameters hence Gridsearch CV is used to calculate the appropriate parameters to improve the efficiency of the classifier

```
from sklearn.model_selection import GridSearchCV
params = {"C":[0.1,1.0,10]
          }
from sklearn.svm import SVC
classifier = SVC()

clas = GridSearchCV(classifier, params, n_jobs=-1, cv=5, verbose=1, scoring='accuracy')
clas.fit(pca_train, y_train)

clas.best_estimator_

classifier = SVC(C=30, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.01, verbose=False)

classifier.fit(pca_train,y_train)
y_pred = classifier.predict(pca_test)

from sklearn.metrics import accuracy_score
print("Accuracy :",accuracy_score(y_test,y_pred))
```

- Decision Tree Classification model is trained using the train dataset and its accuracy is calculated using the test dataset

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()

classifier.fit(pca_train,y_train)
y_pred = classifier.predict(pca_test)
```

```
from sklearn.metrics import accuracy_score
print("Accuracy :",accuracy_score(y_test,y_pred))
```

- Confusion matrix is calculated using the following code:

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(y_test,y_pred)
```

- Classification Report is used to calculate precision, recall and f1 score for Logistic Regression, Random Forest Classifier and Support Vector Machine Classifier.

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

CHAPTER-7

RESULTS AND DECLARATION

The accuracies of the algorithms Logistic Regression, Random Forest Classifier, Support Vector Machine Classifier and Decision Tree Classifier are tabulated below in table 7.1.

Algorithm	**Data Accuracy**
Logistic Regression	89.37%
Random Forest Classifier	96.76%
Support Vector Machine Classifier	98.15%
Decision Tree Classifier	82.58%

Table 7.1: Accuracy Percentage of Algorithms

Screenshots of the accuracy:

Logistic Regression Model

```
Accuracy : 0.8937142857142857
```

Random Forest Classifier

```
Accuracy : 0.9676190476190476
```

Support Vector Machine Classifier

```
Accuracy : 0.9815238095238096
```

Decision Tree Classifier

```
Accuracy : 0.8258095238095238
```

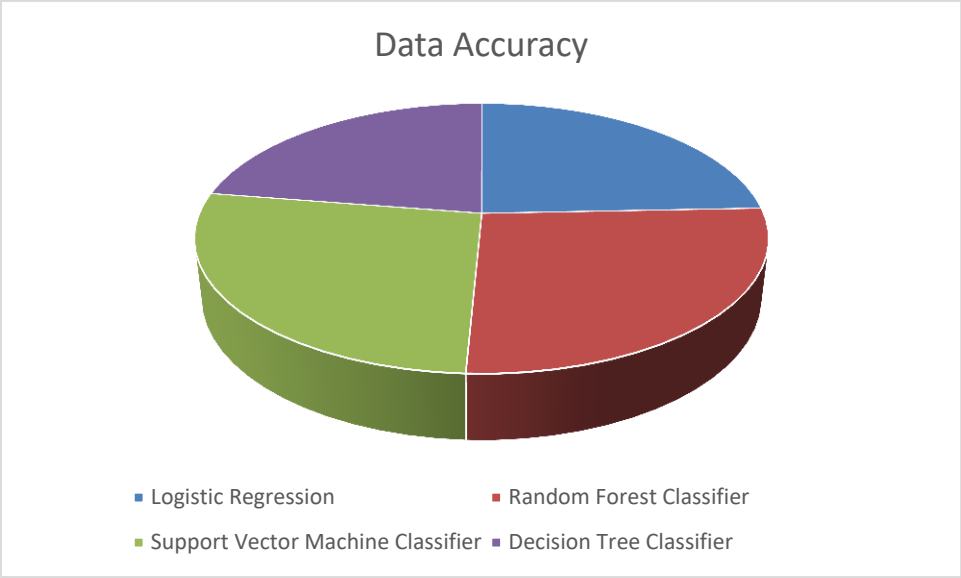


Fig 7.1: Pie chart representing accuracy of different models

It can be clearly observed that Support Vector Machine Classifier has more accuracy compared to Logistic Regression and Random Forest Classifier.

CHAPTER-8

CONCLUSION

In this project, the performances of the algorithms like Support Vector Machine Classifier, Logistic Regression, Decision Tree Classifier and Random Forest Classifier are analysed and compared. Using Binarization, Support Vector Machine Classifier has achieved an accuracy of 98.15%. Similarly, it is observed that Random Forest Classifier has yielded the least accuracy of about 96.76%. The Logistic Regression classifier has achieved the accuracy of 89.37%. The Decision Tree classifier has achieved the accuracy of 82.58%. Therefore, it can be concluded that Support Vector Machine Classifier tends to give better performance for this handwritten digit recognition process.

ANNEXURE: RESEARCH PAPER

Handwritten Digit Classification using Machine Learning models

Vidushi Garg (07914803116), Vandana Choudhary (Assistant Professor)

Maharaja Agrasen Institute of Technology, Rohini, New Delhi-86

Abstract: The rapid growth of new documents and multimedia news has created new challenges in pattern recognition and machine learning. The problem of handwritten digit recognition has long been an open problem in the field of pattern classification. Handwriting recognition is one of the compelling research works going on because every individual in this world has their own style of writing. It is the capability of the computer to identify and understand handwritten digits or characters automatically. The main objective of this paper is to provide efficient and reliable techniques for recognition of handwritten numerals by comparing various classification models. MNIST data set is widely used for this recognition process and it has 70000 handwritten digits. This paper performs the analysis of accuracies and performance measures of algorithms Support Vector Classification model (SVC), Logistic Regression, Decision Tree Classification model and Random Forest Classification model.

I. INTRODUCTION

Handwritten digit recognition is an important problem in optical character recognition, and it can be used as a test case for theories of pattern recognition and machine learning algorithms. To promote research of machine learning and pattern recognition, several standard databases have emerged. Handwriting recognition is one of the compelling and fascinating works because every individual in this world has their own style of writing. The main difficulty of handwritten numerals recognition is the serious variance in size, translation, stroke thickness, rotation and deformation of the numeral image because of handwritten digits are written by different users and their writing style is different from one user to another user.[4] In real-time applications like the conversion of handwritten information into digital format, postal code recognition, bank check processing, verification of signatures, this recognition is required.

This research aims to recognize the handwritten digits by using tools from Machine Learning to train the classifiers, so it produces a high recognition performance.[3] The MNIST data set is widely used for this recognition process. The MNIST data set has 70,000 handwritten digits. Each image in this data set is represented as an array of 28x28. The array has 784 pixels having values ranging from 0 to 255. If the pixel value is ‘0’ it indicates that the background is black and if it is ‘1’ the background is white.

This study focuses on feature extraction and classification. The performance of a classifier can rely as much on the quality of the features as on the classifier itself. In this study, we compare the performance of four different machine learning classifiers for recognition of digits. The four classifiers namely Support Vector Classification model (SVC), Logistic Regression, Decision Tree Classification model and Random Forest Classification model.[8] The main purpose of this research is to

build a reliable method for the recognition of handwritten digit strings. The main contribution in this work is that Support Vector Classification model gives the highest accuracy while compared to the other classification models.[5] Yet 100% accuracy is something that is to be achieved and the research is still actively going on in order to reduce the error rate. The accuracy and correctness are very crucial in handwritten digit recognition applications. Even 1% error may lead to inappropriate results in real-time applications.

II. RESEARCH METHODOLOGY

1. Description of the dataset

The MNIST dataset, a subset of a larger set NIST, is a database of 70,000 handwritten digits, divided into 60,000 training examples and 10,000 testing samples. The images in the MNIST dataset are present in form of an array consisting of 28x28 values representing an image along with their labels. This is also the same in case of the testing images. The graylevel values of each pixel are coded in this work in the [0,255] interval, using a 0 value for white pixels and 255 for black ones.

Digits	# Training	# Testing	Subtotal
9	5949	1009	6958
8	5851	974	6825
7	6265	1028	7293
6	5918	958	6876
5	5421	892	6313
4	5842	982	6824
3	6131	1010	7141
2	5958	1032	6990
1	6742	1135	7877
0	5923	980	6903
Total	60,000	10,000	70,000

Table 1: Dataset

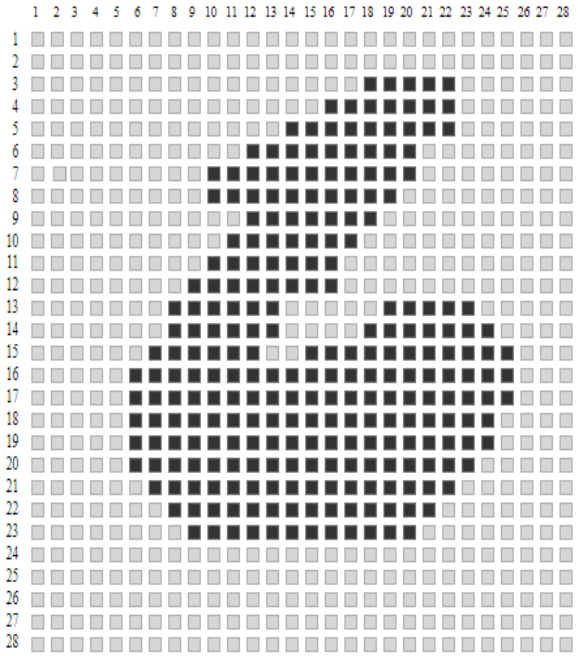


Figure 1: Digit 6 in 28x28 format in MNIST

2. Data Preprocessing

An important point for managing a high performance in the learning process is the construction of a useful training set. The 60000 different patterns contained in the MNIST database can be seen as a rather generous set, but evidence shows that the usual learning algorithms run into serious trouble for about one hundred or more of the test set samples. Therefore, some strategy is needed in order to increase the training set cardinality and variability. Usual actions comprise geometric transformations such as displacements, rotation, scaling and other distortions.

The variables proposed in this paper to make handwritten digit classification require images in binary binary level. The binarization process assumes that images contain two classes of pixel: the foreground (or white pixels, with maximum intensity, i.e., equal to 1) and the background (or black pixels with minimum intensity, i.e., equal to 0).[6] The goal of the method is to classify all pixels with values above of the

given threshold as white, and all other pixels as black. That is, given a threshold value t and an image X with pixels denoted as x(i, j), the binarized image X_b with elements $x_b(i, j)$ is obtained as follows:

If x(i, j) > t $\quad x_b(i, j) = 1$

Else $\quad x_b(i, j) = 0$

Then, the key problem in the binarization is how to select the correct threshold, t, for a given image. We observe that the shape of any object in the image is sensitive to variations in the threshold value, and even more sensitive in the case of handwritten digit. Therefore, we consider that a binary handwritten number is
better recognized computationally if its trace is complete and continuous, this is the criterion that we use to the threshold, being its choice of crucial importance.

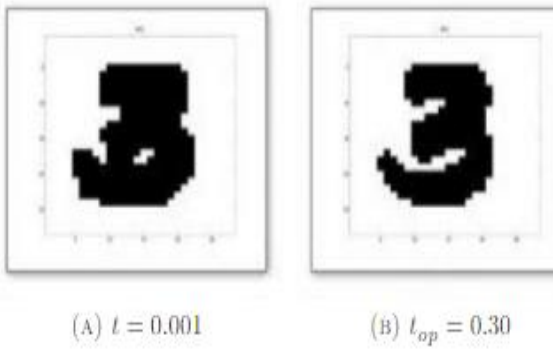


Figure 2: The same digit with different threshold value

3. Implementation

- **Logistic Regression Model**

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

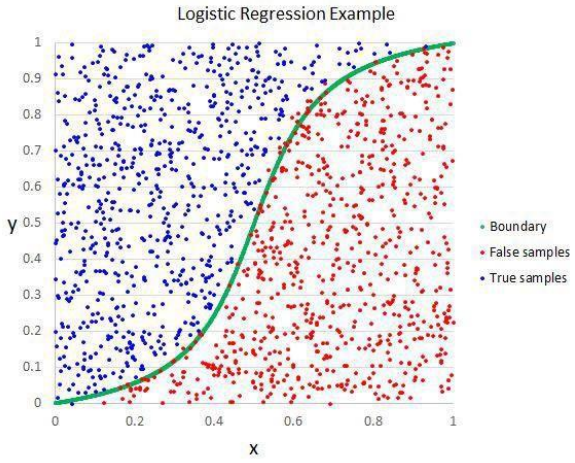


Figure 3: Logistic Regression Model

After that training set data is fed as input to the Logistic Regression model so that the classifier gets trained. The guessed label is matched with the original to get the accuracy of the trained classifier. Once the training is done, the testing data is given to the classifier to predict the labels and testing accuracy is obtained.[5] The confusion matrix is generated which gives the probability between the actual data and the predicted data. Using the confusion matrix, the performance measures like precision, recall and f1 score can be calculated. Using Logistic Regression, an accuracy of 88.97% is obtained on test data.

Precision = TP/(TP+FP)

where TP = True Positive, FP = False Positive

Recall = TP/(TP+FN)

where FN = False Negative

F1score = 2*Precision+Recall/(Precission+Recall)

The confusion matrix obtained here is in the form of matrix M 10x10 because it is a multi-class classification (0-9).

The below Table 2 is the confusion matrix obtained for the trained data set using Logistic Regression Model.

Digit	0	1	2	3	4	5	6	7	8
0	985	0	3	0	3	19	11	1	2
1	0	1115	2	0	1	5	1	3	19
2	12	20	916	19	32	7	22	16	25
3	9	10	35	954	2	57	10	18	26
4	2	5	3	1	944	2	12	5	6
5	17	9	11	31	30	737	25	9	17
6	15	6	7	2	18	10	944	0	7
7	1	15	24	2	25	4	0	1014	3
8	6	34	11	32	10	31	10	10	840
9	8	6	7	15	44	11	2	42	6

Table 2: Confusion matric using Logistic Regression model

The below table 5 shows the precision, recall and f1 score values obtained for the trained data set using the Logistic Regression Model.

Digit	Precision	Recall	f1-score
0	0.93	0.96	0.95
1	0.91	0.97	0.94
2	0.9	0.85	0.88
3	0.9	0.83	0.86
4	0.85	0.92	0.89
5	0.83	0.82	0.83
6	0.91	0.93	0.92
7	0.91	0.89	0.9
8	0.88	0.84	0.86
9	0.85	0.86	0.86

Table 3: Precision, Recall and F1 score for Logistic Regression on trained dataset

- **Random Forest Classifier**

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

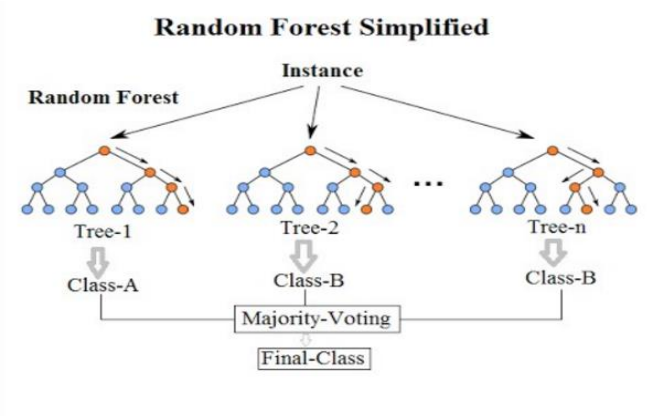


Figure 5: Random Forest Classifier

Here also, the confusion matrix is obtained and precision and recall values are computed.

Digit	0	1	2	3	4	5	6	7	8	9
0	1011	0	2	1	2	1	4	0	4	0
1	0	1135	5	1	1	1	2	1	0	0
2	4	5	1028	5	10	1	6	5	7	1
3	3	1	10	1090	2	11	2	11	11	10
4	2	0	1	0	995	0	7	1	0	18
5	3	2	2	12	0	851	9	2	9	8
6	8	2	0	0	3	8	986	0	3	0
7	1	7	16	2	4	1	0	1080	1	23
8	1	5	2	9	4	10	5	2	964	3
9	5	4	2	15	14	4	2	8	5	975

Table 4: Confusion matrix using Random Forest Classifier

The above Table 4 shows the confusion matrix obtained for the trained data set using the Random Forest Classifier.

The below table 5 shows the precision, recall and f1 score values obtained for the trained data set using the Random Forest Classifier.

Digit	Precision	Recall	f1-score
0	0.97	0.99	0.98
1	0.98	0.99	0.98
2	0.96	0.96	0.96
3	0.96	0.95	0.95
4	0.96	0.97	0.97
5	0.96	0.95	0.95
6	0.96	0.98	0.97
7	0.97	0.95	0.96
8	0.96	0.96	0.96
9	0.94	0.94	0.94

Table 5: Precision, Recall and F1 score for Random Forest Classifier on trained dataset

The Test Data Set obtained accuracy of 96.3% using the Random Forest Classifier on MNIST data set.

- **Support Vector Machine (SVM) Classifier for digit recognition**

Support Vector Machine is a supervised machine learning technique which is applied for classification and regression. It is nothing but the representation of the input data into points in the space and mapped thus classifying them into classes.[7] The SVM classifies or separates the classes using the hyper-plane concept. The separation margin should be equidistant from the classes.

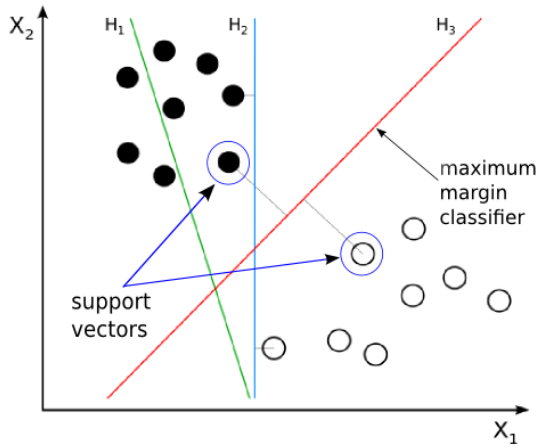


Figure 7: Support Vector Machine Classifier

Here also, the confusion matrix is obtained and precision and recall values are computed.

Digit	0	1	2	3	4	5	6	7	8	9
0	1018	0	2	0	0	0	3	0	0	2
1	1	1140	1	0	1	1	1	1	0	0
2	2	2	1054	3	1	1	0	6	3	0
3	2	1	4	1122	0	8	0	2	8	4
4	3	0	1	0	1000	0	5	3	1	11
5	1	1	0	7	0	879	6	0	1	3
6	3	1	0	0	0	2	1001	0	3	0
7	0	7	6	0	2	1	0	1109	1	9
8	0	2	2	4	3	3	1	1	988	1
9	4	1	1	6	8	6	2	7	4	995

Table 6: Confusion matrix using Support Vector Machine Classifier

The above Table 4 shows the confusion matrix obtained for the trained data set using the Support Vector Machine Classifier.

The below table 5 shows the precision, recall and f1 score values obtained for the trained data set using the Support Vector Machine Classifier.

Digit	Precision	Recall	f1-score
0	0.98	0.99	0.99
1	0.99	0.99	0.99
2	0.98	0.98	0.98
3	0.98	0.97	0.98
4	0.99	0.98	0.98
5	0.98	0.98	0.98
6	0.98	0.99	0.99
7	0.98	0.98	0.98
8	0.98	0.98	0.98
9	0.97	0.96	0.97

Table 7: Precision, Recall and F1 score for Support Vector Machine Classifier on trained dataset

The Test Data Set obtained accuracy of 98.3% using the Random Forest Classifier on MNIST data set.

4. Results and declaration

The accuracies of the algorithms Logistic Regression, Random Forest Classifier and Support Vector Machine Classifier are tabulated below in table 8.

Algorithm	Data Accuracy
Logistic Regression	88.97%
Random Forest Classifier	96.3%
Support Vector Machine Classifier	98.3%

It can be clearly observed that Support Vector Machine Classifier has more accuracy compared to Logistic Regression and Random Forest Classifier.

III. CONCLUSION

In this paper, the performances of the algorithms like Support Vector Machine Classifier, Logistic Regression and Random Forest Classifier are analysed and compared. Using Binarization, Support Vector Machine Classifier has achieved an accuracy of 98.3%. Similarly, it is observed that Random Forest Classifier has yielded the least accuracy of about 96.3%. The Logistic Regression classifier has achieved the accuracy of 88.97%. Therefore, it can be concluded that Support Vector Machine Classifier tends to give better performance for this handwritten digit recognition process.

IV. REFERENCES

[1] Wu, Ming & Zhang, Zhen. (2019). Handwritten Digit Classification using the MNIST Data Set.

[2] S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair" Handwritten Digit Recognition using Machine Learning Algorithms"

[3] Plamondon, R., & Srihari, S. N. Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1), 63-84

[4] Liu, C. L., Yin, F., Wang, D. H., & Wang, Q. F. (2013). Online and offline handwritten Chinese character recognition benchmarking on new databases. Pattern Recognition, 46(1), 155-162

[5] Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras

[6] Al-Wzwazy, Haider& M Albehadili, Hayder&Alwan, Younes& Islam, Naz& E Student, M &, Usa. (2016). Handwritten Digit Recognition Using Convolutional Neural Networks. International Journal of Innovative Research in Computer and Communication Engineering. 4. 1101-1106.

[7] AL-Mansoori, Saeed. (2015). Intelligent Handwritten Digit Recognition using Artificial Neural Network. 10.13140/RG.2.1.2466.0649

[8] A Comprehensive Data Analysis on Handwritten Digit Recognition using Machine Learning Approach

by Meer Zohra, D.Rajeswara Rao. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-6, April 20

REFERENCES

[1] Al-Wzwazy, Haider& M Albehadili, Hayder&Alwan, Younes& Islam, Naz& E Student, M &, Usa. (2016). Handwritten Digit Recognition Using Convolutional Neural Networks. International Journal of Innovative Research in Computer and Communication Engineering. 4. 1101-1106.

[2] AL-Mansoori, Saeed. (2015). Intelligent Handwritten Digit Recognition using Artificial Neural Network. 10.13140/RG.2.1.2466.0649

[3] Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras

[4] Liu, C. L., Yin, F., Wang, D. H., & Wang, Q. F. (2013). Online and offline handwritten Chinese character recognition benchmarking on new databases. Pattern Recognition, 46(1), 155-162

[5] A Comprehensive Data Analysis on Handwritten Digit Recognition using Machine Learning Approach by Meer Zohra, D.Rajeswara Rao. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-6, April 2019

[6] Plamondon, R., & Srihari, S. N. Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1), 63-84

[7] S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair" Handwritten Digit Recognition using Machine Learning Algorithms"

[8] Wu, Ming & Zhang, Zhen. (2019). Handwritten Digit Classification using the MNIST Data Set.