BUAN 6341 Applied Machine Learning **Date: 04/13/2020**
Assignment 3
Implementing learning Algorithms for classification
Prepared by: Kamini Bokefode

DATASETS

Two datasets have been used to evaluate the performance of classification algorithms. Details of the first dataset are as follows:
Dataset1: The data is related to the direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. Dataset has been downloaded from the following link. Only bank-full.csv has been used for this assignment.
http://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Dataset2: This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 241600 possible parameter combinations. For each tested combination, 4 runs were performed and their results are reported as the 4 last columns. All times are measured in milliseconds*. There are 14 parameter, the first 10 are ordinal and can only take up to 4 different powers of two values, and the 4 last variables are binary. Dataset can be downloaded from the following link :
https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance

MACHINE LEARNING ALGORITHM & EXPERIMENTS

The following Machine Learning Algorithm has been used for the classification problem for both datasets.

1. Artificial Neural Networks
   - Activation Function Relu (varying number of nodes and number of hidden layers)
   - Activation Function SIGMOID (varying number of nodes and number of hidden layers)
   - Activation Function TANH (varying number of nodes and number of hidden layers)
2. K- Nearest Neighbors
   - The varying number of neighbors
   - Metric: Euclidian Distance
   - Metric: Manhattan Distance

Artificial Neural Networks

To create a neural network, we simply begin to add layers of perceptron's together, creating a multi-layer perceptron model of a neural network. We will have an input layer that directly takes in data and an output layer which will create the resulting outputs. Any layers in between are known as hidden layers because they don't directly "see" the feature inputs within the data you feed in or the outputs.

K Nearest Neighbors

KNN (K-Nearest Neighbor) is a simple supervised classification algorithm we can use to assign a class to a new data point. It can be used for regression as well, KNN does not make any assumptions on the data distribution, hence it is non-parametric. It keeps all the training data to make future predictions by computing the similarity between an input sample and each training instance.

Packages used

Keras has been used to carry out the experiments of Neural Networks. It is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning **models**. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows us to define and train neural network models in just a few lines of code.
Scikit-learn is carefully organized into modules so that we can import the relevant classes easily. Importing the class 'KNeighborsClassifer' from the 'neighbors' module and Instantiate the estimator ('estimator' is scikit-learn's term for a model).

Data Preprocessing

We have split our data into training and testing sets, this is done easily with SciKit Learn's train_test_split function from model_selection. The training set is further divided into trained and testing set, this is done to use the first test set for model evaluation.

The neural network in Python may have difficulty converging before the maximum number of iterations allowed if the data is not normalized. Multi-layer Perceptron is sensitive to feature scaling, so it is highly recommended to scale the data. We have used the built-in preprocessing method of Scikit for the standardization of data for both the algorithms.

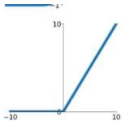
Evaluation Metrics:

Accuracy has been used to evaluate the performance of the different models. It is the ratio of the number of correct predictions to the total number of input samples.

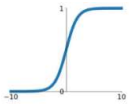
Artificial Neural Networks: Activation Functions:

Mathematical and pictorial representation of the activation functions used in this experiments are as follows:

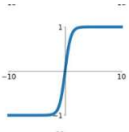
ReLU
$\max(0, x)$



Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



tanh
$\tanh(x)$



ReLU is the most commonly used Activation Functions, because of its simplicity during backpropagation and its not computationally expensive.

Sigmoid Function is one of the special functions in the Deep Learning Field, because of its simplification during Back Propagation.

Tanh can be considered as a good example in case when input>0, so the gradients we will obtain will either be all positive or negative, which can lead to an explosion or vanishing issue, thus usage of tanh can be a good thing, but this still faces the problem of Saturated Gradients

EXPERIMENTS

All the experiments carried out in this section were performed by varying the following three parameters.

No of Nodes used : 3, 5, 10
No of Hidden Layers: 1, 2, 3
No of Iterations: 10, 100, 500

While experimenting with hidden layers, we first started with hidden layer1 and tried to identify the number of nodes that would give the best accuracy and least difference in train and test accuracy.

We then fixed those many numbers of nodes in layer 1 when experimenting with the best number of nodes in layer 2.

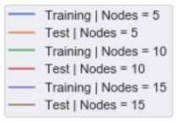
Likewise a number of nodes in layer 1 and layer 2 are fixed based on the previous two experiments and for the third layer, the number of nodes and no of iterations is varied to get the accuracy for both train and test data.

Artificial Neural Networks: DATASET 1 (Bank Data)

As the neural networks as taking a considerable amount of time to learn the data, few experiments were carried out to identify the ideal number of sample sizes for carrying out all the experiments.

Activation Function	nodes in 1st hidden layer	iterations	Sample size
RELU	10, 20, 30	10 to 500	50000, 2000, 5000, 15000

The results were as follows. It can be seen that the sample performance of neural networks with sample size 5000 and 2000 have underperformed with test accuracy being higher than training accuracy for the different number of nodes in layer 1. Whereas the outcome of the experiment with the sample size of 15000 seems relatively better in terms of the difference in the accuracy of training and test data. Also, the run time of all the experiments with a sample size of 15000 is considerable fine.



We, therefore, select a sample size of 15000 for all the experiments.

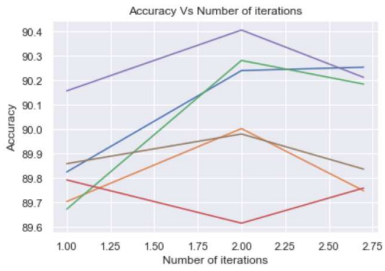


Fig1 : Sample size = 50000

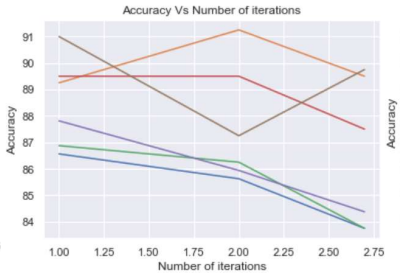


Fig1 : Sample size = 2000

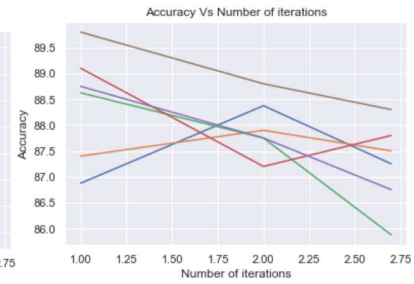


Fig1 : Sample size = 5000

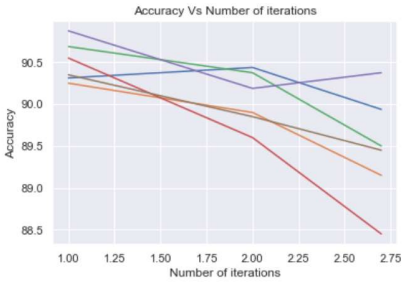
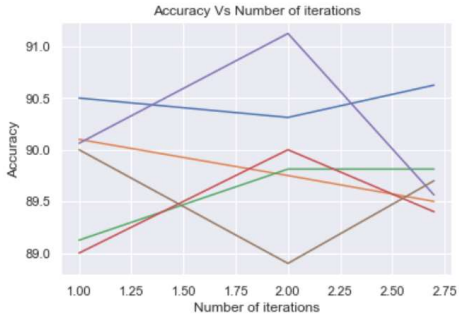
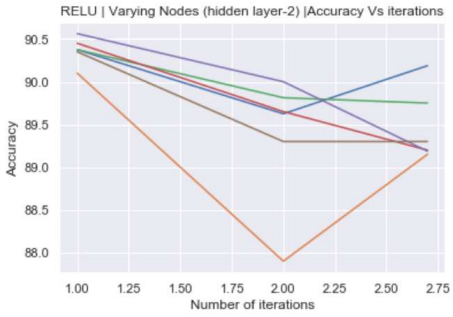
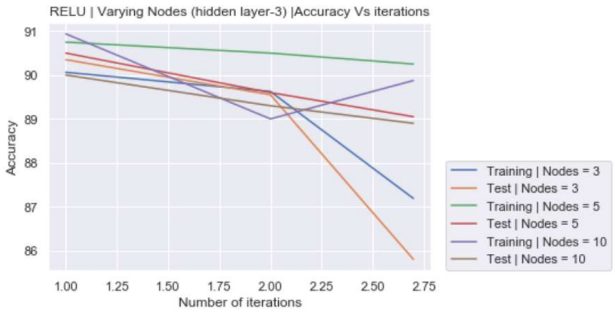


Fig1 : Sample size = 15000

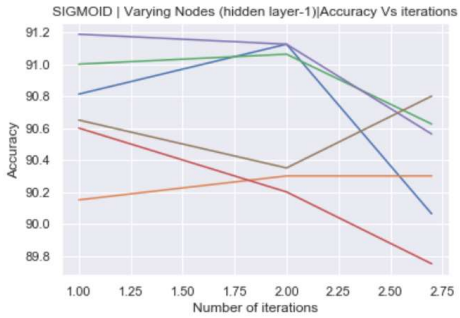
RELU Activation Function

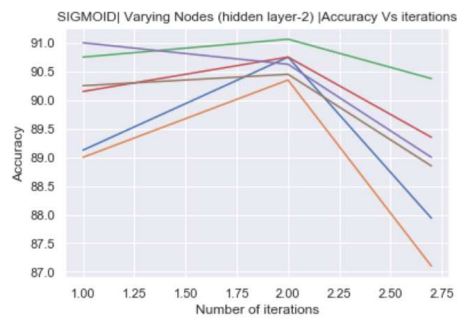


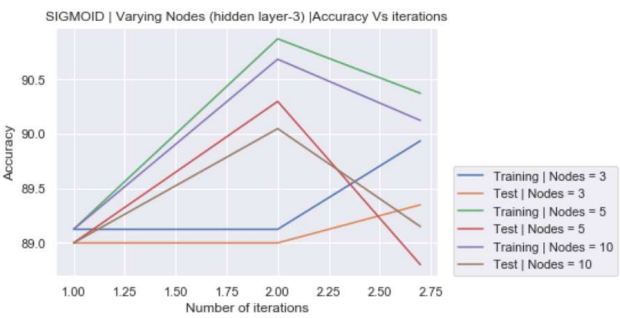




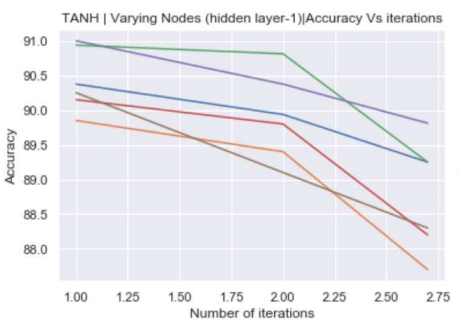
SIGMOID Activation Function

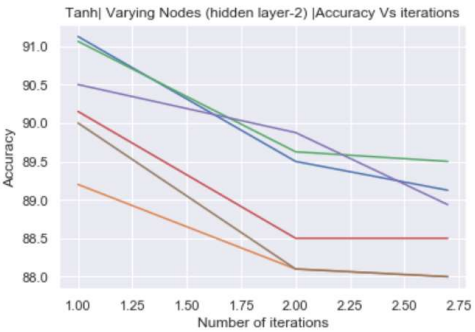


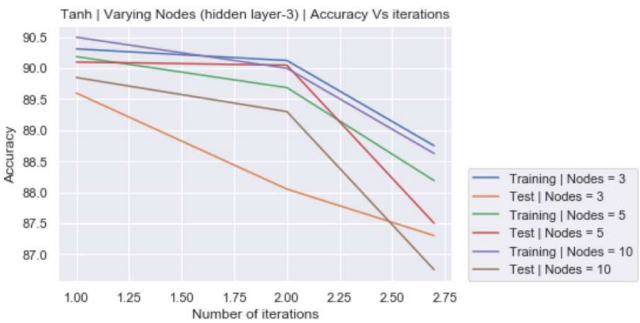




TANH Activation Function







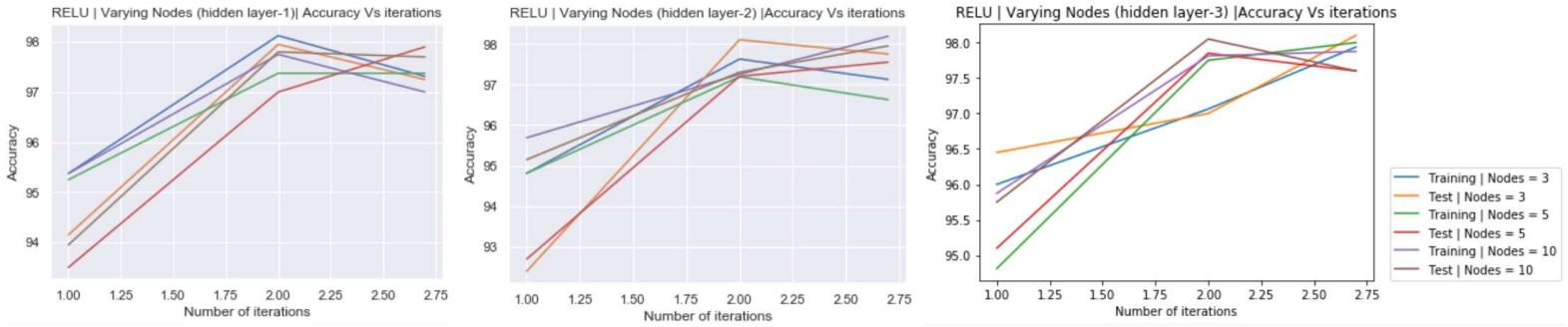
The summary of the experiments performed is represented below in the tabular form. In total 9 experiments were performed, 3 for each activation function.

Activation Function	No. of nodes in layer 1	No. of nodes in layer 2	No. of nodes in layer 3	Iterations	Accuracy_Train	Accuracy_Test	Difference
Relu	5	0	0	100	90	89.65	0.35
Relu	5	5	0	10	90.49	90.48	0.01
Relu	5	5	10	10	91	90	1
SIGMOID	5	0	0	10	91	90.6	0.4
SIGMOID	5	5	0	100	91	90.55	0.45
SIGMOID	5	5	3	100	89.2	89	0.2
TANH	5	0	0	10	90.9	90.2	0.7
TANH	**5**	**5**	**0**	**10**	**91.2**	**90.2**	**1**
TANH	5	5	5	10	90.2	90.1	0.1

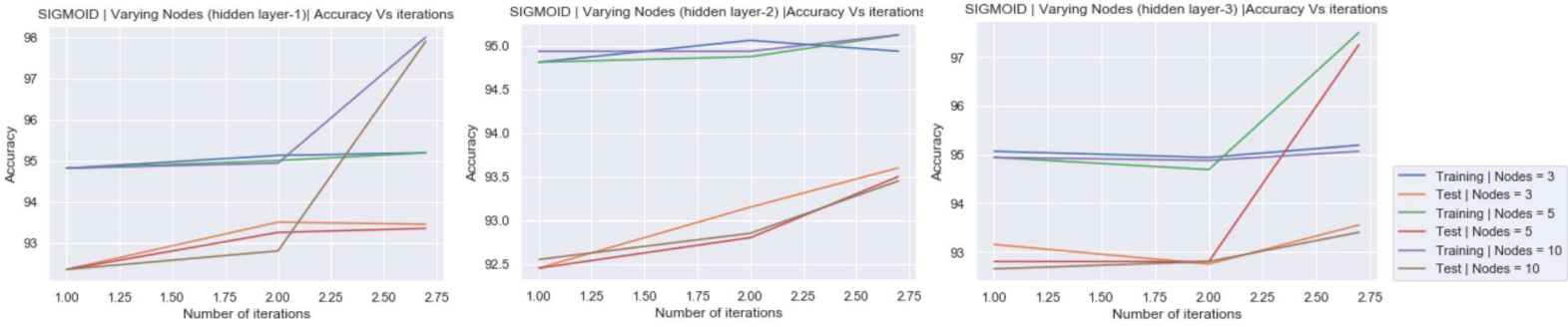
Experiments with the least number of iterations, higher Training & test accuracy and also the least difference between these two accuracies can be considered as a good choice of model.
It can be clearly seen that of all the models, the model with **Tanh** as activation function, with **two hidden layers** each having **5 nodes** has given the **best performance** in terms of test and training data.

Artificial Neural Networks : DATASET 2 (GPU Run time)

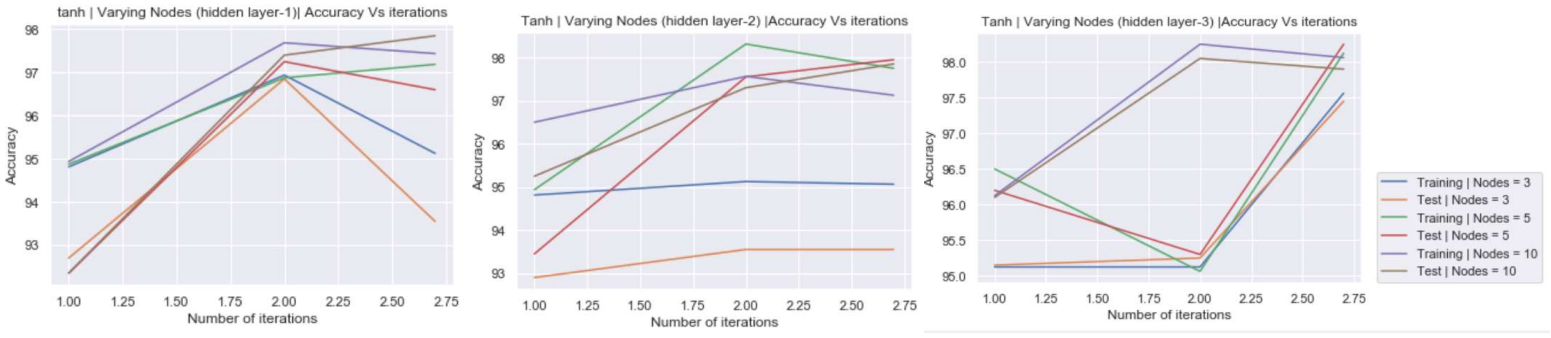
RELU Activation Function



SIGMOID Activation Function



TANH Activation Function



Experiments with the least number of iterations, higher Training & test accuracy and also the least difference between these two accuracies can be considered as a good choice of model.
The summary of the experiments performed is represented below in the tabular form. In total 9 experiments were performed, 3 for each activation function.

Activation Function	No. of nodes in layer 1	No. of nodes in layer 2	No. of nodes in layer 3	Iterations	Accuracy_Train	Accuracy_Test	Difference
Relu	**3**	**0**	**0**	**100**	**98.1**	**98**	**0.1**
Relu	3	3	0	100	98	97.5	0.5
Relu	3	3	10	100	98	97.97	0.03
SIGMOID	10	0	0	100	98	97.89	0.11
SIGMOID	3	10	0	100	95	93.5	1.5
SIGMOID	3	10	5	100	97.5	97.2	0.3
TANH	10	0	0	100	97.6	97.5	0.1
TANH	10	5	0	100	98.3	97.5	0.8
TANH	10	5	10	100	98.2	98	0.2

It can be clearly seen that of all the models, the model with **Relu** as activation function, with **1 hidden layer** with having **3 nodes** has given the **best performance** in terms of test and training data. It can see a model with Tanh as activation function, with three hidden layers that have given similar performance however it is computationally more expensive than the selected model.

K – Nearest Neighbors: Metric

K-NN and K-Means depend upon the distance between two data points to predict the output. Therefore, the metric we use to compute distances plays an important role in these models.
Minkowski distance is a generalized distance metric. We can manipulate the above formula by substituting 'p' to calculate the distance between two data points in different ways. Thus, Minkowski Distance is also known as Lp norm distance.
Some common values of 'p' are:-p = 1, Manhattan Distance, p = 2, Euclidean Distance

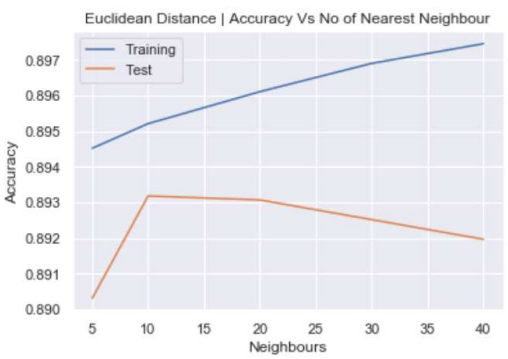
Precision(TP/TP+FP): Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false-positive rate.
Recall(TP/TP+FN): Recall is the ratio of correctly predicted positive observations to all observations in actual class – yes

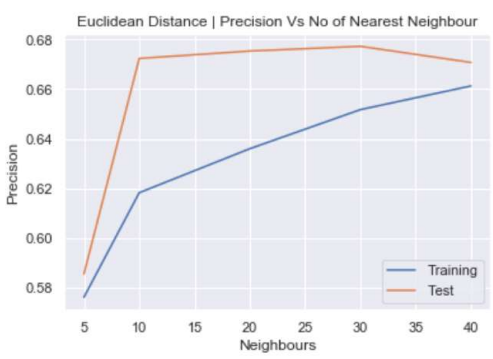
EXPERIMENTS

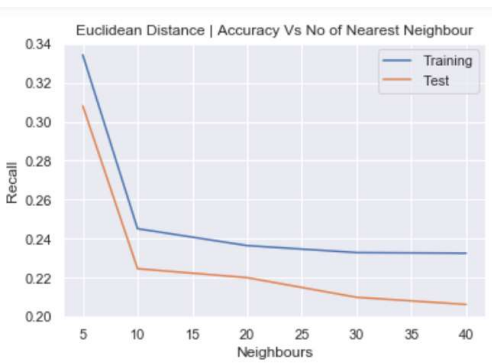
Experiments have been carried out by changing the distance metric and number of nearest neighbors(k) for both the datasets. Accuracy, precision and recall has been plotted with varying number of K for each metric to identify the best model for this algorithm

K – Nearest Neighbors : DATASET 1 (Bank Data)

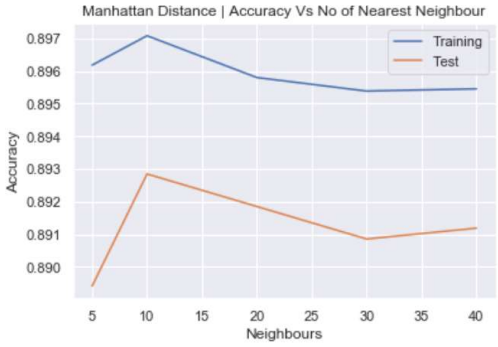
Metric : Euclidean

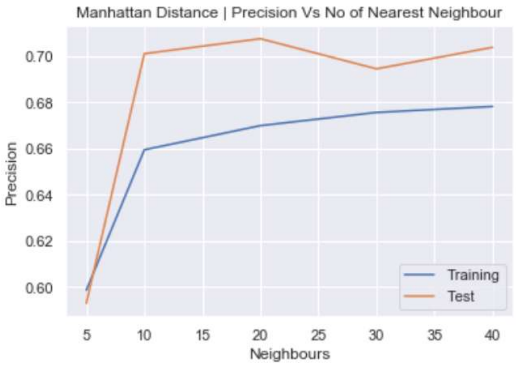


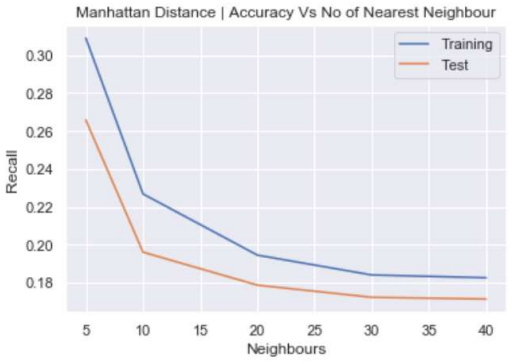




Metric : Manhattan







The summary of the experiments performed is represented below in the tabular form. In total 2 experiments were performed, 1 for each metric and varying the number of nearest neighbors from the set of [5,10,20,30,40]. Of given k values, outputs from k=5,10,40 seem to have given a significant contribution in making the decision for the optimum value of nearest neighbors.

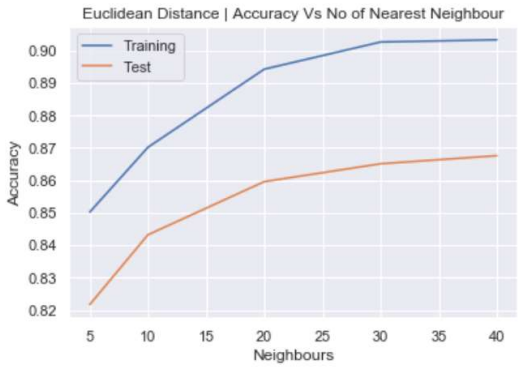
Metric	K	Accuracy_Training	Accuracy_Test	Precision_Training	Precision_Test	Recall_Training	Recall_Test
Euclidean	**5**	**0.8905**	**0.8945**	**0.583**	**0.58**	**0.328**	**0.31**
Euclidean	10	0.895	0.893	0.67	0.62	0.24	0.22
Euclidean	40	0.897	0.892	0.66	0.67	0.23	0.21
Manhattan	5	0.896	0.885	0.6	0.6	0.33	0.263
Manhattan	10	0.897	0.893	0.66	0.7	0.23	0.2
Manhattan	**40**	**0.902**	**0.868**	**0.892**	**0.85**	**0.95**	**0.945**

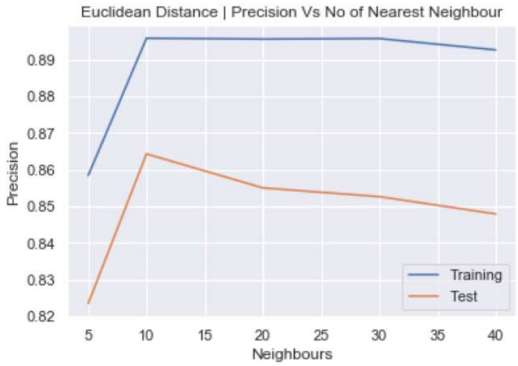
We select the last model from the above table with metric = Manhattan distance, K=40 which gives test accuracy of 86.8%, precision of 85% and recall of 94%.

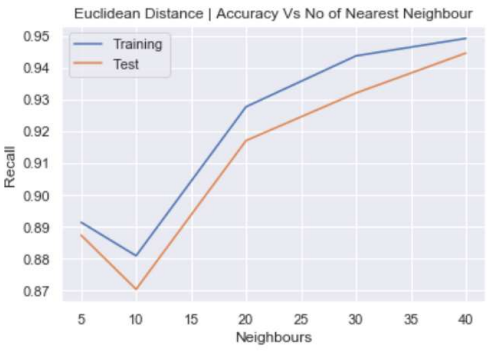
It can be seen that though the Euclidean distance has given the highest accuracy for test data with the value of K = 5, the model with Manhattan distance as metric has outperformed in terms of precision and recall. Also, the difference in the accuracy of the two models is less than that of 4% where are precision and recall significantly high in the chosen model.

K – Nearest Neighbors: DATASET 2 (GPU Run time)

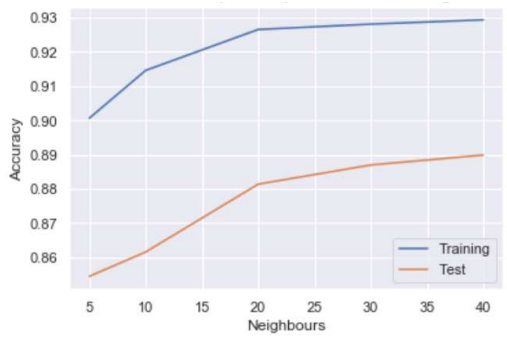
Metric: Euclidean

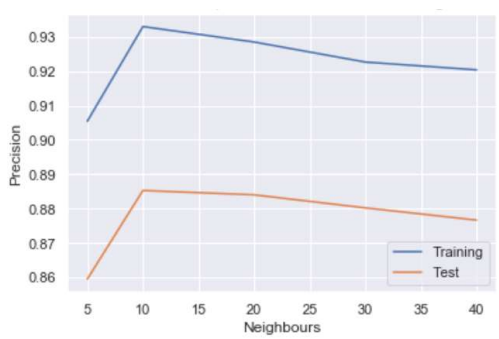


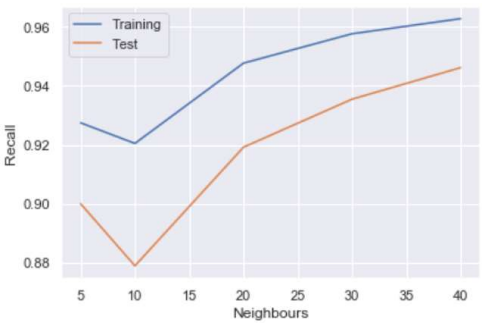




Metric: Manhattan







The summary of the experiments performed is represented below in the tabular form. In total 2 experiments were performed, 1 for each metric and varying the number of nearest neighbors from the set of [5,10,20,30,40]. Of given k values, outputs from k=5,10,40 seem to have given a significant contribution in making the decision for the optimum value of nearest neighbors.

Metric	K	Accuracy_Training	Accuracy_Test	Precision_Training	Precision_Test	Recall_Training	Recall_Test
Euclidean	5	0.85	0.822	0.86	0.821	0.89	0.888
Euclidean	10	0.87	0.845	0.896	0.865	0.88	0.87
Euclidean	40	0.92	0.868	0.892	0.848	**0.95**	**0.945**
Manhattan	5	0.901	0.855	0.905	0.86	0.93	0.9
Manhattan	10	**0.915**	**0.862**	**0.932**	**0.885**	0.92	0.88
Manhattan	**40**	**0.93**	**0.89**	**0.92**	**0.878**	**0.961**	**0.942**

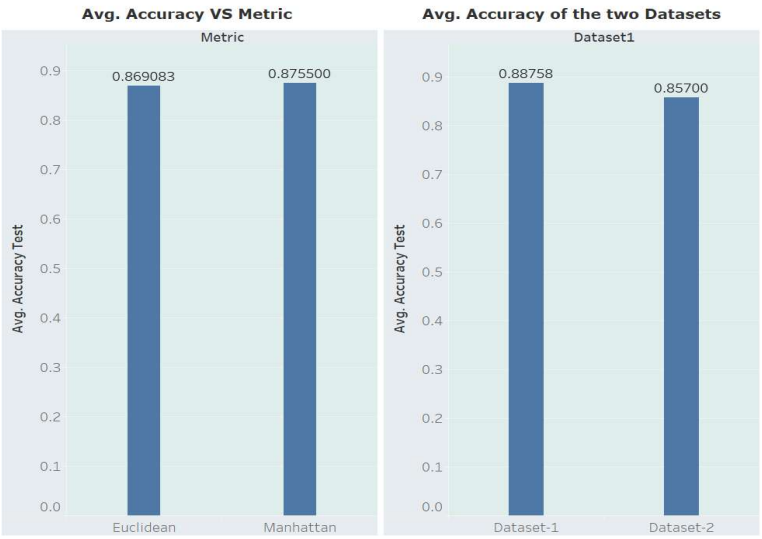
We select the last model from the above table with metric = Manhattan distance, K=40 which gives test accuracy of 89%, precision of 87.8% and recall of 94.2%.
It can be seen that the Euclidean distance has given a similar performance on test data with the value of K = 40, however, the model with Manhattan distance as metric has given slightly better performance in all the parameters.

Model Evaluation:

Performance analysis of all the algorithms:

- Here is a quick review of the performance of two datasets, and the activation functions used in the ANN algorithm.

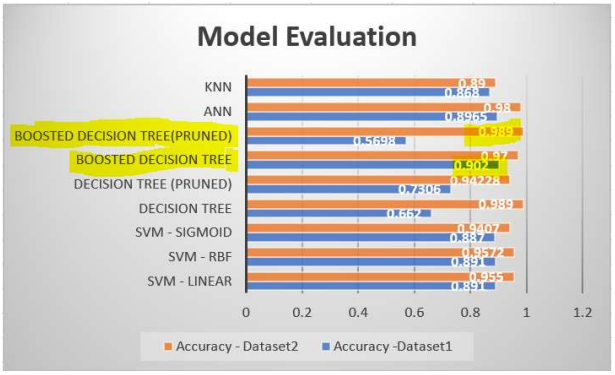
- Here is a quick review of the performance of two datasets, and the activation functions used in the ANN algorithm.



Best algorithm?

- For Dataset 1, Boosted Decision Tree(pruned) and ANN both can be considered as best algorithms for classification as the difference in the accuracy of both the models is not even 2%.
- For Dataset 2, Boosted Decision Tree(pruned) and ANN both can be considered as best algorithms for classification as the difference in the accuracy of both the models is not even 1%.

Algorithm	Accuracy -Dataset1	Accuracy - Dataset2
SVM - Linear	0.891	0.955
SVM - RBF	0.891	0.9572
SVM - Sigmoid	0.887	0.9407
Decision Tree	0.662	0.989
Decision Tree (pruned)	0.7306	0.94228
Boosted Decision Tree	0.902	0.97
Boosted Decision Tree(pruned)	0.5698	0.989
ANN	0.8965	0.98
KNN	0.868	0.89



What other steps you could have taken with regards to modeling to get better results?

- Neural network has various activation functions (Sigmoid, Tanh, Relu, Selu, soft plus, soft sign, etc). We have used 4 of them and found Tanh has given the best results. We can try different activation functions in different layers.
- Neural network has various optimizers (SGD, Adam, Adagrad etc). We can try different activation functions with different optimizers.
- Neural network has dropout regularization. We have used dropout is 0.2. We can vary dropouts and find the optimal value.