ML assignment 3

By Rupal Shrivastava

Table of Contents

Introduction 1
Dataset Description 1
Artificial Neural Network (ANN) 1
Experiment 1: Number of Epochs 1
Dataset 1: GPU Runtime 1
Dataset 2: Bank Marketing 2
Experiment 2: Number of Layers 2
Dataset 1: GPU Runtime 2
Dataset 2: Bank Marketing 3
Experiment 3: Number of Nodes 3
Dataset 1: GPU Runtime 3
Dataset 2: Bank Marketing 4
Experiment 4: Activation Functions 4
Dataset 1: GPU Runtime 4
Dataset 2: Bank Marketing 5
K-Nearest Neighbors 5
Experiment 1: Number of Neighbors 5
Dataset 1: GPU Runtime 5
Dataset 2: Bank Marketing 6
Experiment 2: Distance Metrics 6
Dataset 1: GPU Runtime 6
Dataset 2: Bank Marketing 6
Comparisons: 7
ANN vs KNN 7
Dataset 1: GPU Runtime 7
Dataset 2: Bank Marketing 7
All algorithms comparison 7
Dataset 1: GPU Runtime 7
Dataset 2: Bank Marketing 7

Introduction

In this assignment, we will implement Support Vector Machine, Decision Tree and Boosting Algorithm on the GPU runtime and Bank Marketing dataset. We are performing Binary Classification on the target variable by implementing k-fold cross validation on the training dataset for all the three algorithm and identifying the best way to classify each dataset.

Dataset Description

The GPU dataset used can be found at: https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance
The dataset consists of 14 features (independent variables) and 241,600 rows. First 10 features are ordinal and last 4 as binary variables. There were 4 runs performed on this dataset, which corresponds to 4 runtime variables in the dataset. More details on the dataset can be found in the link mentioned above.
The Bank Marketing dataset can be found at: http://archive.ics.uci.edu/ml/datasets/Bank+Marketing#
This dataset contains Direct Marketing (cold calling) campaign information of a Portuguese Bank. The goal is to predict if the client will subscribe to a term deposit. I chose this dataset because it is interesting to see how many conversations does cold calling bring to the business. This dataset contains 15 features, and 45211 rows. It has a combination ordinal, nominal and interval variables.
The assignment is divided in three parts: Data Preprocessing (which remains the same as the previous assignment for both the datasets), Algorithms and Experiments.

Artificial Neural Network (ANN)

Here, I am running an artificial neural network created using Keras by TensorFlow, using sigmoid as the output function for binary classification.

Experiment 1: Number of Epochs

In this experiment, we are changing how many times the dataset passes through the neural network to optimize the beta coefficient and weight for the nodes. As a default, I am using 2 layers, with 16 and 32 as nodes and 'relu' as the input activation function. I ran this for 3 values of epochs – 100, 250, 500. The output is follows:

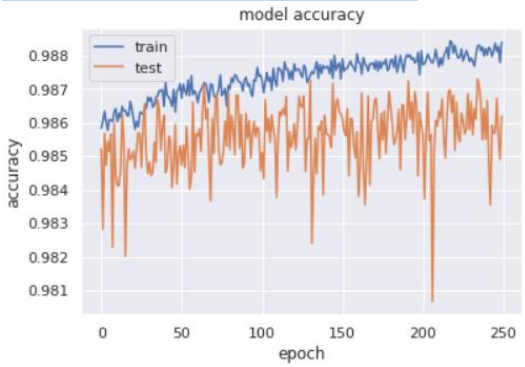
Dataset 1: GPU Runtime

In this dataset, we are getting best accuracy for with only 250 and 500 epochs as the dataset large. The precision score is slightly increasing upon increasing the number of epochs but not by much amount. This dataset has low noise.

Epoch	Test Accuracy	Precision Score
100	98.42%	96.62%
250	98.61%	97.42%
500	98.61%	97.48%

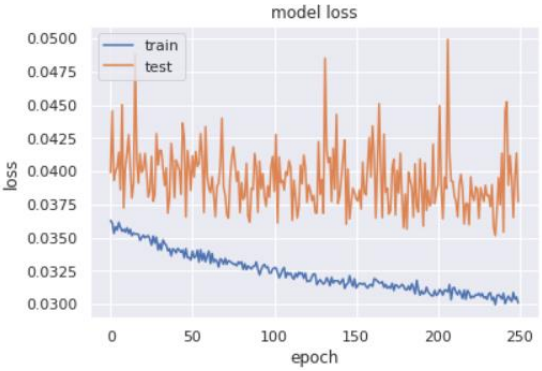
Confusion Matrix	Positive (1)	Negative (0)
Positive (1)	39164	250
Negative (0)	650	24386

Considering both accuracy and precision we select *250* as the best number of epochs for the subsequent experiments.
From the confusion matrix, we can infer that there is low type 2 error, as the number of false negatives is less. Type 1 error is also very low given false positive are low.



Now we have model accuracy as a function of epochs. We can see that as we increase the number of epochs the train accuracy increases and test accuracy is slightly fluctuating. The accuracy of testing dataset is almost constant, which means the model is generalizing well, and have low variance.

Now let have a look at model loss, here we can see that training loss keeps on reducing on increasing epochs, but testing loss remains the same, indicating that we may have sucessfully reach local minima. This means our model is able to reduce the bias and is a decent model.



Dataset 2: Bank Marketing

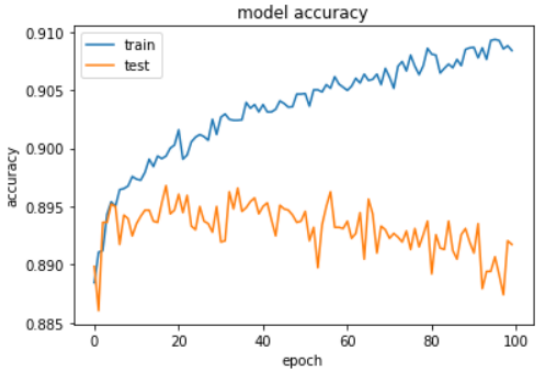
In this dataset, we are getting best accuracy for with only 100 epochs as the dataset is smaller than GPU dataset. The precision score is slightly increasing upon increasing the number of epochs but overall, we will observe low precision in this dataset, because of the possible noise in the dataset.

Epoch	Test Accuracy	Precision Score
100	88.59%	22.90%
250	88.08%	23.33%
500	87.67%	23.38%

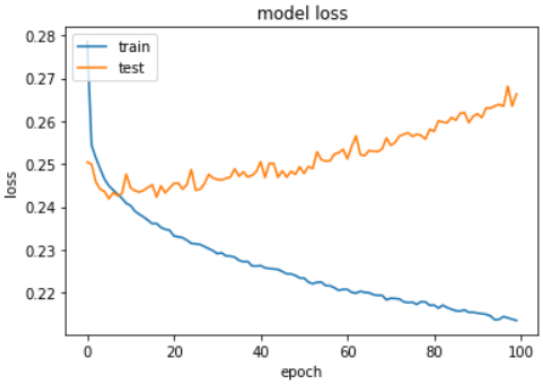
Confusion Matrix	Positive (1)	Negative (0)
Positive (1)	11584	385
Negative (0)	1163	432

Considering both accuracy and precision we select *100* as the best number of epochs for the subsequent experiments.

From the confusion matrix, we can infer that there is low type 2 error, as the number of false negatives is less. Type 1 error is higher given false positive are ~9% of the dataset.



Now we have model accuracy as a function of epochs. We can see that as we increase the number of epochs the train accuracy increases and test accuracy reduces, but not significantly. The accuracy testing dataset is almost constant, which means the model is generalizing well, and have low variance.



Now let have a look at model loss, here we can see that training loss keeps on reducing on increasing epochs, but testing loss increases with a slight amount. But again the model loss is not changing much indicating that we may have sucessfully reach local minima. This means our model is able to reduce the bias and is a decent model.

Experiment 2: Number of Layers

In this experiment, we are changing the number of layers in the neural network. Now I am using 100 epochs, with 16 and 32 as nodes and 'relu' as the input activation function. I ran this for 3 values of layers – 1, 2, 3. The output is follows:

Dataset 1: GPU Runtime

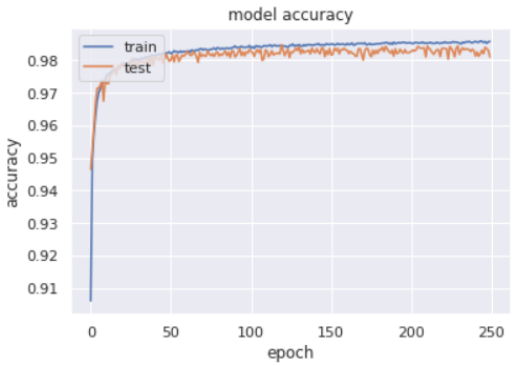
In this dataset, we are getting best accuracy for with 3 layers, again as the dataset is large. The precision score is marginally increasing upon increasing the number of layers to 3 and overall, we observe increase than the previous experiment.

Layers	Test Accuracy	Precision Score
1	96.80%	93.38%
2	98.12%	96.76%
3	98.64%	97.20%

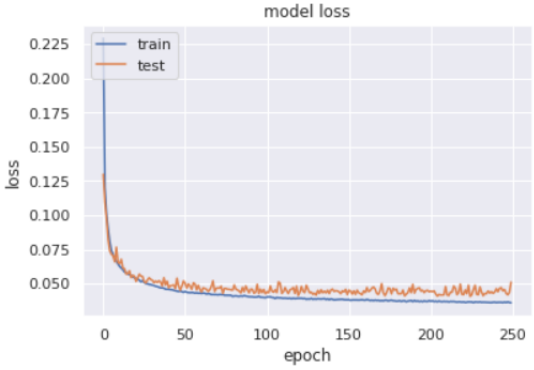
Confusion Matrix	Positive (1)	Negative (0)
Positive (1)	39241	173
Negative (0)	1043	23993

Considering both accuracy and precision we select *3* as the best number of layers for the subsequent experiments.

From the confusion matrix, we can infer that there is very low type 2 error, as the number of false negatives is also less. Type 2 error is now lower than last time.



Again, we have model accuracy as a function of epochs for the chosen number of nodes. We can see that as we increase the number of epochs the train accuracy increases and test accuracy also increases. This means the model is generalizing well, and have low variance.



Similarly, let have a look at model loss, here we can see that training loss keeps on reducing on increasing epochs, so does the testing loss. This model loss is changing slightly indicating that we can still imporve the model to reach local minima. This means our model is not able to reduce the bias well.

Dataset 2: Bank Marketing

In this dataset, we are getting best accuracy for with only 1 layer, again as the dataset is smaller than GPU dataset. The precision score is marginally increasing upon increasing the number of layers to 2 and overall, we observe 5% increase than the previous experiment.

Layers	**Test Accuracy**	**Precision Score**
1	89.10%	25.19%
2	88.13%	27.08%
3	87.37%	23.83%

Confusion Matrix	**Positive (1)**	**Negative (0)**
Positive (1)	11305	664
Negative (0)	946	649

Considering both accuracy and precision we select *2* as the best number of layers for the subsequent experiments. From the confusion matrix, we can infer that there is low type 2 error, as the number of false negatives is less, but double the previous experiment. Type 1 error is now lower than last time. The model accuracy as a function of epochs and model loss, remains similar as the previous case, since we used the same number of layer.

Experiment 3: Number of Nodes

Now we are experimenting with number of nodes or activation functions in each of the two hidden layers. I ran this for 3 combination of nodes – (16,16,32), (32,16,32), (32,32,63). The output is as follows:

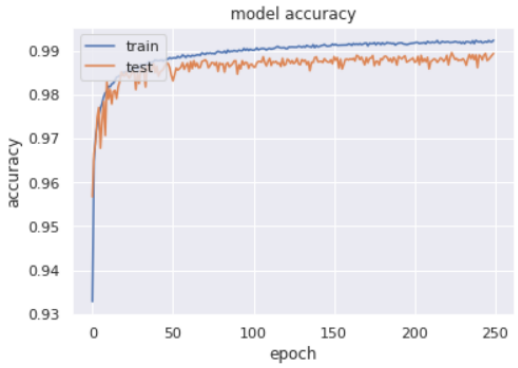
Dataset 1: GPU Runtime

In this dataset, we are getting best accuracy for with only 32 nodes in the first and last layer and 16 nodes in the second layer. The precision score is also slightly smaller than (32,32,63) case. We observe a slight increase in accuracy and in precision from the previous experiment.

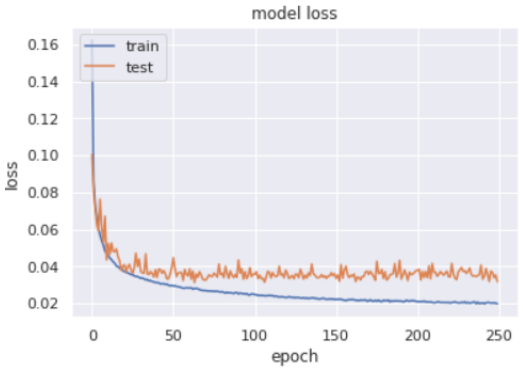
Nodes	**Test Accuracy**	**Precision Score**
16, 16, 32	98.52%	97.06%
32, 16, 32	98.90%	97.79%
32, 32, 64	98.85%	97.92%

Confusion Matrix	**Positive (1)**	**Negative (0)**
Positive (1)	39096	318
Negative (0)	390	24646

Considering both accuracy and precision we select *32, 16, 32 nodes for each layer respectively* as the best number of nodes for the subsequent experiments.
From the confusion matrix, we can infer that there is low type 2 error, as the number of false negatives is less, and lower than the previous experiment. Type 1 error is also lower than last time.



Again, we have model accuracy as a function of epochs for the chosen number of nodes. We can see that as we increase the number of epochs increases the train and test accuracy. Again, the accuracy testing dataset becomes almost constant, which means the model is generalizing well, and have low variance.



Similarly, let have a look at model loss, here we can see that training loss keeps on reducing on increasing epochs, but testing loss is constant. Again, this means our model is able to reduce the bias well.

Dataset 2: Bank Marketing

In this dataset, we are getting best accuracy for with only 32 nodes in the first layer and 16 nodes in the second layer. The precision score is also the best for this combination as opposed to others. We observe a slight increase in accuracy and slight decrease in precision from the previous experiment.

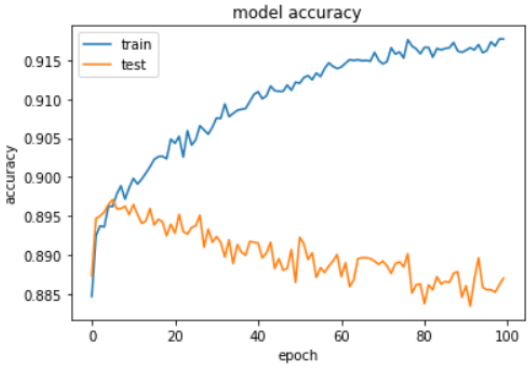
Nodes	**Test Accuracy**	**Precision Score**
16, 16	88.82%	25.64%
32, 16	88.38%	25.78%
32, 32	87.62%	26.04%

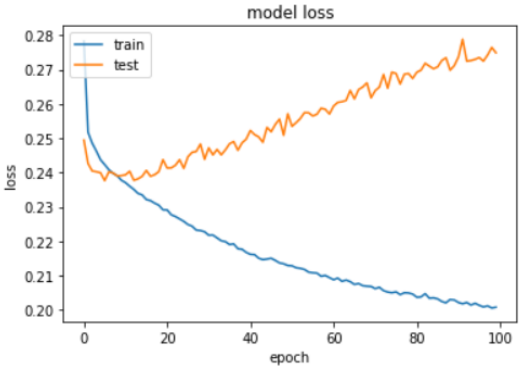
Confusion Matrix	**Positive (1)**	**Negative (0)**
Positive (1)	11416	553
Negative (0)	1023	572

Considering both accuracy and precision we select *32, 16 nodes for each layer respectively* as the best number of nodes for the subsequent experiments.

From the confusion matrix, we can infer that there is low type 2 error, as the number of false negatives is less, and lower than the previous experiment. Type 1 error is now slightly higher than last time.

Again, we have model accuracy as a function of epochs for the chosen number of nodes. We can see that as we increase the number of epochs the train accuracy increases and test accuracy reduces. Again, the accuracy testing dataset is almost constant, which means the model is generalizing well, and have low variance.





Similarly, let have a look at model loss, here we can see that training loss keeps on reducing on increasing epochs, but testing loss increases. This model loss is changing slightly indicating that we can still imporve the model to reach local minima. This means our model is not able to reduce the bias well.

Experiment 4: Activation Functions

Here we are changing the activation function in both the hidden layers. I ran this for 3 other activation functions than relu – tanh, sigmoid and softmax. The output is as follows:

Dataset 1: GPU Runtime

In this dataset, we are getting best accuracy for with relu as the activation function in all the layers. The precision score is also the best for this function. So all other results remain the same. We observe the best accuracy and precision in this model as opposed to all the previous experiment.

Activation Function	**Test Accuracy**	**Precision Score**
relu	98.52%	97.06%
tanh	98.47%	96.79%
sigmoid	98.43%	97.52%
softmax	98.67%	97.56%

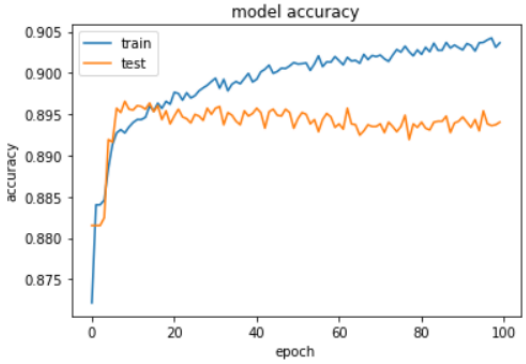
Dataset 2: Bank Marketing

In this dataset, we are getting best accuracy for with softmax as the activation function in both the layers. The precision score is also the best for this function. We observe the best accuracy and precision in this model as opposed to all the previous experiment.

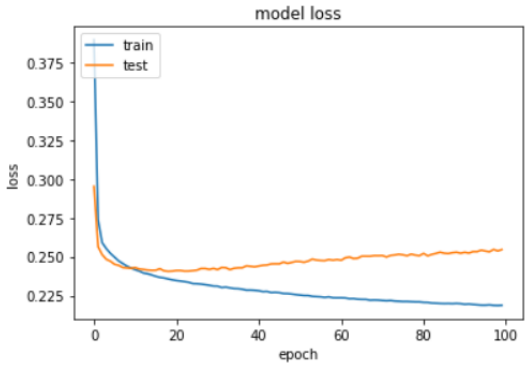
Activation Function	Test Accuracy	Precision Score
relu	88.38%	25.78%
tanh	87.90%	25.65%
sigmoid	88.74%	27.99%
softmax	88.92%	29.43%

Confusion Matrix	Positive (1)	Negative (0)
Positive (1)	11390	579
Negative (0)	923	672

Considering both accuracy and precision we select *this as the best ANN model*.
From the confusion matrix also, we can infer that there is low type 2 error and type 1 error as opposed to all previous models.



Again, we have model accuracy as a function of epochs for the chosen activation function. We can see that as we increase the number of epochs the train accuracy increases and test accuracy reduces. Again, the accuracy testing dataset is almost constant with equivalent to no variation, which means the model is generalizing well, and have low variance.



Similarly, let have a look at model loss, here we can see that training loss keeps on reducing on increasing epochs, and testing loss is also low and constant. This model loss is hardly changing indicationg we have reached a local minima. This means our model is able to reduce the bias well.

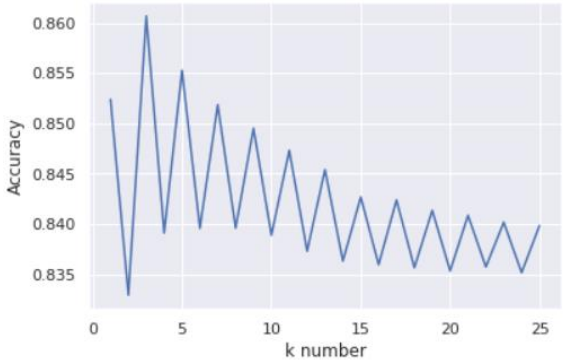
K-Nearest Neighbors

Another classification algorithm, where clusters of the datapoints is created using the training datasets and the new data points are classified in as a part of any of the cluster based on the distance of the data point from the cluster center. I have applied 10 folds cross validation in these experiments to improve the model performance.

Experiment 1: Number of Neighbors

In this experiment we are changing the number of clusters and calculating model performance for each iteration. The cluster number for which model which provides the best result is chosen as the optimum model. I have experimented for 25 values of k. The results are as follows:

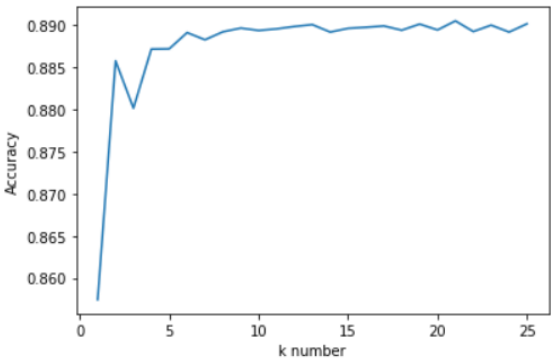
Dataset 1: GPU Runtime



In this dataset, we get the highest accuracy for k = 3 as it can be seen from the graph. The model accuracy at k = 3 is 86.70% and precision is 85.56%.
After k = 3 the accuracy becomes almost constant indicating that new data point is being classified in of the existing cluster.
Looking at the confusion matrix we can see high type 1 and type 2 error, which is not good and have scope of improvement.

Confusion Matrix	Positive (1)	Negative (0)
Positive (1)	35489	3925
Negative (0)	4643	20393

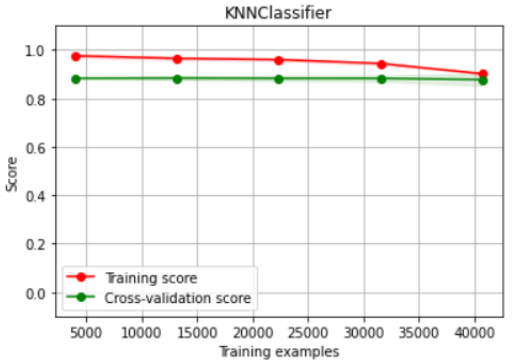
Dataset 2: Bank Marketing



In this dataset, we get the highest accuracy for k = 9 as it can be seen from the graph. The model accuracy at k = 9 is 89.32% and precision is 32.54%.

After k = 9 the accuracy becomes constant indicating that new data point is being classified in of the existing cluster.

Confusion Matrix	Positive (1)	Negative (0)
Positive (1)	11640	329
Negative (0)	1229	366



Looking at the confusion matrix we can see low type 2 error and high type 1 error, which is acceptable, but have scope of improvement. We also have the learning curve for k = 9, where we can see almost constant accuracy for validation dataset, implying the model is generalizing well.

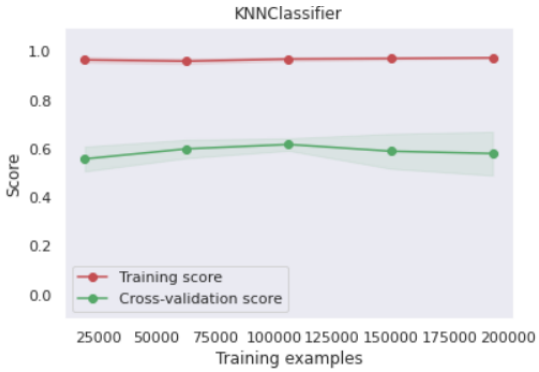
Experiment 2: Distance Metrics

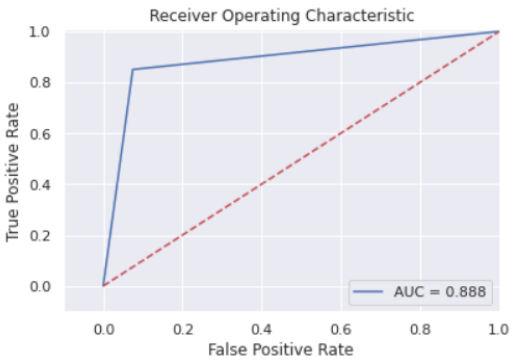
In this experiment we are changing the distance metrics, used to calculate the distance between the center and the data point. The distance metrics used in the experiment are: Euclidean, Manhattan and Chebyshev. The output is as follows:

Dataset 1: GPU Runtime

When using the manhattan metric, we are getting the highest accuracy and precision (as compared to others), as can be seen in the table. We also have the learning curve this metric, where we can see almost constant accuracy for validation dataset, implying the model is generalizing well. Also the ROC curve is showing that model is precise and has low type 1 and type 2 errors.

Nodes	**Test Accuracy**	**Precision Score**
euclidean	86.71%	85.46%
manhattan	89.62%	89.02%
chebyshev	80.49%	79.81%

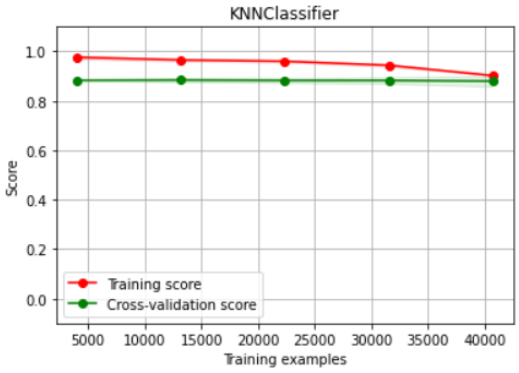




Dataset 2: Bank Marketing

When using the manhattan metric, we are getting the highest accuracy and decent precision (as compared to others), as can be seen in the table. We also have the learning curve this metric, where we can see almost constant accuracy for validation dataset, implying the model is generalizing well. But the ROC curve is showing that model is not precise and has type 1 and type 2 errors.

Nodes	**Test Accuracy**	**Precision Score**
euclidean	88.51%	32.64%
manhattan	88.58%	30.78%
chebyshev	87.40%	30.04%



Comparisons:

Now let's compare the classification algorithms for both the dataset:

ANN vs KNN

Dataset 1: GPU Runtime

In this dataset we can see that ANN is performing way better than KNN as the dataset is large.

Algorithm	**Test Accuracy**
ANN	98.90%
KNN	89.62%

Dataset 2: Bank Marketing

In this dataset we can see that ANN is performing slightly better than KNN but as we have seen both the algorithms are not precise, but KNN has better precision than ANN.

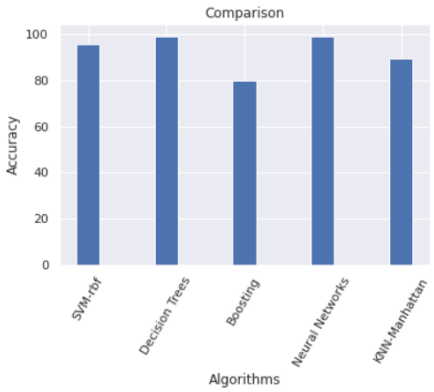
Algorithm	**Test Accuracy**
ANN	88.92%
KNN	88.58%

All algorithms comparison

Dataset 1: GPU Runtime

For this dataset, we get the model classification model using Decision tree but that may be overfitting the dataset. So the best algorithm here would be ANN, which makes sense as we know the data is large.

Algorithm	**Test Accuracy**
SVM - rbf	95.94%
Decision Tree	99.32%
Boosting	80.03%
ANN	98.90%
KNN	89.62%



Dataset 2: Bank Marketing

For this dataset, we get the model classification model using Boosting, which makes sense as we know the data has noise and precision is low.

Algorithm	**Test Accuracy**
SVM - rbf	88.78%
Decision Tree	85.29%
Boosting	89.52%
ANN	88.92%
KNN	88.58%