Contents

INTRODUCTION

The following report is a combined study of various algorithms like K-nearest neighbor and Neural Networks.

We have two datasets:

1. Runtime performance (sgemm_product)
2. Diabetic Retinopathy

We have worked previously with both the datasets and performed a descriptive analysis in the previous report. The description of the diabetic retinopathy dataset is following:

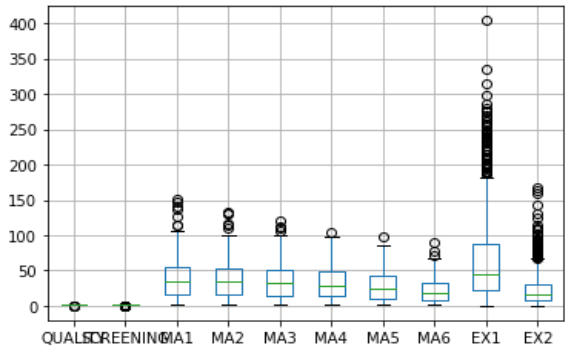
1. The data contains 1151 rows and 20 columns
2. The dependent variable is 'Class' which was UTF-8 encoded and was decoded as 0 (b'\x30') and 1 (b'\x31')
3. The data was scaled to handle the skewness and bring to common scale using [-1,1]
4. The dataset does not have any null values
5. The dataset was split into train and test ratio as 70:30

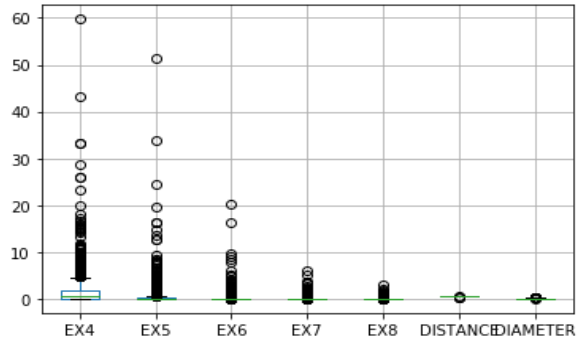
Reason for choosing data Set

People with diabetes can have an eye disease called diabetic retinopathy. This is when high blood sugar levels cause damage to blood vessels in the retina. These blood vessels can swell and leak. Or they can close, stopping blood from passing through. Sometimes abnormal new blood vessels grow on the retina. All these changes can steal your vision.

This can be used in the healthcare industry which will reduce the inaccuracies to test with human eye or could be validation to medical reports. This can affect the patient's life and would be a boon for healthcare artificial intelligence. Also, we have learnt in the class about how neural networks can help us in predictions and going forward it will be interesting to compare the variegated models. This will enhance the model and learning as it contains both numerical and categorical features.

Following is the box plot of features in the dataset:

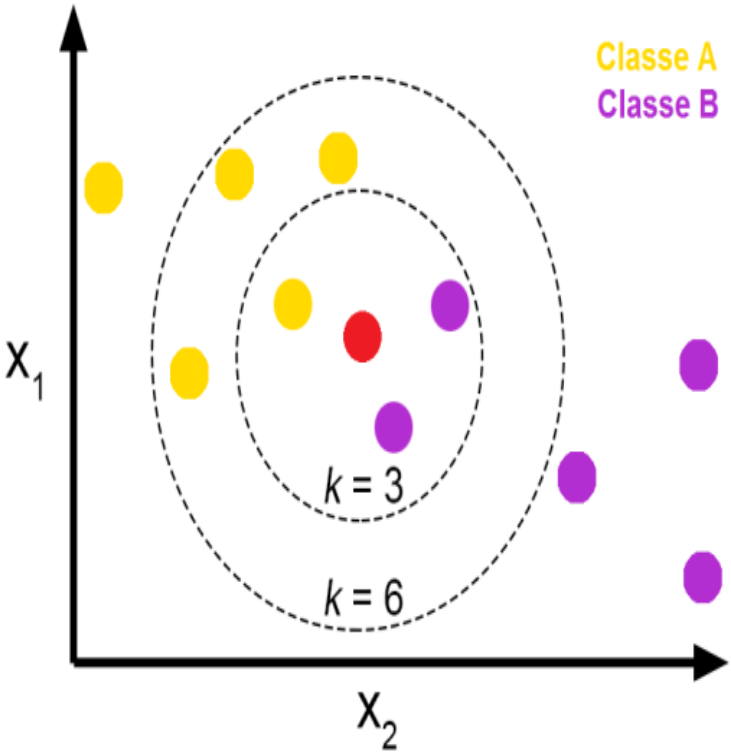




Outlier Treatment

Since, we can see the outliers in MA (Microaneurysms) and EX (Exudates) the urge could be to remove them from the dataset. But, the range of MA and EX is very large and can from individual to individual. In healthcare industry, the patients who are fit would have similar levels of the test while the sick's level can vary depending on severity. Hence, we will not remove the outliers in this case.

K-NEAREST NEIGHBOUR



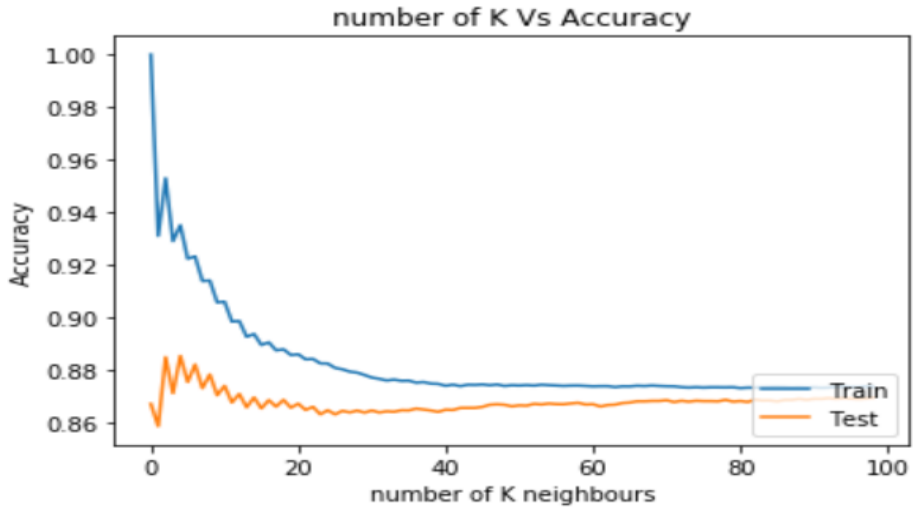
K-NN is a **non-parametric** and **lazy learning algorithm**. Non-parametric means there is no assumption for underlying data distribution i.e. the model structure determined from the dataset. It is called Lazy algorithm because it does not need any training data points for model generation. All training data is used in the testing phase which makes training faster and testing phase slower and costlier.

It stores all the available cases and classifies the new data or case based on a similarity measure.

To determine which of the K instances in the training dataset are most similar to a new input, **a distance measure is used**. For real-valued input variables, the most popular distance measure is the Euclidean distance. The Euclidean distance is the most common distance metric used in **low dimensional data sets**. It is also known as the **L2 norm**. The Euclidean distance is the usual way distance is measured in the real world.

Dataset- Diabetic Retinopathy

The KNN classification was applied on the dataset with different values of K. Initially, an arbitrary number of neighbors was used to check how the algorithm behaves. Generally, the rule of thumb to choose K is sqrt(n) where n is the number of samples. Hence, the test and train accuracies were recording various K which is plotted as below.



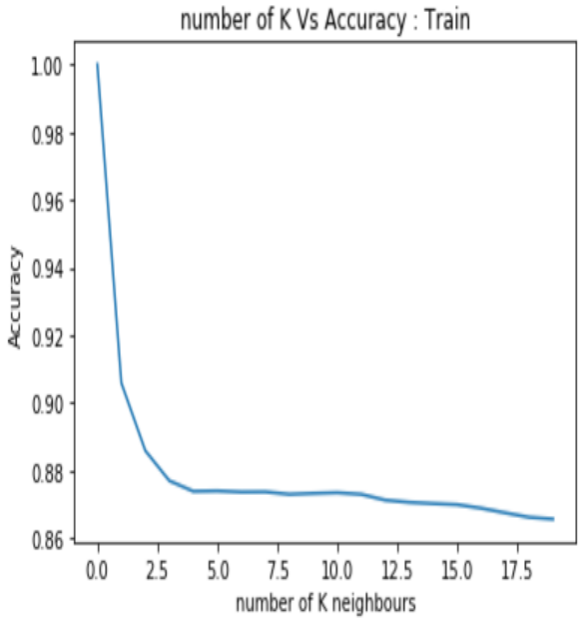
- We can infer from the graph that doesn't increase with K but reaches an average threshold.

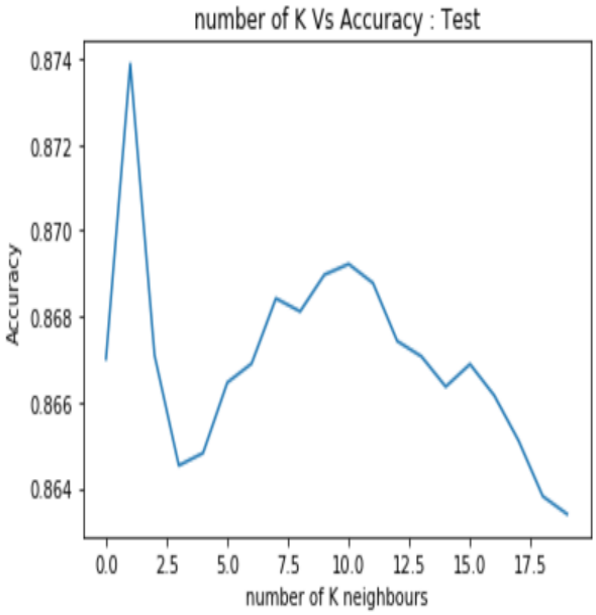
Number of neighbors (K)	Accuracy	
	Train	Test
2	79.75%	52.60%
3	81.99%	61.27%
4	75.90%	56.65%
5	76.02%	60.98%
10	71.80%	56.65%
20	69.44%	58.67%
30	67.70%	57.80%

- From the train and test plot and the recorded accuracy we infer that the maximum accuracy is obtained at K=3.
- The sample of accuracies for various K is recorded in the table on the left
- The number of samples were 1151 and sqrt (1151) =33 is the threshold for finding K
- But we found maximum accuracy at K-3 and hence we chose that

Dataset- Runtime

The KNN classification was applied on the dataset with different values of K. Initially, an arbitrary number of neighbors was used to check how the algorithm behaves. Generally, the rule of thumb to choose K is sqrt(n) where n is the number of samples. Hence, the test and train accuracies were recording various K which is plotted as below:





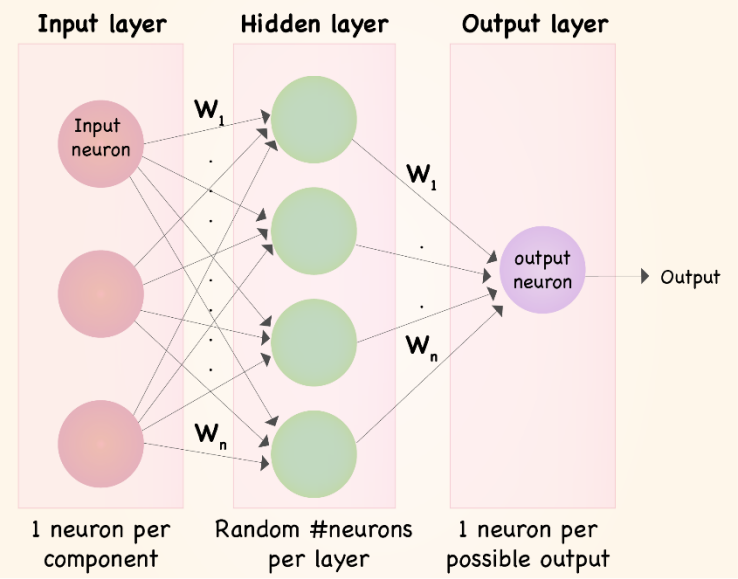
Again, as we saw for the previous dataset, the accuracy reaches a threshold after a certain number of nearest neighbours.

Number of neighbors (K)	Accuracy	
	Train	Test
11	90.59%	87.39%
21	88.59%	86.71%
31	87.70%	86.45%
41	87.39%	86.48%
51	87.40%	86.65%
61	87.37%	86.69%
71	87.37%	86.84%
81	87.30%	86.81%
91	87.33%	86.90%
101	87.35%	86.92%

- As the number of samples in this data is 169,000 the number of K range from 1 to 100
- The accuracy for various values of K are recorded in the table in the left
- We infer from the plots above and table on the left that K=11 as the algorithm achieves maximum accuracy at that value

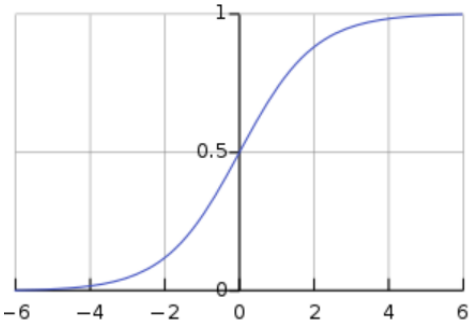
NEURAL NETWORKS

The neural networks are called artificial neural networks and are based on what science knows about the human brain's structure and function. Briefly, a neural network is defined as a computing system that consist of a number of simple but highly interconnected elements or nodes, called 'neurons', which are organized in layers which process information using dynamic state responses to external inputs. This algorithm is extremely useful in finding patterns that are too complex for being manually extracted and taught to recognize to the machine. In the context of this structure, patterns are introduced to the neural network by the *input layer* that has one neuron for each component present in the input data and is communicated to one or more *hidden layers* present in the network; called 'hidden' only due to the fact that they do not constitute the input or output layer. It is in the hidden layers where all the processing actually happens through a system of connections characterized by ***weights and biases*** *(commonly referred as W and b):* the input is received, the neuron calculate a weighted sum adding also the bias and according to the result and a pre-set **activation function** (most common one is sigmoid, *σ,* even though it almost not used anymore and there are better ones like ReLu*)*, it decides whether it should be 'fired' or activated. Afterwards, the neuron transmits the information downstream to other connected neurons in a process called '*forward pass*'. At the end of this process, the last hidden layer is linked to the *output layer* which has one neuron for each possible desired output.

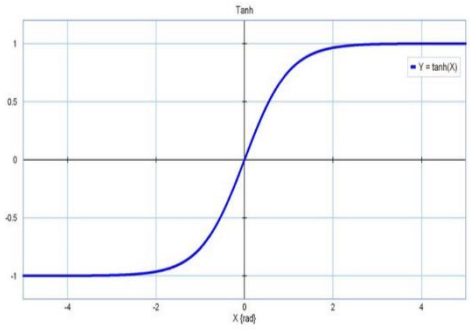


Dataset- Diabetic Retinopathy

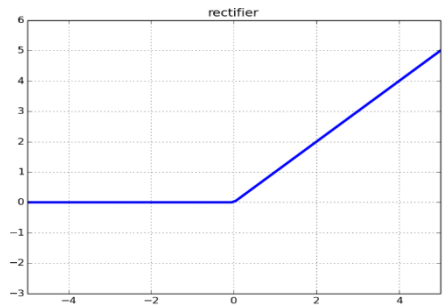
The neural network was built on the diabetic retinopathy dataset which has 18 features in total. The key task is to experiment with the number of hidden layers and nodes, epochs and the batch size. There are various activation functions which can be used to trigger the ANN. Some of commonly used are sigmoid, tanh and ReLu. The gradient of sigmoid becomes increasingly small as the absolute value of the variable increases. ReLu is better to use as the constant gradient of ReLu results in faster learning. We generally use Relu for the input and hidden layers while we use sigmoid for the output layer. The graph of these functions is below:



The Sigmoid function



The tanh function

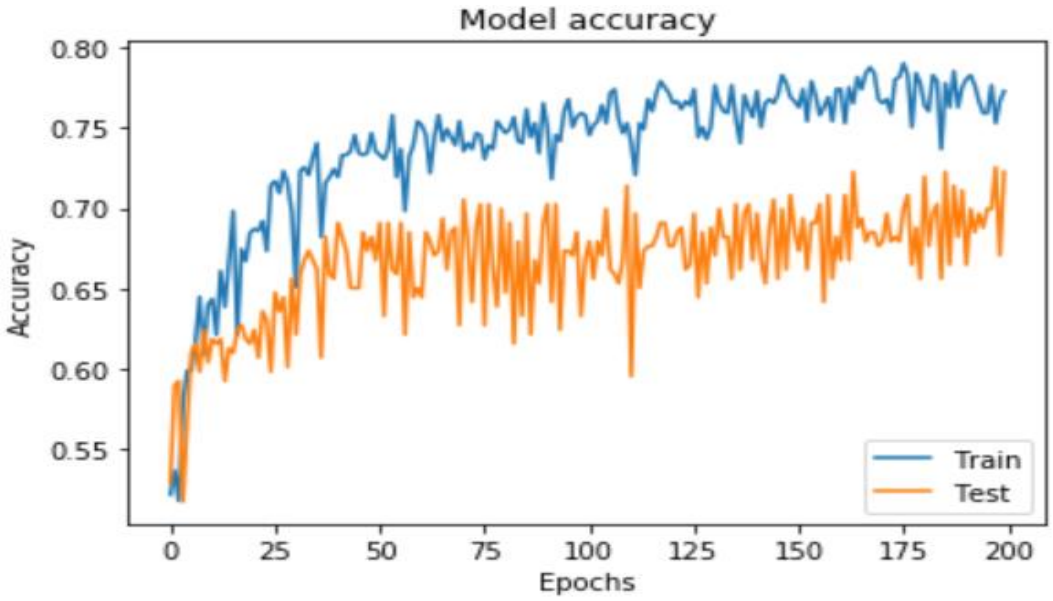


The ReLU function

The table below compares the accuracy of the model with various activation function and hypermeters:

Activation: Relu and Sigmoid							
Number of nodes				Hyperparameters		Accuracy	
Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Epochs	Batch size	Train	Test
19	8	4	-	150	40	78.51%	67.92%
19	12	8	4	100	160	68.20%	63.58%
19	12	8	-	200	100	75.78%	72.25%
Activation: Tanh and Sigmoid							
19	12	8	-	150	40	80.87%	65.32%
19	12	8	-	100	160	74.29%	66.18%
19	12	8	-	200	100	80.62%	66.47%
Activation: Sigmoid							
19	12	8	-	150	40	83.35%	71.10%
19	12	8	-	100	160	68.70%	58.96%
19	12	8	-	200	100	78.39%	67.92%

The best accuracy was achieved with Relu and sigmoid as the activation function. The combination of hidden layers and nodes was then kept same while exploring the other activation functions. But the number of epochs and batch size was altered to achieve higher accuracy. But we can infer that the activation function ReLu and sigmoid were the best neural networks with train accuracy= 75.78% and test accuracy=72.25%. Following is the train and test accuracy Vs epochs for the best model:

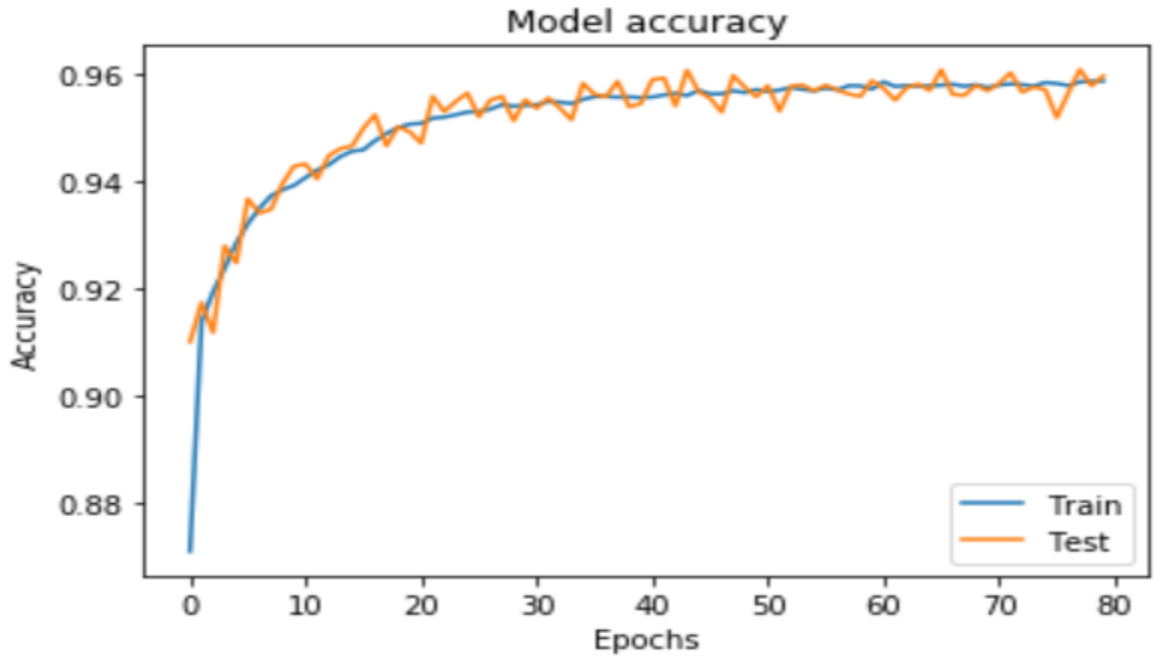


Dataset- Runtime

The neural network was built on the GPU runtime dataset which has 14 features in total. The key task is to experiment with the number of hidden layers and nodes, epochs and the batch size. There are various activation functions which can be used to trigger the ANN. We generally use Relu for the input and hidden layers while we use sigmoid for the output layer. The table below compares the accuracy of the model with various activation function and hypermeters:

Activation: Relu and Sigmoid							
Number of nodes				Hyperparameters		Accuracy	
Input layer	Hidden layer 1	Hidden layer 2	Hidden layer 3	Epochs	Batch size	Train	Test
15	8	4	-	100	150	95.84%	95.78%
15	10	6	1	80	20	95.72%	95.71%
15	10	6	4	50	150	95.97%	95.83%
Activation: Tanh and Sigmoid							
15	8	4	-	100	150	94.05%	93.91%
15	10	6	1	80	20	95.43%	95.33%
15	10	6	4	50	150	94.93%	94.99%
Activation: Sigmoid							
15	8	4	-	100	150	95.60%	95.46%
15	10	6	1	80	20	95.91%	95.95%
15	10	6	4	50	150	94.13%	94.02%

The best accuracy was achieved with Relu and sigmoid as the activation function. The combination of hidden layers and nodes was then kept same while exploring the other activation functions. But the number of epochs and batch size was altered to achieve higher accuracy. But we can infer that the activation function sigmoid beat ReLu and sigmoid with train accuracy= 95.91% and test accuracy=95.95%. Following is the train and test accuracy Vs epochs for the best model:



BEST MODEL COMPARISON

We applied various classification techniques to both the data sets in order to achieve maximum accuracy. Some of the techniques we have used is logistic regression, SVM, decision trees etc. In the real life, it is difficult to guess the correct model for a dataset and therefore a plethora of techniques are used to get the best results. Let us compare the results of all the algorithms and select the best model:

Dataset- Diabetic Retinopathy

Technique/Algorithm	Train accuracy	Test accuracy
Logistic Regression	-	-
Neural Network	75.78%	72.25%
SVM (kernel = linear) - K cross	69.33%	70.14%
SVM (kernel = linear)	75.53%	69.36%
Decision Trees (depth=12)- Boosting (K cross)	95.71%	63.99%
Decision Trees (depth=12) - K cross	87.55%	63.42%
Decision Trees (depth=12)- Boosting	100.00%	62.43%
Decision Trees (No pruning)- K cross	100.00%	61.77%
KNN	81.99%	61.27%
Decision Trees (No pruning)- Boosting (K cross)	100.00%	60.99%
Decision Trees (depth=12)	100.00%	58.96%
Decision Trees (No pruning)	100.00%	58.67%
Decision Trees (No pruning)- Boosting	100.00%	57.80%

- The above table has all the algorithms we have used so far for classification of the diabetic retinopathy dataset
- It is sorted by the test accuracy in the descending order

- We can see from the table that the neural networks give the best accuracy and is considered the best of all the models
- The accuracy obtained by the neural network is Train accuracy= 75.78 % and Test accuracy= 72.25 %
- The second-best model is the cross validation of SVM with linear kernel with Train accuracy= 69.33 % and Test accuracy= 70.14 %

Dataset- Runtime

Technique/Algorithm	Train accuracy	Test accuracy
Decision Trees (depth=18)- Boosting	100.00%	99.20%
Decision Trees (depth=18)	100.00%	99.18%
Decision Trees (No pruning)- Boosting	100.00%	99.17%
Decision Trees (No pruning)	100.00%	99.17%
SVM (kernel = rbf)	96.13%	96.07%
Neural Network	95.91%	95.95%
SVM (kernel = rbf)- K cross	95.02%	89.43%
KNN	90.59%	87.39%
Logistic Regression	80.95%	81.22%
Decision Trees (depth=18)- Boosting (K cross)	100.00%	77.85%
Decision Trees (No pruning)- Boosting (K cross)	99.57%	77.69%
Decision Trees (No pruning)- K cross	100.00%	77.69%
Decision Trees (depth=18) - K cross	99.85%	76.25%

- The above table has all the algorithms we have used so far for classification of the GPU runtime dataset
- It is sorted by the test accuracy in the descending order
- We can infer from the table that the decision trees with gradient boosting at depth=18 give the best accuracy and is considered the best of all the models
- The accuracy obtained by the decision tress is Train accuracy= 100 % and Test accuracy= 99.20 %
- This is because the gradient boosting with Adaboost allows the weak learners to lean and improve the accuracy of the model
- The second-best model is decision trees without boosting at depth=18 with Train accuracy= 69.33 % and Test accuracy= 99.18 %

CONCLUSION

- We have used various techniques/algorithms in these 3 assignments and come up with the best model so far
- One way of improving the accuracy further is to improve the domain knowledge or know the variables better to know the importance of the variables to be included or not in the model