**Assignment 3**

**Introduction:**

The objective of this project is to predict high intensive GPU loads and telecom customer churn using the selected two datasets. We will be using artificial neural networks and k nearest neighbor algorithms to find the best model to perform each classification task. We will be using learning curves to fine tune parameters like activation functions, number of layers and number of nodes for ANN depth and number of neighbors for KNN.

**Part 1**

**Telecom Customer Churn Prediction**

**Dataset:**

The dataset we will be using for this project has a total of 21 features and 7043 records. We will be dropping customerId and PaymentMethod from the dataset and will be using churn as our target variable. Partner, Dependents, PhoneService, PaperlessBilling, Churn, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies variables are converted to binary. Dummy variables have been added to convert InternetService, Contract and Gender to categorical variables. After the above changes we have 20 features we will be using as our independent variables to predict out target variable which is churn.

**Experiment 1:**

In this experiment we will be analyzing the performance of three different ANN models using learning curves and cross fold validation. We will be choosing the best model for predicting the customer churn based of the bias and variance seen in the learning curves and the average accuracy on the validation set of each model. The ANN activation functions we will be using for this experiment are sigmoid, Tanh and Relu. Each of these models will have one hidden layer with 128 nodes and an output layers with two nodes. Our model will use the Stochastic gradient descent optimizer to update our weights and bias values. It will also use binary cross entropy as the loss function and accuracy as the evaluation metrics.

**Sigmoid activation performance:**

To plot the learning curve, we will be running the ANN model with sigmoid function for different sizes of the training set:



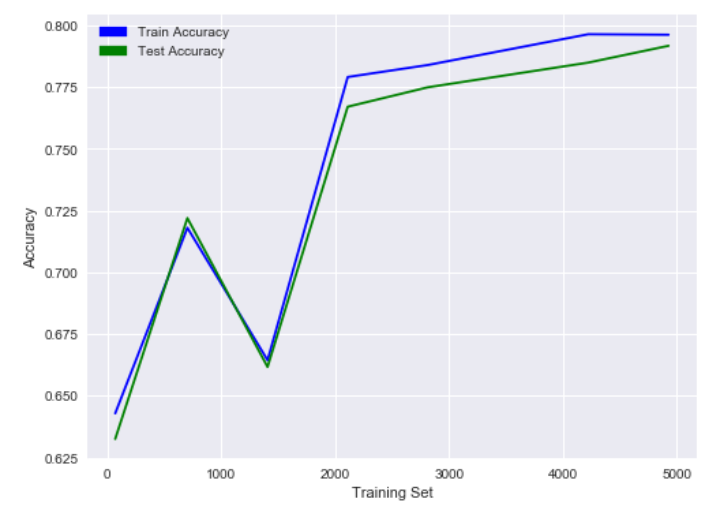
We can see from the graph that the variance becomes low when training size is between 2200 and 4300. The bias seems to be high initially causing training accuracy to be low, but it doesn’t seem to affect test accuracy which is strange. As the training size increases, we can see that the model performs well till it starts to overfit above 4300 training data points.

Next, we run cross fold validation on 90% of the training set with 5 folds and check the average accuracy of our validation sets. This will give use a good idea of how well this model will perform on the churn dataset as all the datapoints will be used in training and test atleast once. The average accuracy of this model using a sigmoid activation function is given below:

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 73.71% |

**Tanh activation performance:**

Repeating the above experiment twice with a Tanh activation function in hidden and output layers gives the below learning curves:



From the above curves we can see that the model is very unreliable for training set sizes below 2100. The model seems to be unable to handle the bias when the dataset is small. Although the performance of the model is better as the training set size increases, it is better if we avoid using this model.

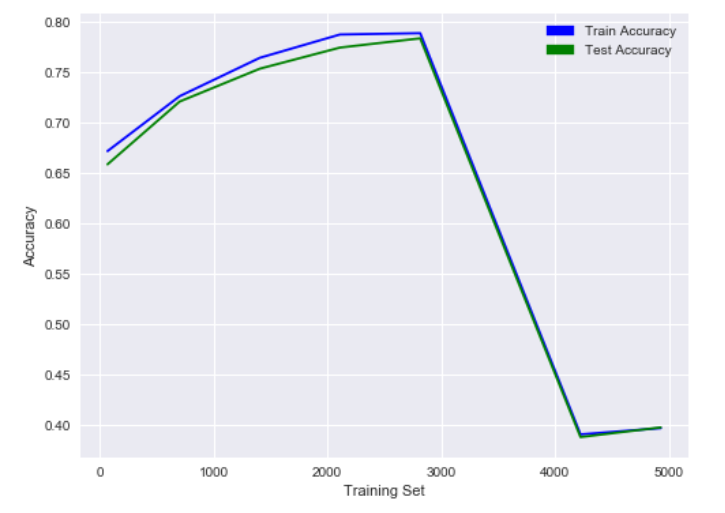
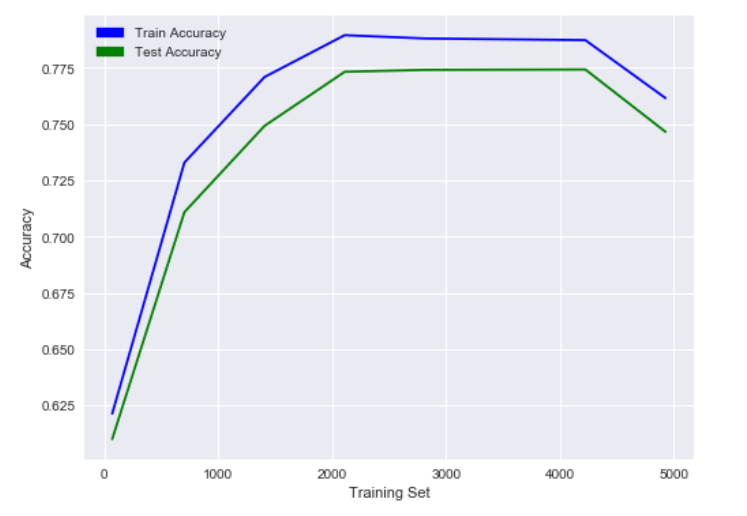
The average validation set accuracy of this model using cross validations with 5 folds and 90% of the data is:

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 79.71% |

Although this average accuracy of this model is high, we can see from the learning curve that this model doesn’t generalize well when bias is high.

**Relu activation performance:**

Running the model twice with Relu activation function in both hidden and output layers gives the below learning curves:



From the curve we can see that the performance of the Relu activation function is even more unreliable when compared to Tanh. The performance seems to vary drastically between each run. This suggests that the model is not adept at handling the noise in this dataset.

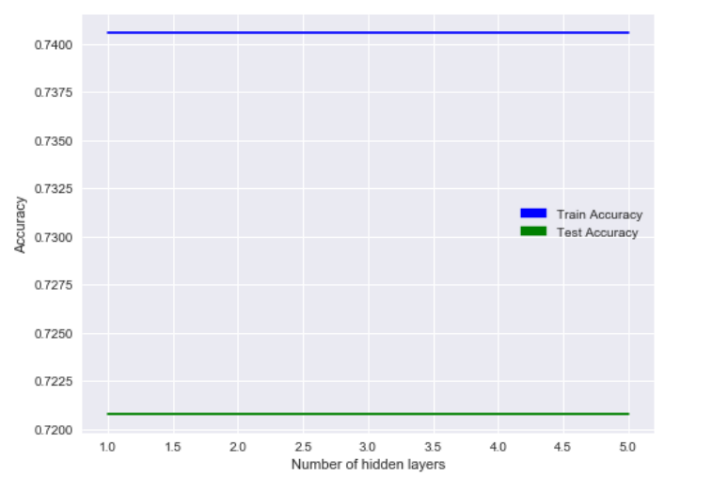
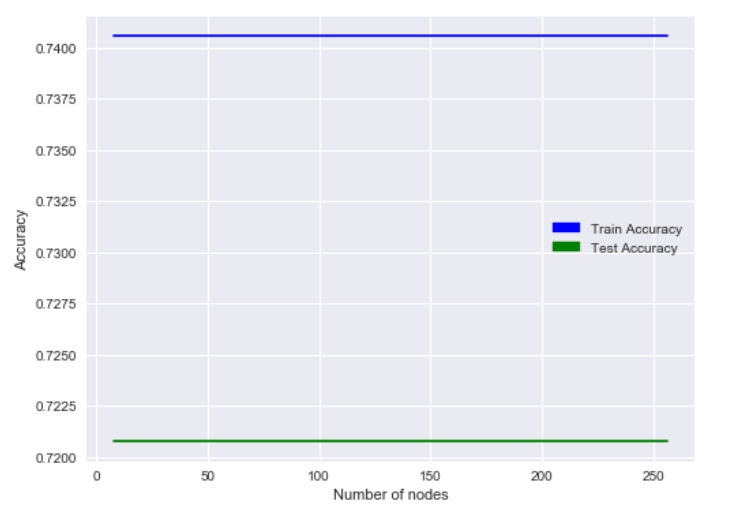
The average validation set accuracy of this model using cross validations with 5 folds and 90% of the data is

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 52.69% |

The accuracy of these models cannot be trusted as the performance of the model changes with each run. Therefore, we can conclude from this experiment that the sigmoid activation function gives the best performance for this dataset.

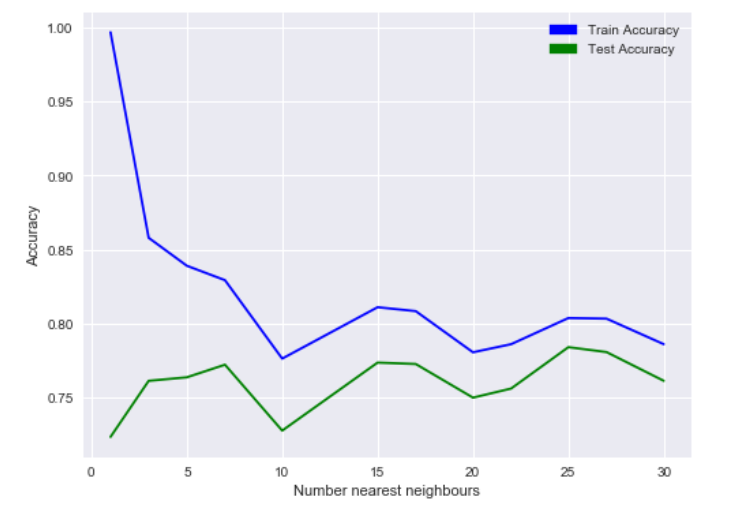
**Experiment 2:**

In this experiment we will be studying the impact of increasing the number of nodes and number of hidden layers in an ANN model with sigmoid activation function for the churn dataset. We will be running the models for different number of nodes in the hidden layer and compare the training and test accuracy to select the best number of nodes for this activation function. We will repeat the same by varying the number of hidden layers to get the best values. Plotting the test and train accuracy of the above experiments we get the following graphs.

From the plots we can see that the number of nodes and hidden layers has no impact on the test and train accuracy of this model. So, we conclude that the performance of the model does not depend on the number of nodes and number of hidden layers.

**Experiment 3:**

In this experiment we will be predicting the customer churn using KNN. We will be running KNN with varying number of neighbors and we will be plotting them against test and train accuracy to pick the best model. We will be using the default weigh function which will weigh all points in the neighborhood equally. Plotting number of neighbors versus train and test accuracy we get the below plot.



From the graph we can we that the bias is high when the number of neighbors is below 7 and decrease as we increase the number of neighbors. We can also see that the variance decreases as we increase the number of neighbors. From these results we can conclude that 25 neighbors give us the best results for this dataset.

Computing the average validation set accuracy of this model with 25 neighbors using cross validations with 5 folds and 90% of the data we get:

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 78.20% |

**Conclusion:**

In order to select the best algorithm for predicting the telecom customer churn we need to compare the accuracy of the best models from both ANN and KNN. From experiments one to three we can conclude that he best ANN model is the first sigmoid model with 128 nodes in the hidden layer. Comparing this to the best models from experiment four and assignment 2 we get the below results.

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| ANN with sigmoid activation | 73.71% |
| KNN with 25 neighbors | 78.20% |
| SVM with linear kernel | 79.36% |
| SVM with sigmoid kernel | 79.36% |
| Decision tree | 79.43% |
| Gradient boosted tree | 80.32% |

From the above results we can see that the ANN and KNN models are outperformed by all the models from assignment 2. So, we conclude that the gradient boosted tree is still the best classifier for this dataset.

**Part 2**

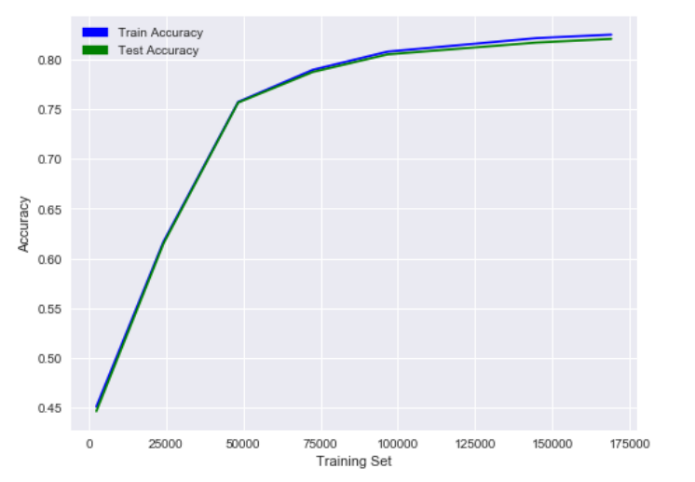
**GPU Average Runtime Prediction**

**Dataset:**

The dataset we will be using for this project has a total of 18 features and 241600 records. We will be taking the average of 4 of these features Run1, Run2, Run3 and Run4 as AverageRun and will be using the remaining 14 for our prediction. For the logistic problem we will be predicting if the average GPU run time is greater than the median. This splits our data almost equally with 50% of ones and 50% zeroes.

**Experiment 1:**

In this experiment we will be analyzing the performance of three different ANN models using learning curves and cross fold validation. We will be choosing the best model for predicting the high intensive GPU loads based of the bias and variance seen in the learning curves and the average accuracy on the validation set of each model. We will be running the test on the same 3 activation functions as the previous data set and we will also be using the name optimizer, metrics, loss function, etc.

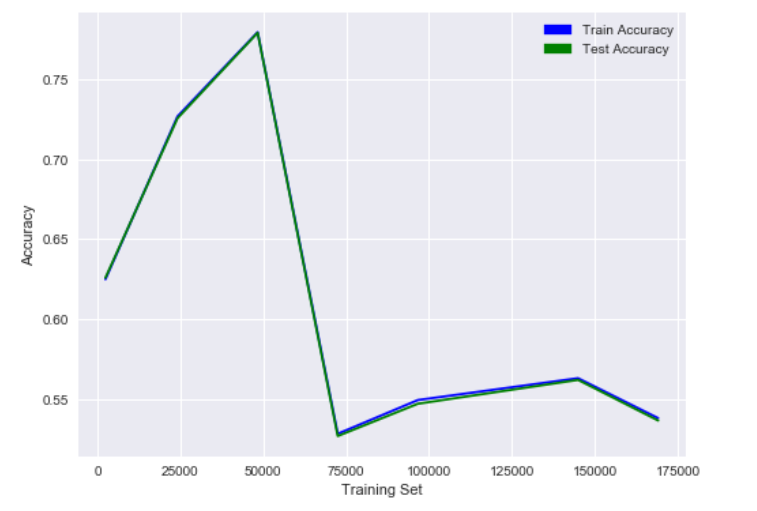
**Sigmoid activation performance:**To plot this learning curve, we will be running the ANN model twice with sigmoid function for different sizes of the training set:

We can see from the graphs that the output of this model is not consistent. Looking at the trends we can say that the model is heavy influenced by the bias in the data. In the first case we see a lot of variance while in the second graph we do not see any impact of variance on the accuracy. Although these results are not consistent, they seem to generalize when dataset is larger enough. So, if we do not get better results from the other activation functions, this model can be used to successfully for classification.

Next, we run cross fold validation on 90% of the training set with 5 folds and check the average accuracy of our validation sets. This will give use a good idea of how well this model will perform on the churn dataset as all the datapoints will be used in training and test atleast once. The average accuracy of this model using a sigmoid activation function is given below:

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 81.93% |

**Tanh activation performance:** Repeating the above experiment twice with a Tanh activation function in hidden and output layers gives the below learning curves:

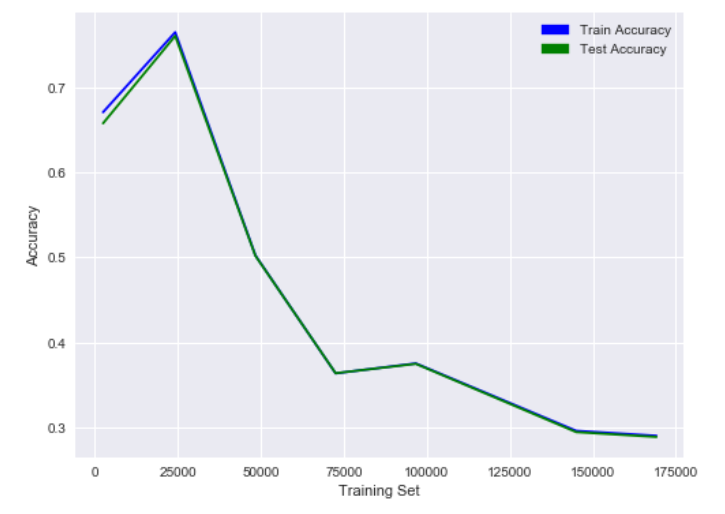
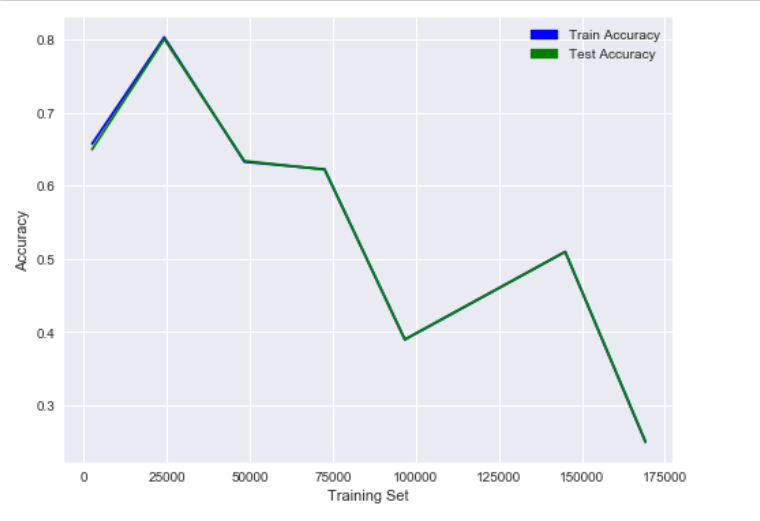


From the above curves we can see that the model is very unreliable and doesn’t generalize well. The model seems to be unable to handle the bias error. In the first graph we can see the model performing well on the dataset with low bias and low variance as the training set size increase. But in the second graph we see the bias bring down the accuracy do to underfitting. Even though the variance is low, the accuracy is bad due to the model fitting too much to the noise.

The average validation set accuracy of this model using cross validations with 5 folds and 90% of the data is:

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 49.73% |

**Relu activation performance:** Running the model twice with Relu activation function in both hidden and output layers gives the below learning curves:



From the curve we can see that the performance of the Relu activation is bad as it is unable to handle the bias in the data. As the training set size increases, we can see the accuracy decrease and the model starts to overfit due to high bias and low variance.

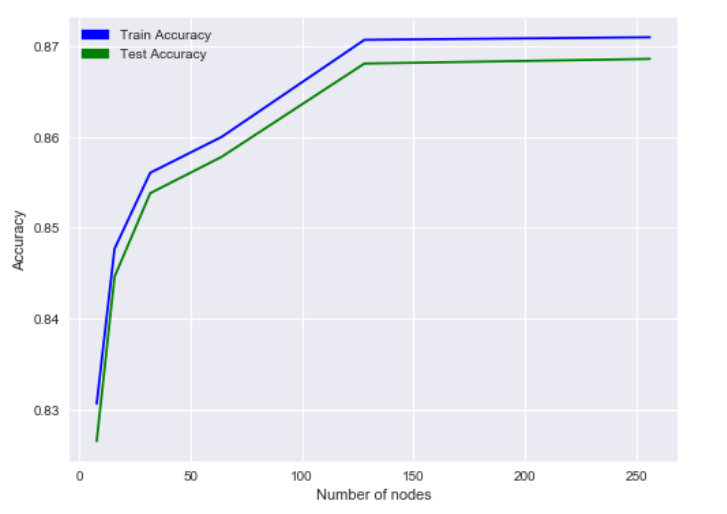
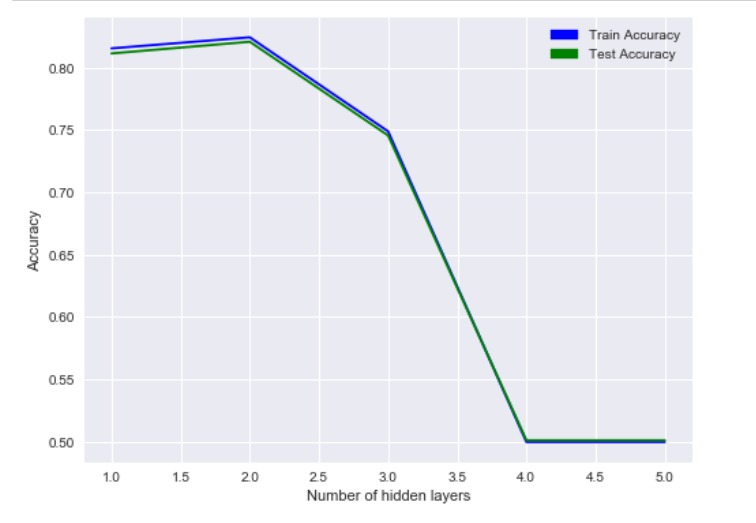
The average validation set accuracy of this model using cross validations with 5 folds and 90% of the data is

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 32.61% |

From the above tests we can see that the sigmoid function is the only activation function capable of handling the bias in this dataset.

**Experiment 2:**

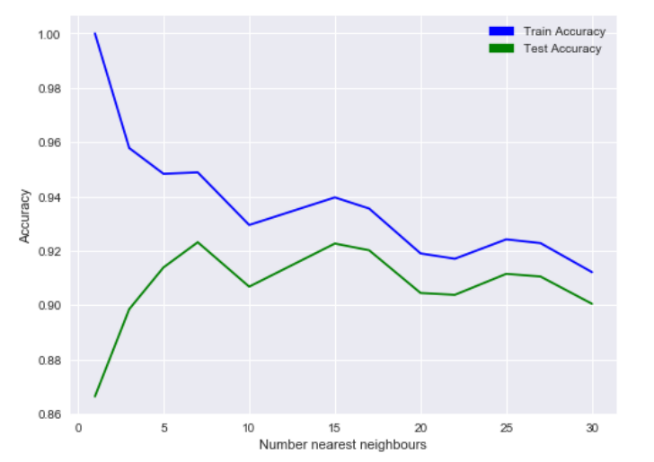
In this experiment we will be studying the impact of increasing the number of nodes and number of hidden layers in an ANN model with sigmoid activation function for the GPU dataset. We will be running the models for different number of nodes in the hidden layer and compare the training and test accuracy to select the best number of nodes for this activation function. We will repeat the same by varying the number of hidden layers to get the best values. Plotting the test and train accuracy of the above experiments we get the following graphs.

From the first plot we can see that the data starts the bias becomes low as the number of nodes becomes great than or equal to 128. There doesn’t seems to be any issue with handling variance even when number of nodes is less than 128. From the second graph we can see that increasing the number of hidden layers to above 2 causes the model to over fit due to high bias and low variance. So we can conclude that to get the best results for sigmoid activation we should use a model with 128 nodes and 1 or 2 hidden layer.

**Experiment 3:**

In this experiment we will be classifying GPU loads using KNN. We will be running KNN with varying number of neighbors and we will be plotting them against test and train accuracy to pick the best model. We will be using the default weigh function which will weigh all points in the neighborhood equally. Plotting number of neighbors versus train and test accuracy we get the below plot.



From the graph we can we that the bias is high when the number of neighbors is below 7 and decrease as we increase the number of neighbors. We can also see that the variance decreases as we increase the number of neighbors. From these results we can conclude that 25 neighbors give us the best results for this dataset.

Computing the average validation set accuracy of this model with 25 neighbors using cross validations with 5 folds and 90% of the data we get:

|  |  |
| --- | --- |
| Mean Validation Set accuracy | 91.41% |

**Conclusion:**

Comparing the results from the best models from assignment 3 and assignment 2 we get the below table:

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| ANN with sigmoid activation | 81.93% |
| KNN with 25 neighbors | 91.41% |
| SVM with RBF kernel | 95.67% |
| Decision tree | 98.91% |
| Gradient boosted tree | 82.64% |

From the above results we can see that the ANN is outperformed by all the models from assignment 2. And Even though KNN results are better they cannot compete with the results from the decision tree. So, we conclude that the decision tree is still the best classifier for this dataset.