**Machine Learning – Assignment 3**

Chinmaya R Naguri

8. (a) The dataset given is about Reuters articles corresponding to *corporate acquisitions.* The feature values correspond to the count of occurrences of the different words. Using machine learning we have to predict if a new article given belongs to our topic or not.

(b) The Filter methods which we can use for this kind of dataset are: Pearson Correlation Coefficient, Signal to Noise Ratios, T-Test. For each method we need to observe the distribution of the feature. i.e By Mean of the feature values in different classes, their standard deviations etc. We can easily compute these and make an estimate of how reliable a feature is in prediction.

As our dataset is with binary classification and we have huge number of features (i.e. 20000 features) Mutual Information method is not suitable to use. And also there are huge number of features which are simply empty i.e without at least one occurrence in any of the examples. And Chi Square is not suitable here because we have to know the possible range for the features. But here, we can’t limit the range of the feature values.

I have ranked the features using the three filter methods. This is done using *RankFeatures.java* file in the *FeatureSelection* package in the submitted code. This generates *.rank* files for each filter method so that we can use them while learning & prediction.

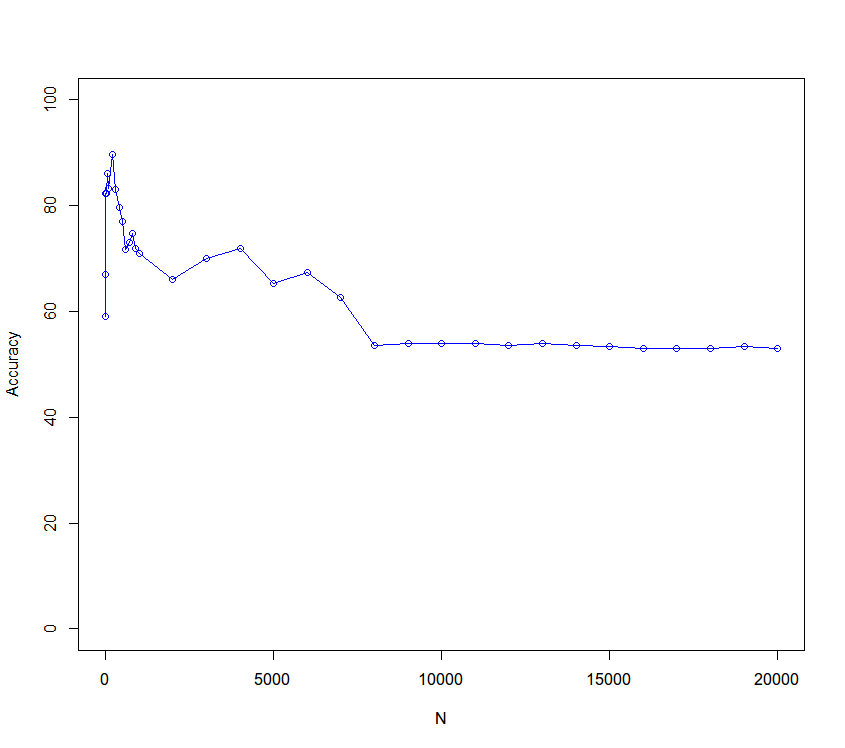
Now, we have to create a new set of training files for both the Learning Algorithms, as our features vary now after ranking (we need only top N features for learning). So, to generate these files use *CreateSVMInputFiles.java* & *CreateWekaInputFiles.java.* While generating these files normalize the feature vectors and save them to the files. Here we are taking top N features based on their ranks. Where N = {1, 5, 10, 20, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000, 12000, 13000, 14000, 15000, 16000, 17000, 18000, 19000, 20000}.

Using these files as inputs we start learning now. In linear SVM we use capacity parameter, C=1. Use shell script *training.sh* to automatically train for all top N features and saves the .model files. K-NN need not learn anything as it compares a test point during prediction and decides its class. For K-NN we will test for k = {1, 5, 10}.

(c) Now, we will test on the validation data and show the Accuracy vs top N feature selection for each learning algorithm:

**Pearson-Correlation Coefficient:**

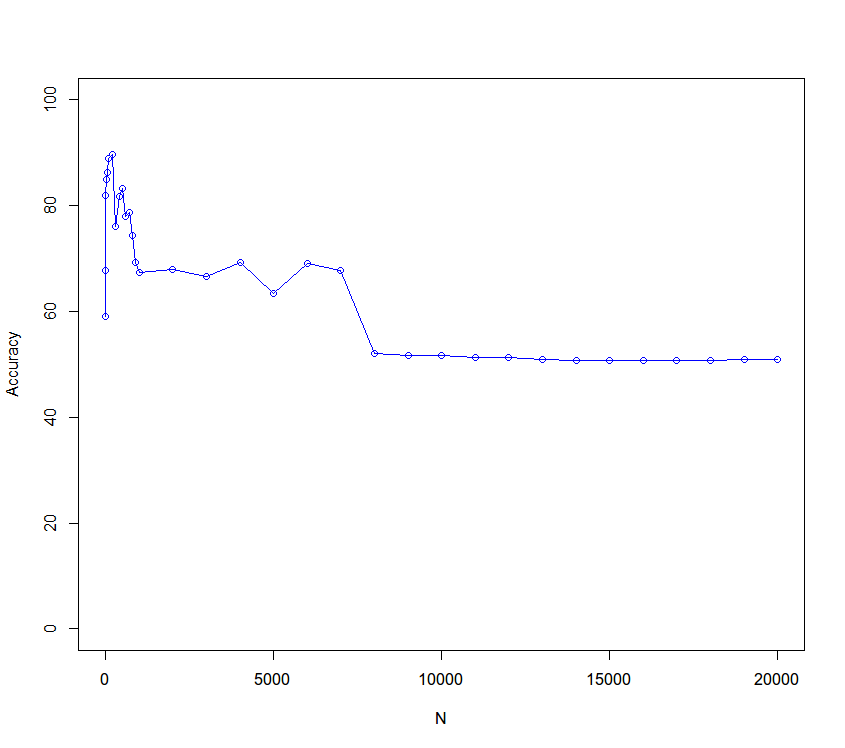
For K-NN when K = 1:



Best Accuracy is reached at N = 200 with 89.67% i.e 269 correctly classified in 300 examples.

And Second Best is at N=50, with 86% (258/300)

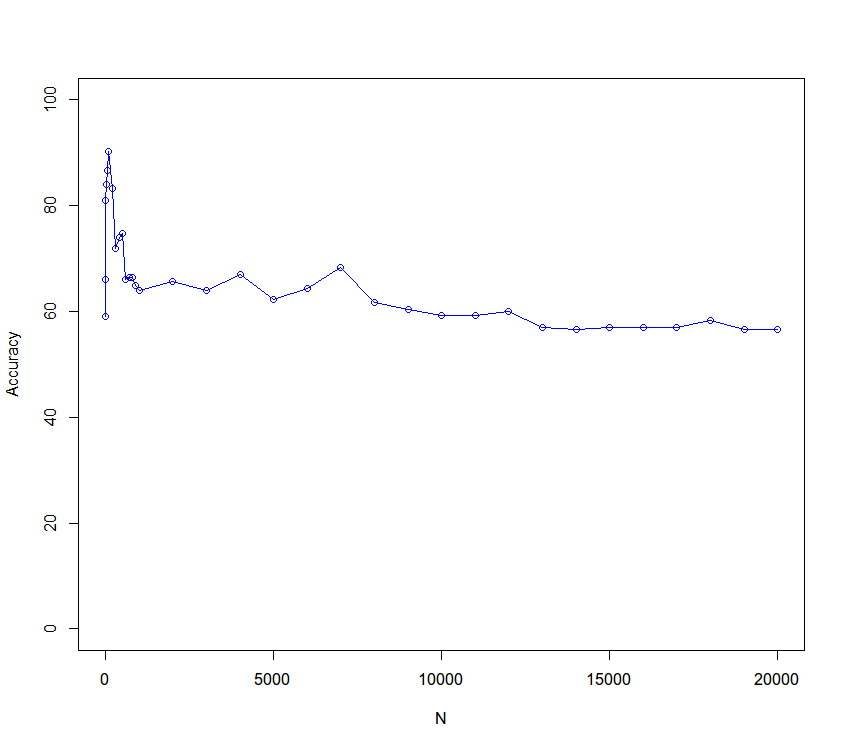
For K-NN when K = 5:



Best Accuracy is reached at N = 200 with 89.67% i.e 269 correctly classified in 300 examples.

And Second Best is at N=50, with 86.33% (269/300)

For K-NN when K = 10:

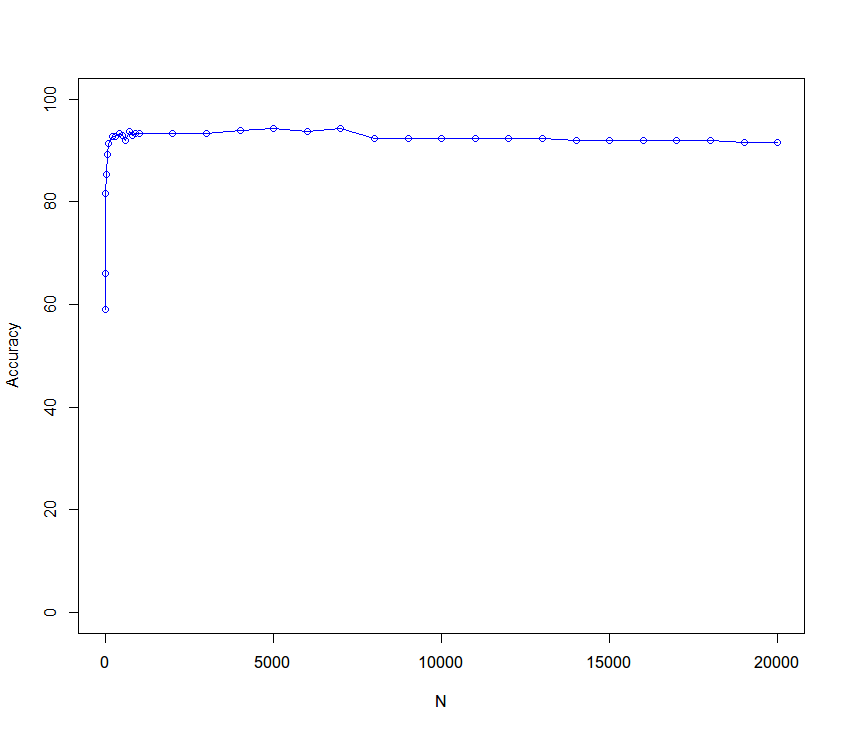


Best Accuracy is reached at

N = 100 with 90.33% i.e 271 correctly classified in 300 examples.

And Second Best is at N=50, with 86.66% (260/300)

For Linear SVM (with C=1):



Best Accuracy is reached at

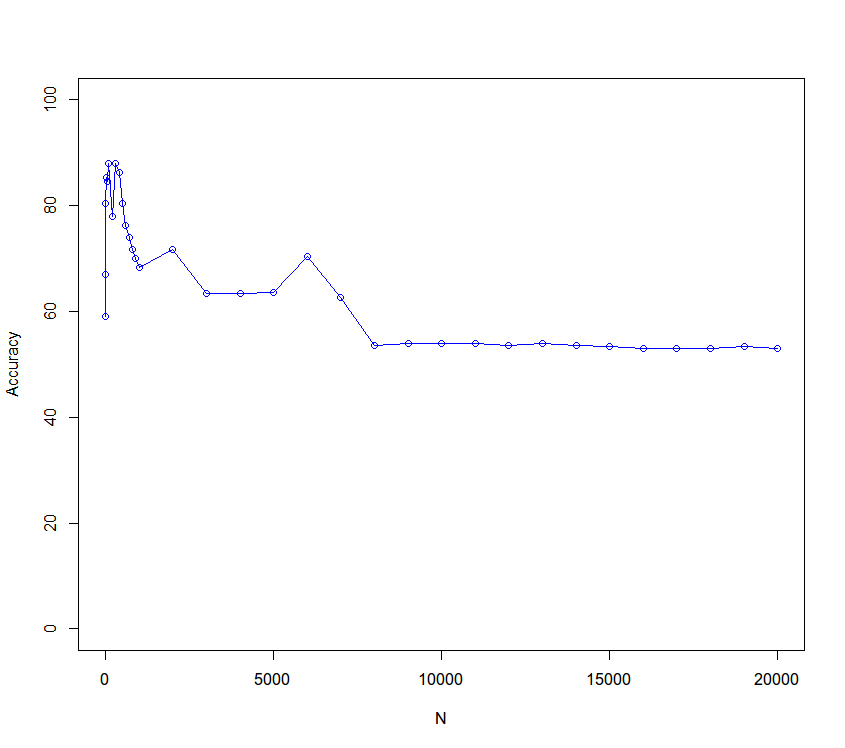
N = 5000 with 94.33% i.e 287 correctly classified in 300 examples.

And Second Best is at N=4000, with 94% (282/300).

I observed that Linear SVM has got the best performance overall for Pearson Correlation coefficient & in K-NN algorithm with 10 nearest neighbors our model performs well.

**Signal to Noise Ratio:**

For K-NN when K=1:

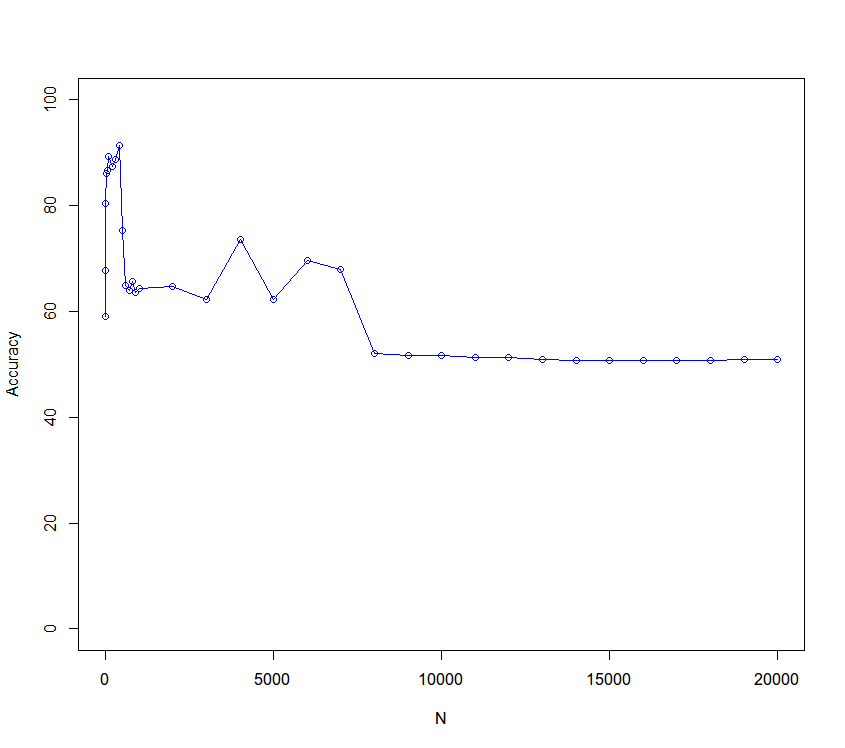


Best Accuracy is reached at

N = 400 with 86.33%.

And Second Best is at N=20, with 85.33%.

For K-NN when K=5:

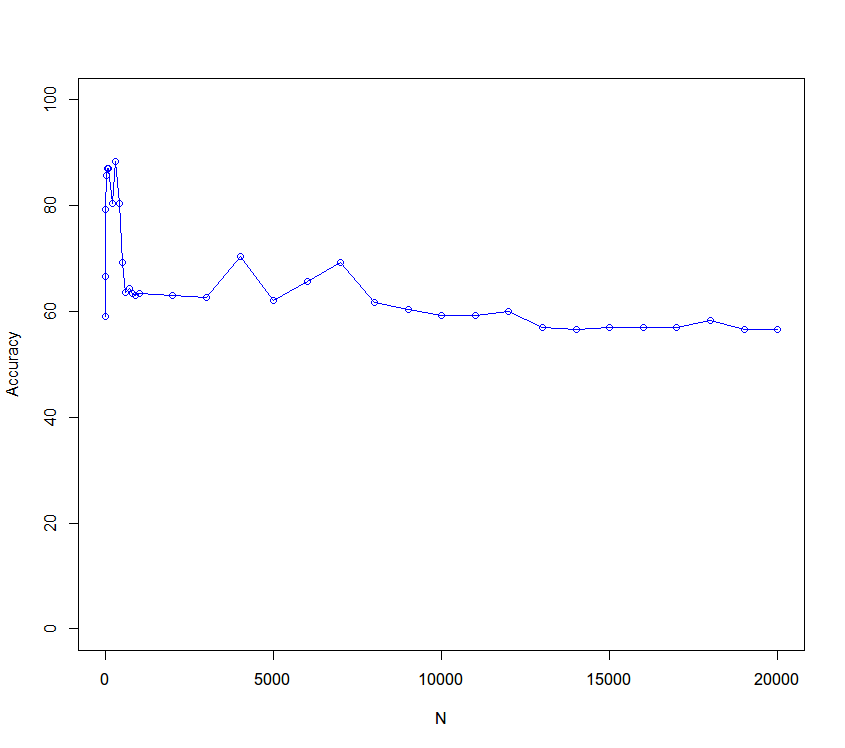


Best Accuracy is reached at

N = 400 with 91.33%.

And Second Best is at N=300, with 88.67%.

For K-NN when K=10:

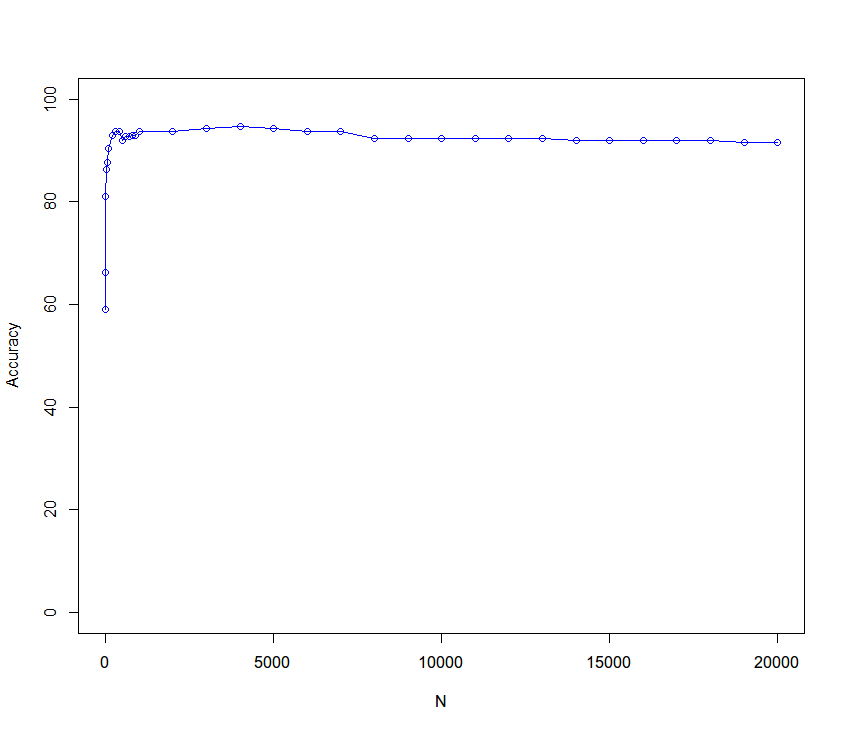


Best Accuracy is reached at

N = 300 with 88.33%.

And Second Best is at N=20, with 85.67%.

For Linear SVM (with C=1):



Best Accuracy is reached at

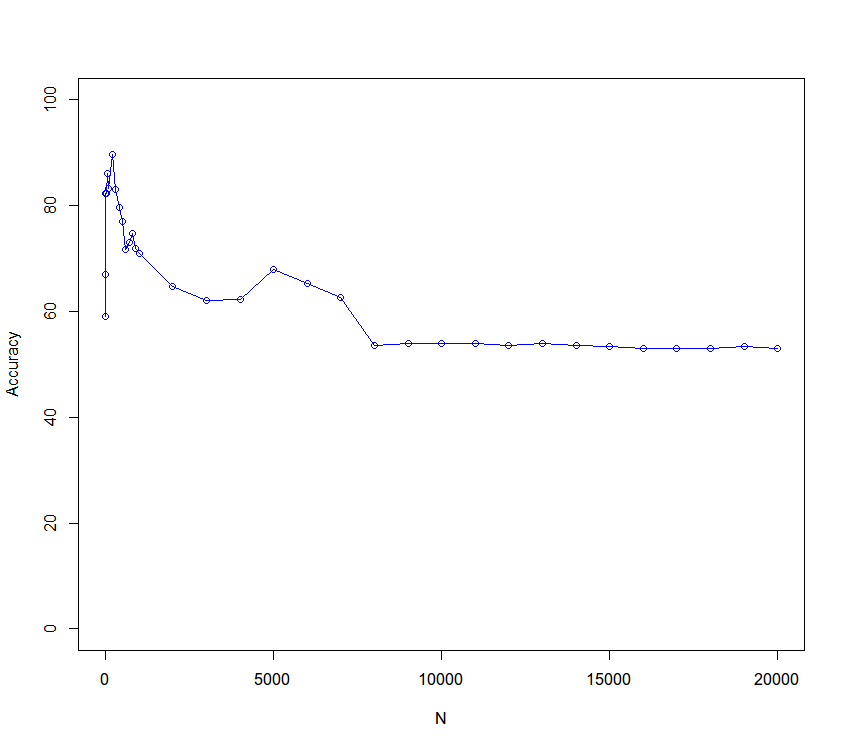
N = 4000 with 94.67% i.e 284 correctly classified in 300 examples.

And Second Best is at N=3000 & 5000, with 94.33% (284/300).

I observed that Linear SVM has got the best performance overall for Signal To Noise Ratio & in K-NN algorithm with 5 nearest neighbors our model performs well.

**T-Test:**

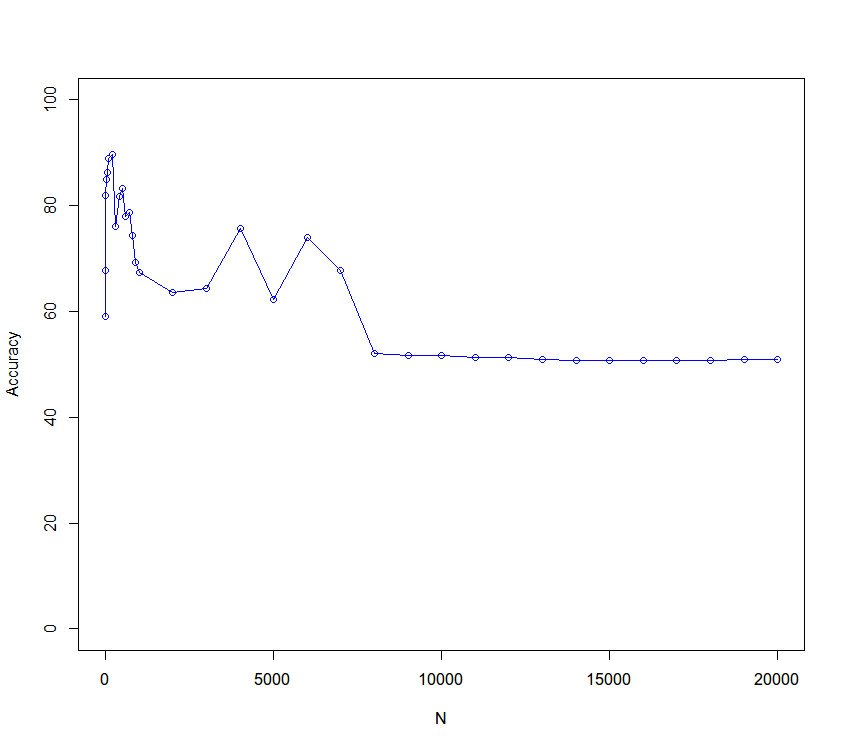
For K-NN with k = 1:



Best Accuracy is reached at

N = 200 with 89.67%.   
And Second Best is at N=50, with 86%

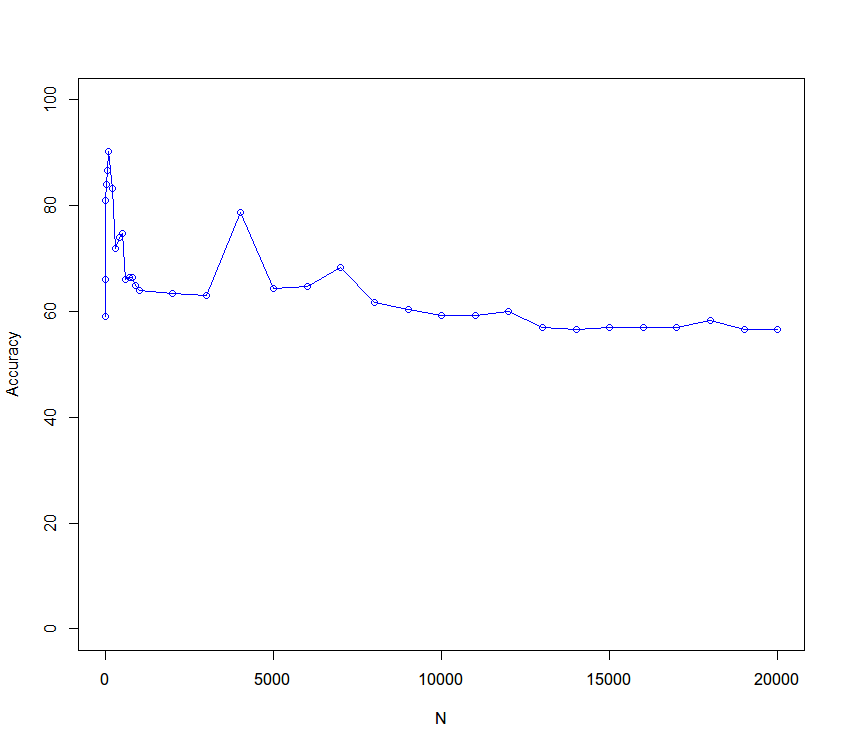
For K-NN with k=5:



Best Accuracy is reached at

N = 200 with 89.67%.   
And Second Best is at N=100, with 89%

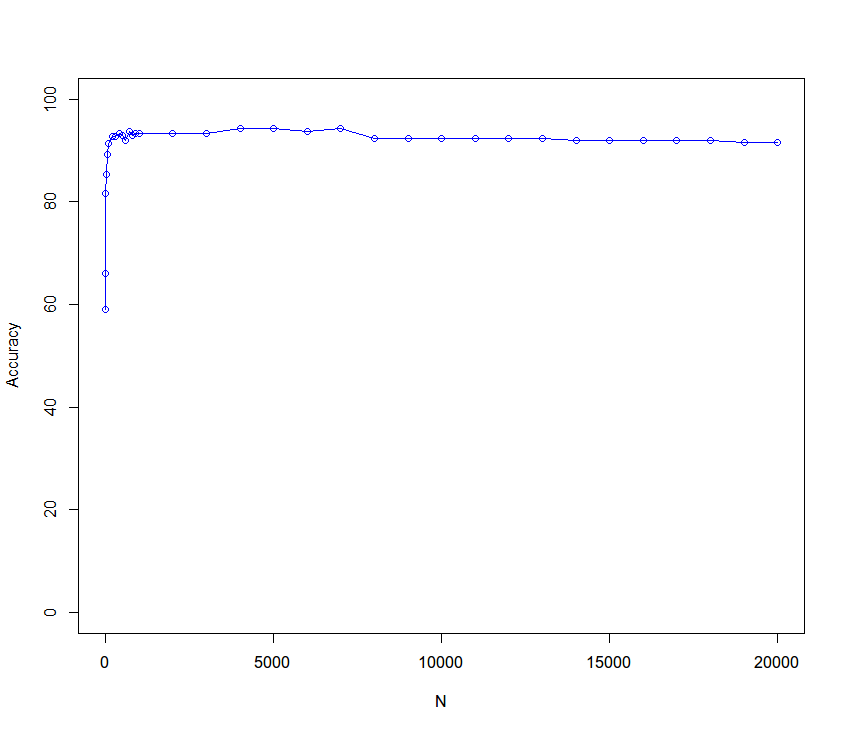
For K-NN with k=10:



Best Accuracy is reached at

N = 100 with 90.33%.   
And Second Best is at N=50, with 86.67%.

For Linear SVM (with C=1):



Best Accuracy is reached at

N = 4000 & 5000 with 94.33% i.e. 283 correctly classified in 300 examples.

And Second Best is at N=6000, with 93.67% (281/300).

I observed that Linear SVM has got the best performance overall for T-Test & in K-NN algorithm with 10 nearest neighbors our model performs well.

So, after looking into all the graphs & observations we can say that SVM’s have performed very well overall in comparison to K-NN, which is naïve in classification as it just looks for nearby members. And In K-NN models when k = 10, the performance is little better than when k = 5.

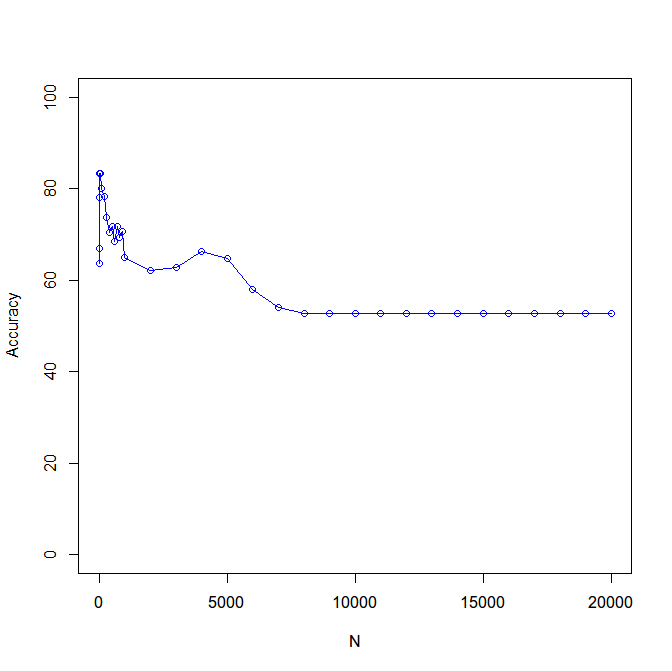
And we can observe that including too many features may not help the prediction but it degrades the performance. That is why we should be really careful in feature selection.

**NOTE**: The graphs for SVM are flat for when N > 3000. This is due to most features are zero and thus the SVM algorithm ignores them completely and all the models after N > 3000 has fairly similar support vectors. So, the classification accuracies don’t vary considerably.

(d) Now, we will rerun all the above accuracy tests for each Filter Method & Algorithm using data which is not normalized.

**Pearson-Correlation Coefficient**:

For KNN with K=1:

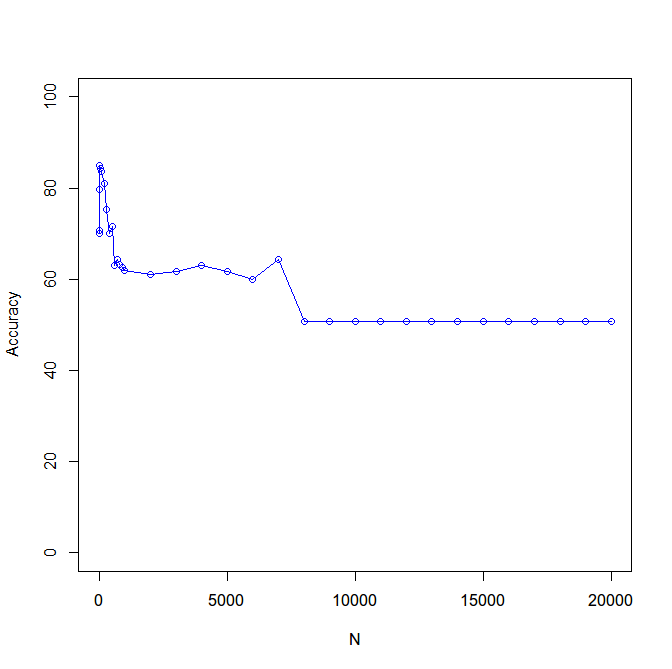


Best Accuracy is reached at

N = 20 & 50 with 83.34%.

And Second Best is at N=100, with 80%.

For KNN with K=5:

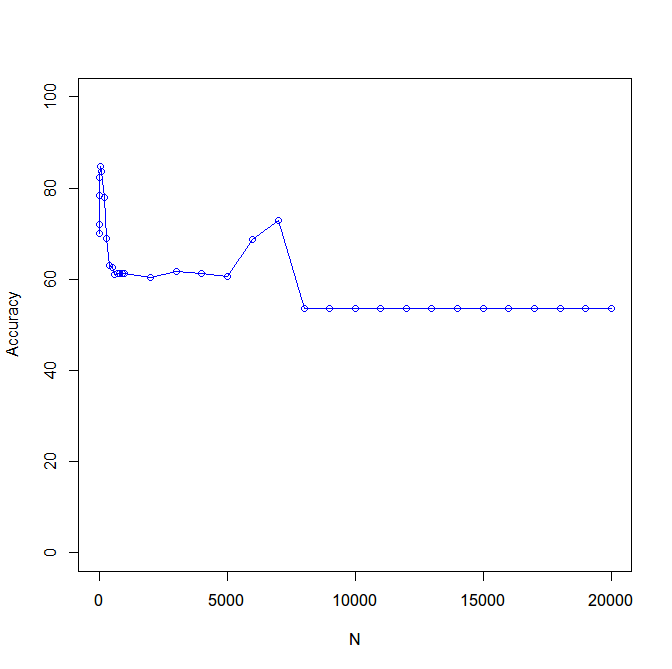


Best Accuracy is reached at

N = 20 with 85%.

And Second Best is at N=50, with 84.33%.

For KNN with K=10:

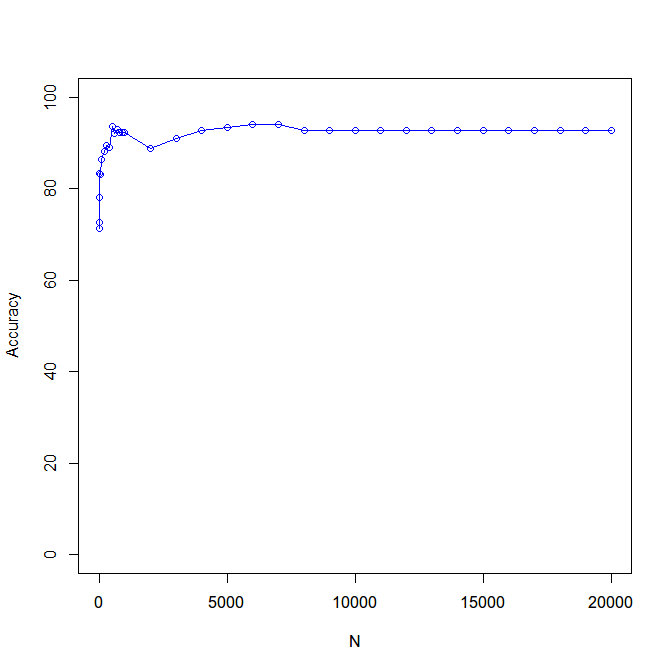


Best Accuracy is reached at

N = 50 with 84.67%.

And Second Best is at N=100, with 83.67%.

For Linear SVM with C=1:



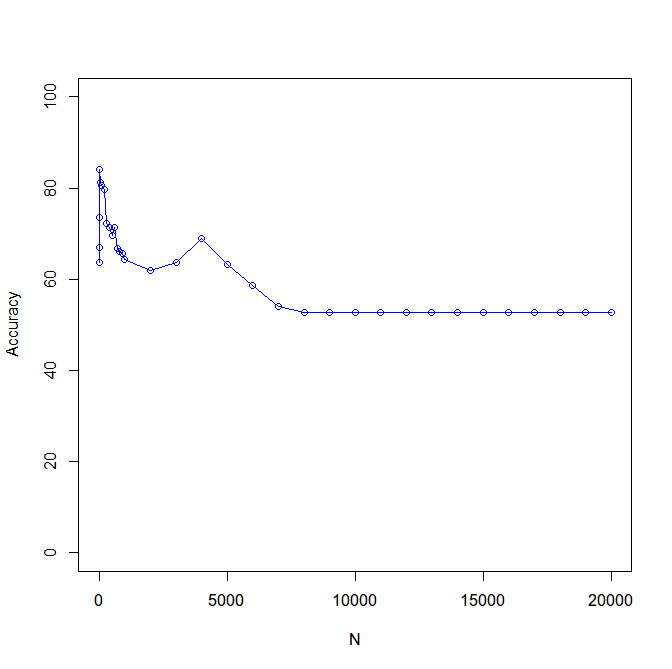
Best Accuracy is reached at

N = 6000 & 7000 with 94.00% i.e 282 correctly classified in 300 examples.

And Second Best is at N=5000, with 93.33% (280/300).

**Signal To Noise Ratio:**

For KNN with K=1:

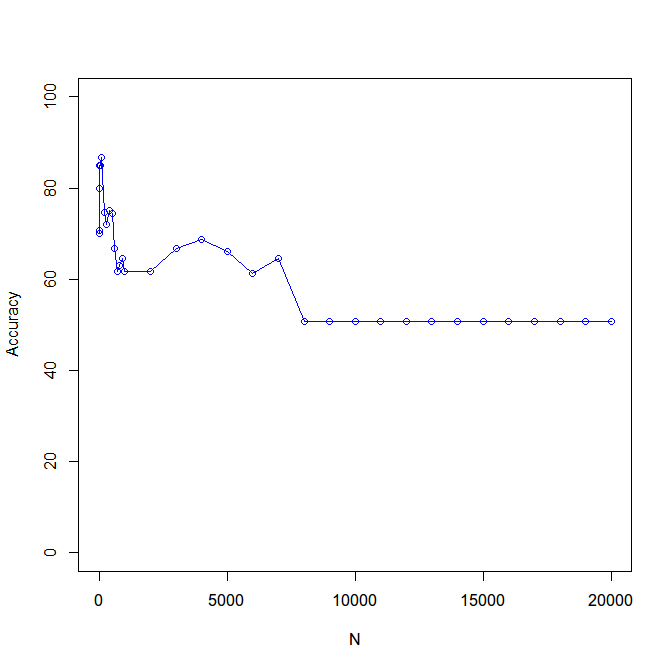


Best Accuracy is reached at

N = 20 with 84%.

And Second Best is at N=50, with 81.33%.

For KNN with K=5:

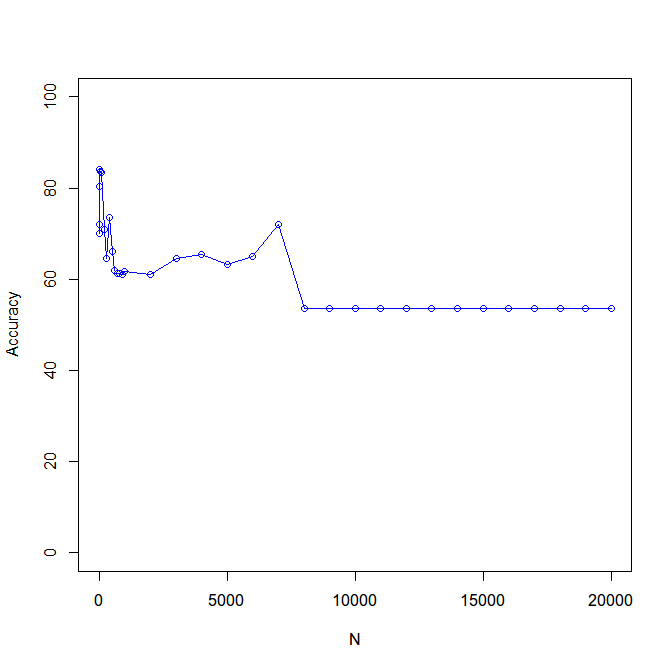


Best Accuracy is reached at

N = 100 with 86.67%.

And Second Best is at N=20 & 50, with 85%.

For KNN with K=10:

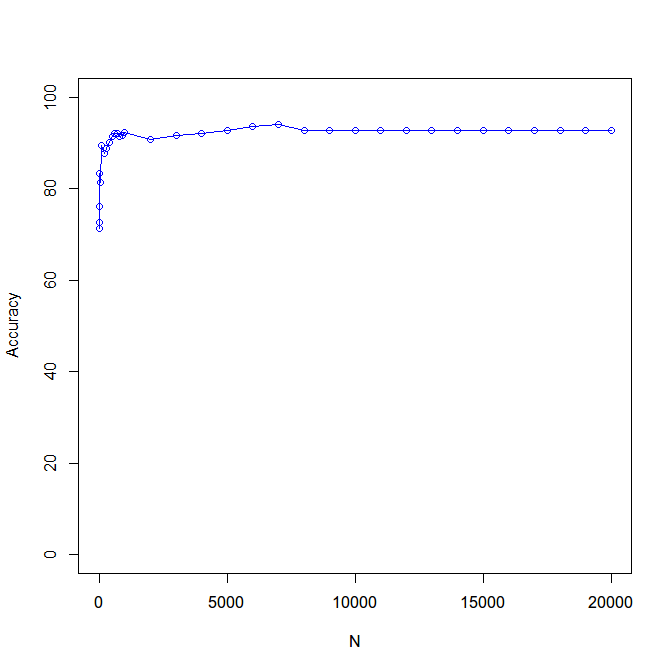


Best Accuracy is reached at

N = 20 with 84%.

And Second Best is at N=50, with 83.67%.

For Linear SVM with C=1:



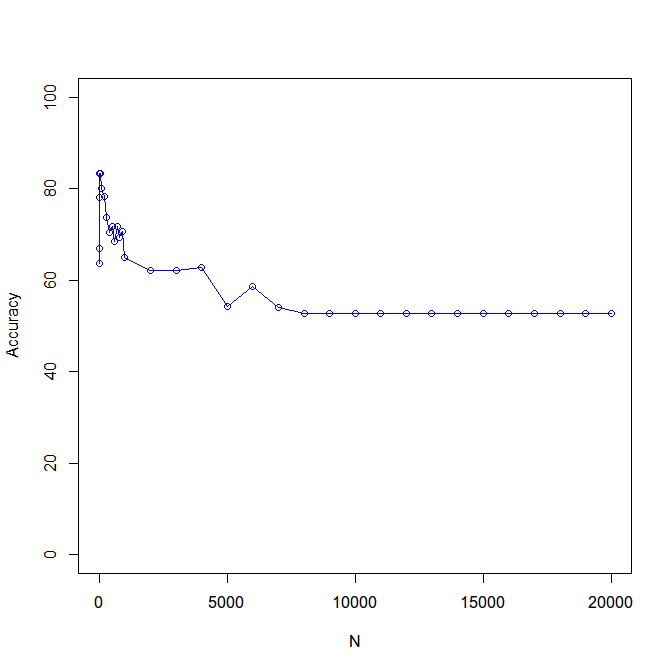
Best Accuracy is reached at

N = 7000 with 94.00% i.e 282 correctly classified in 300 examples.

And Second Best is at N=6000, with 93.67% (281/300).

**T-Test :**

For KNN with K=1:

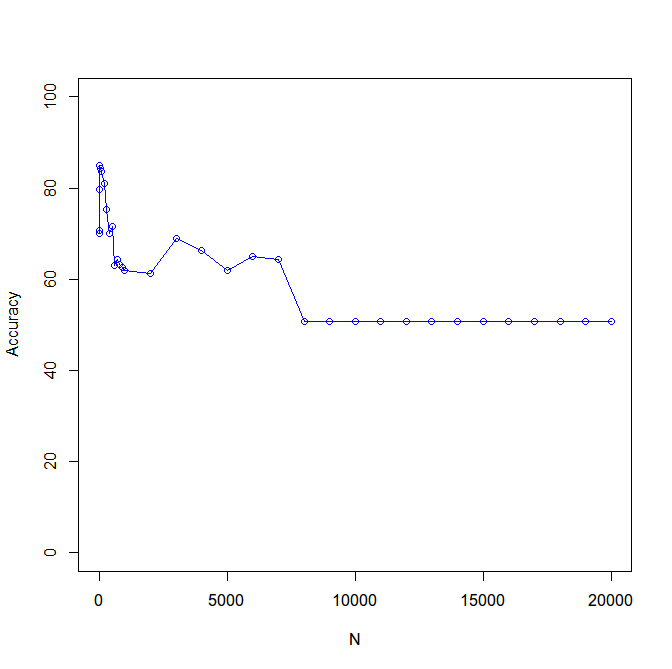


Best Accuracy is reached at

N = 20 & 50 with 83.34%.

And Second Best is at N=100, with 80%.

For KNN with K=5:

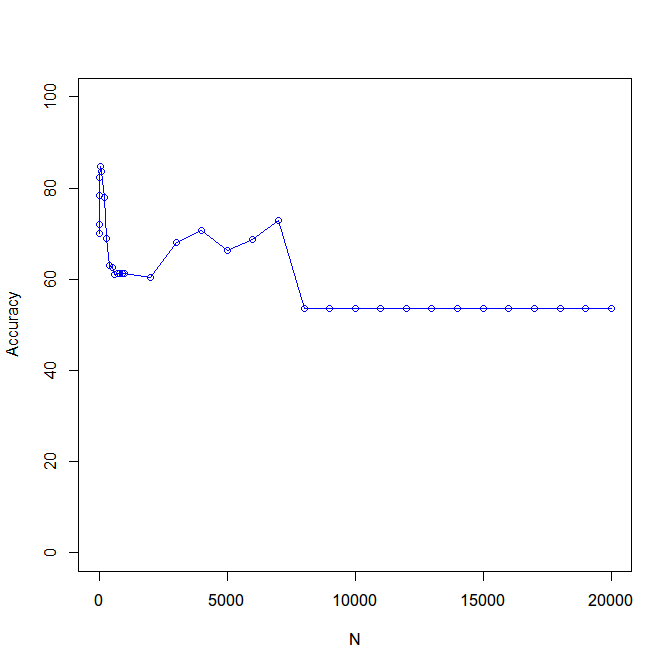


Best Accuracy is reached at

N = 20 with 85%.

And Second Best is at N=50, with 84.33%.

For KNN with K=10:

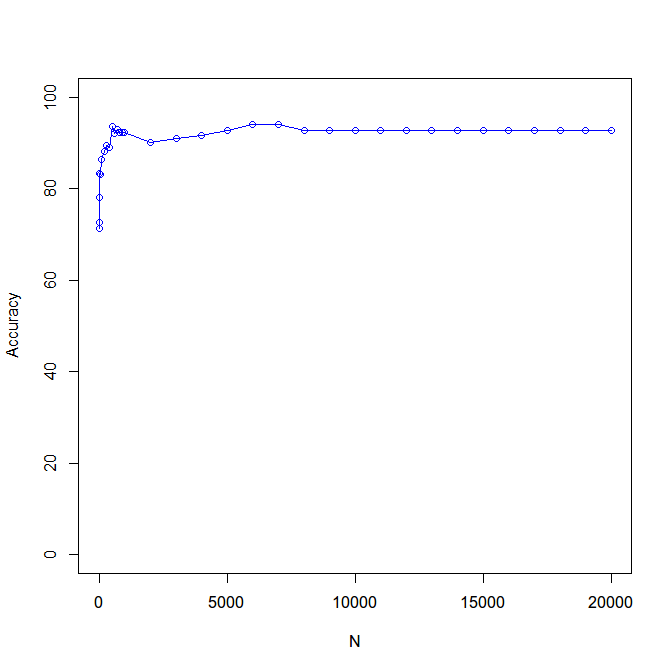


Best Accuracy is reached at

N = 50 with 84.67%.

And Second Best is at N=100, with 83.67%.

For Linear SVM with C=1:



Best Accuracy is reached at

N = 6000 & 7000 with 94.00% i.e 282 correctly classified in 300 examples.

And Second Best is at N=5000, 8000 and till 20000 with 92.67% (281/300).

I have observed that with in all the graphs in K-Nearest Neighbor, after k = 8000, The graph is flat. As there are many features whose values are zero in both training and test examples. So, they don’t make any difference in the prediction. Therefore the accuracy values are same after that threshold.

We can see the same in SVM graphs as well as discussed in the normalized section.

I also observed that there is not much of a difference in accuracy values for normalization and not normalization. Few methods have better accuracies in normalization, such as all of PCC, S2N Filter Methods & T-Test filter method with Linear SVM and where as other perform better in without normalization, such as T- Test KNN where K=1. But we know that we can ignore the accuracies here as K=1 is not a good method for classification as it just depends on only one nearest neighbor.