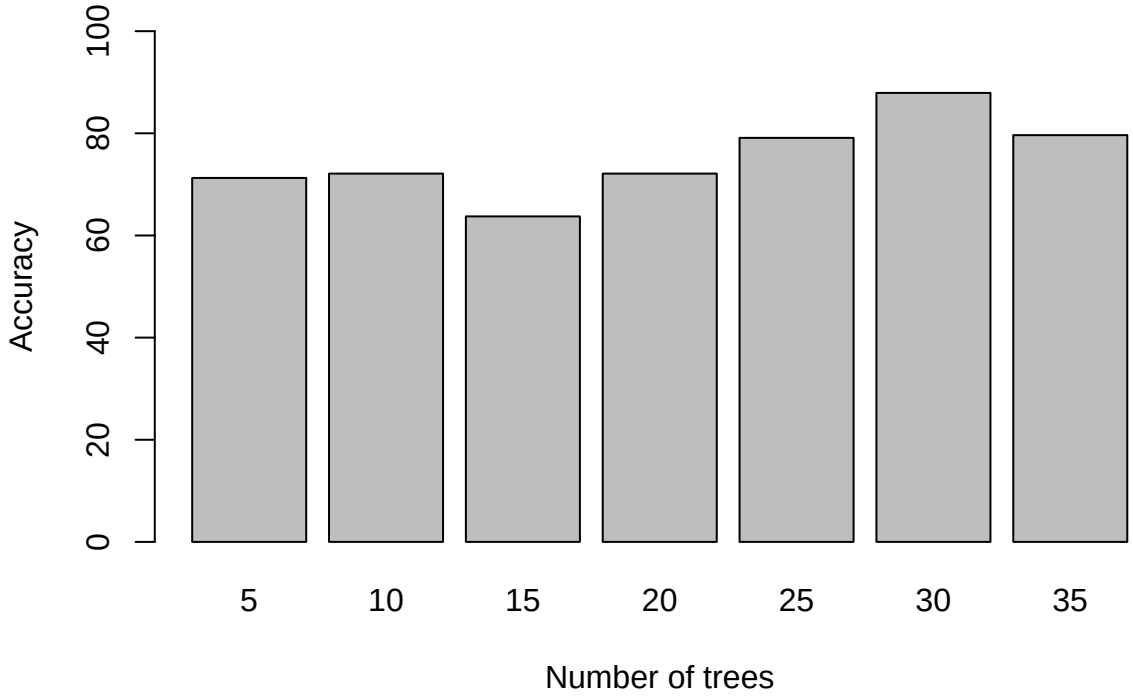
Random Forests:

We used random samples (rows) and random features as well (columns). We first split on 128 which was giving a good enough accuracy of 67, but we wanted to improve the accuracy, so we checked each value on each column and associated gini index and the one with the lowest gini index is selected. So it assures that we are taking the best value, instead of taking 128 as the split point (fixed threshold). Also the max depth of the tree we went till 40. When we took it to 80, there was no significant improvement in accuracy, and hence we set it to 40 and the min leaf we set to 1 even though it has a chance of overfitting because it gives a good accuracy. We created 20 decision trees randomly and we took a vote among all of them to classify each image.

```
##   Number.of.trees Training.time Testing.time Accuracy
## 1               5         1m 1s       0.318s    71.26
## 2              10         2m 8s       0.383s    72.11
## 3              15        3m 10s       0.453s    63.73
## 4              20         4m 8s       0.514s    72.11
## 5              25        5m 11s       0.577s    79.11
## 6              30        5m 59s       0.650s    87.91
## 7              35        6m 58s       0.712s    79.64
```

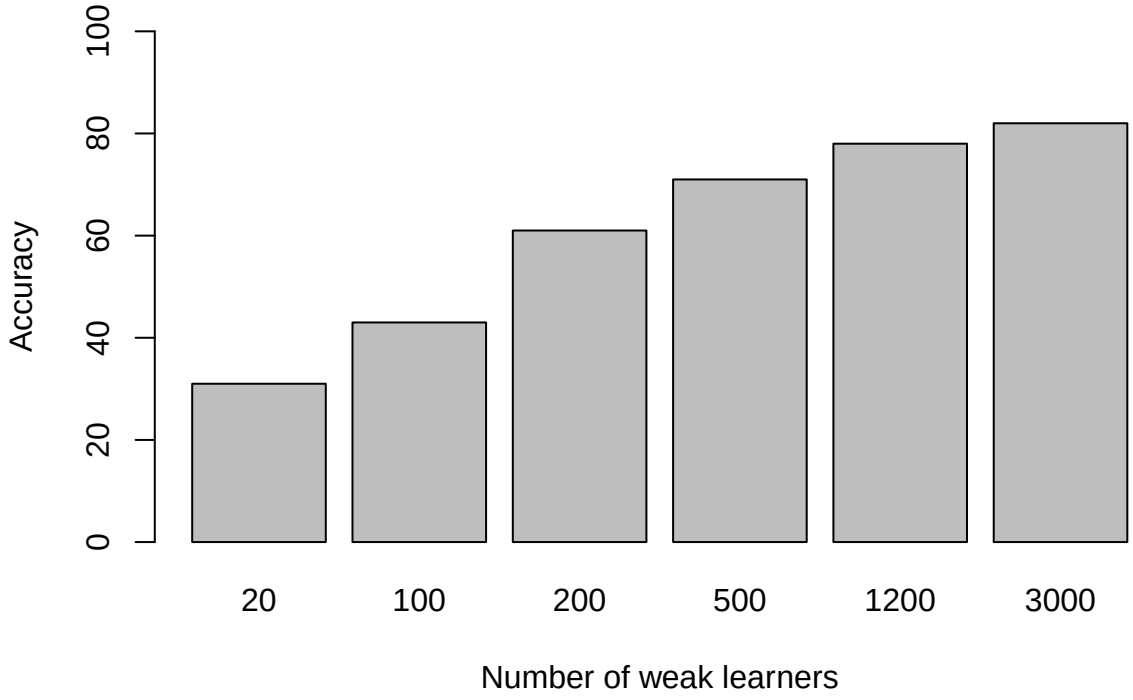


AdaBoost:

In Adaboost, we used pair of columns as a stub where our stub (weak learner) hypothesizes as subtraction of values at the given column pairs, and if it is less than 0, we label it as positive, otherwise negative. Since Adaboost works only for binary classification, we had to improvise to make 4 one-versus-all classifiers, and using these classifiers and the learned parameters a_i, we calculated the H values for a single image in 4 classifiers and the one with highest positive value was given it's respective corresponding class label. If none of the classifiers were able to classify, then we give it a random class. We started with 20 learners giving an accuracy of 31%, going up to 1,200 learners giving an accuracy of 78% but incorporating more learners ~3000 didn't provide significant increase in accuracy. And also considering the time trade-off, we stopped at 3,000 learners.

```
##   Number.of.weak.learners Training.time Testing.time Accuracy
```

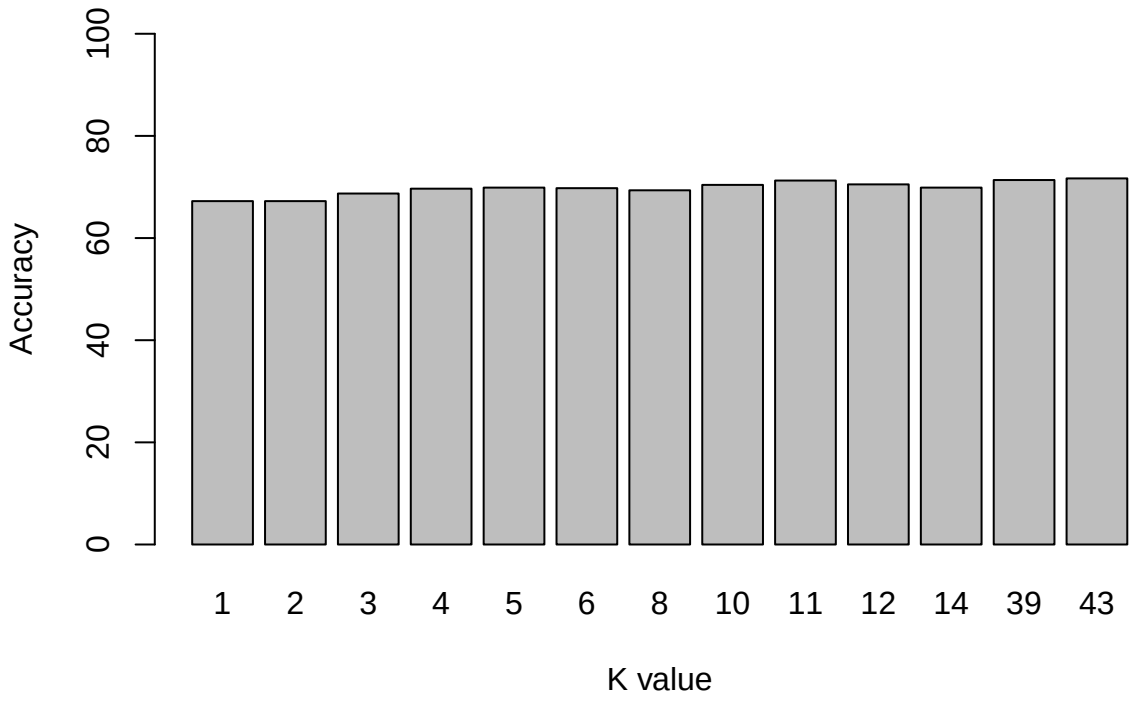
```
## 1                    20          10s          57s       31
## 2                   100           1m        1m 3s       43
## 3                   200        2m 27s       1m 11s       61
## 4                   500        3m 42s       1m 19s       71
## 5                  1200         5m 3s       1m 24s       78
## 6                  3000       16m 11s       1m 30s       82
```



K-Nearest Neighbours:

To find the nearest neighbours, we used the Euclidean distance. We analyzed the accuracy for different values of K and found out the best performance at K=11. Though K=43 gives a slightly better accuracy, the extra time is not worth the increase in accuracy.

```
##      K Testing.time Accuracy
## 1    1       5m 53s    67.23
## 2    2           6m    67.23
## 3    3       6m 18s    68.72
## 4    4       6m 14s    69.67
## 5    5       6m 22s    69.88
## 6    6       6m 41s    69.77
## 7    8       6m 34s    69.35
## 8   10       6m 34s    70.41
## 9   11       6m 52s    71.26
## 10 12        6m 5s     70.51
## 11 14       6m 21s    69.88
## 12 39       7m 32s    71.36
## 13 43       7m 47s    71.68
```



Among all classifiers, taking time and computation into consideration, we would suggest AdaBoost to a client even though random forest gives high accuracy in some cases but the accuracies are erratic and are subject to a lot of parameter tuning.