

Assignment 4 Report

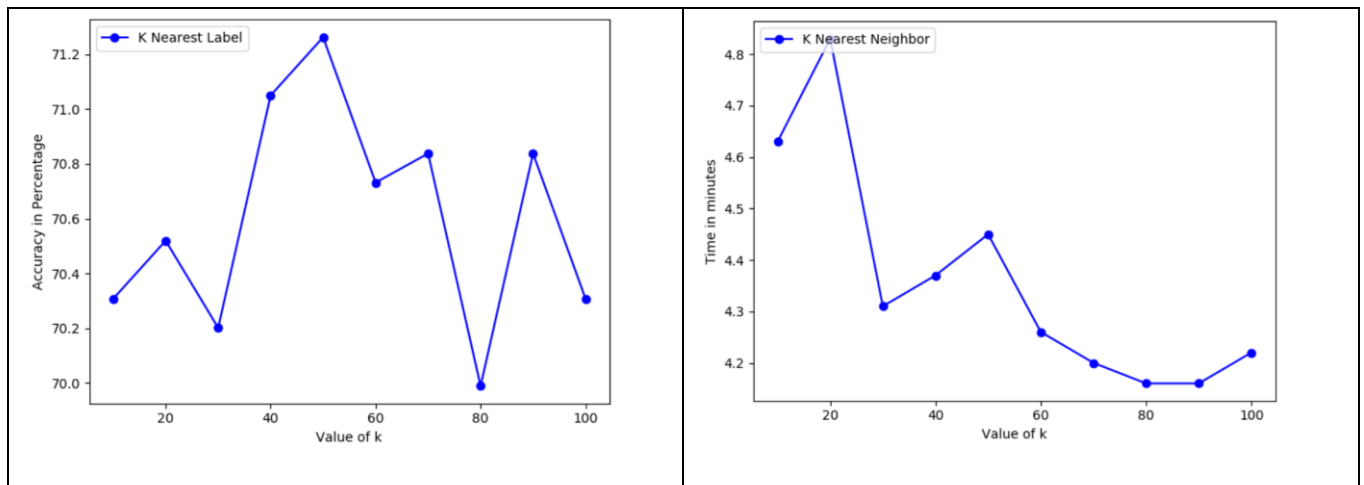
K-Nearest Neighbor

1. Classification accuracy and running time as a function of K

Value of K	Accuracy in percentage	Running time in minutes
10	70.308	4.36
20	70.520	4.68
30	70.201	4.37
40	71.050	4.36
50	71.262	4.37
60	70.732	4.37
70	70.838	4.86
80	69.989	4.97
90	70.838	4.49
100	70.308	4.33

Table 1

Graph of classification accuracy and running time as a function of K



Therefore, from the above graph best value of K = 50

2. Performance depending upon the training dataset size when K=50

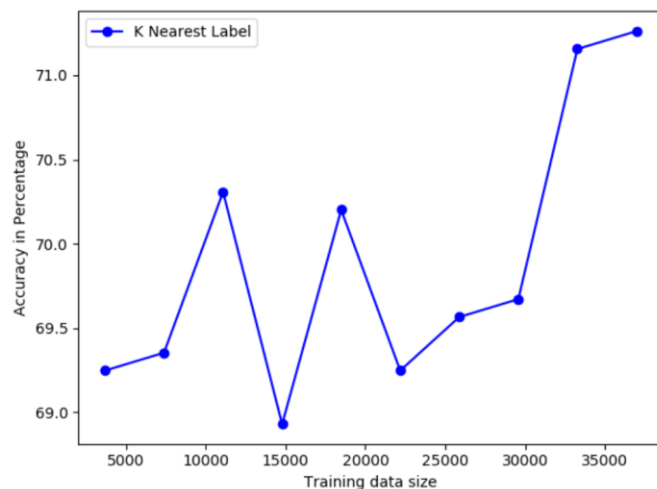
We have 36,976 entries or train vectors in our train-data.txt file which is being used to train the classifier. To determine classification accuracy as well as the running time with respect to the training data size we divide the training-data size into 10 intervals with each interval size greater than the previous interval by 3697 entries. We observe how the classification accuracy and running time is affected. We chose the best value of K=50.

The following observations are made:

Training data size	Accuracy in percentage	Time taken in minutes
3697	69.247	0.4
7394	69.353	0.81
11091	70.308	1.22
14788	68.929	1.65
18485	70.201	2.06
22128	69.247	2.49
25879	69.565	2.9
29576	69.671	3.33
33273	71.156	3.7
36970	71.262	4.11

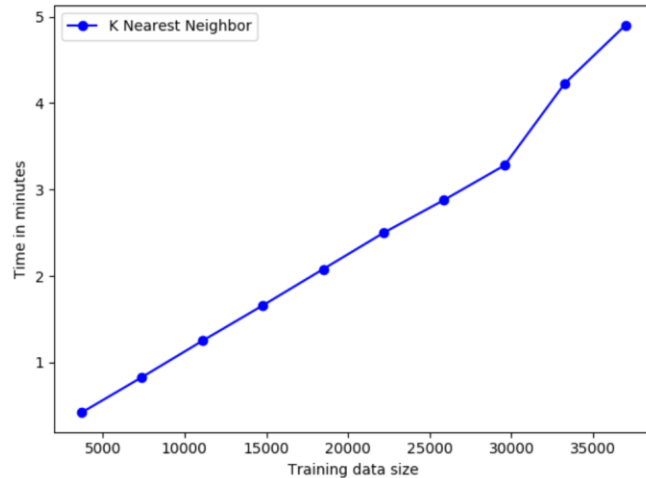
Table 2

Classification accuracy with respect to the training data size



Although, we can see from the graph above that with increase in the training data size we are getting very small variable difference in classification accuracy. However after a certain amount of training data size the accuracy tends to increase i.e. in the graph we can see that after 30,000 entries there is a linear increase in the classification accuracy. Therefore, we can deduce that the accuracy is directly proportional to the size of the training data. Greater the size of training data better the chance of getting a more accurate classification.

Running time of the algorithm with respect to the training data size



By looking at the graph above we can clearly say that running time of the algorithm is directly proportional to the training data size. In other words, greater the amount of training data, more the amount of computations the algorithm performs and hence greater the running time.

3. Patterns in the errors

Images classified incorrectly :

test/10196604813.jpg		
test/10304005245.jpg		

test/10351347465.jpg		
test/10352491496.jpg		
test/10484444553.jpg		
test/10577249185.jpg		

Observation:

Therefore, looking at the table above we can see a pattern that images which are either black and white or contain high levels of contrast (dark images with bluish and blackish tones) are classified incorrectly.

These images do not have clear boundary of separation between different components and have overlapping colors.

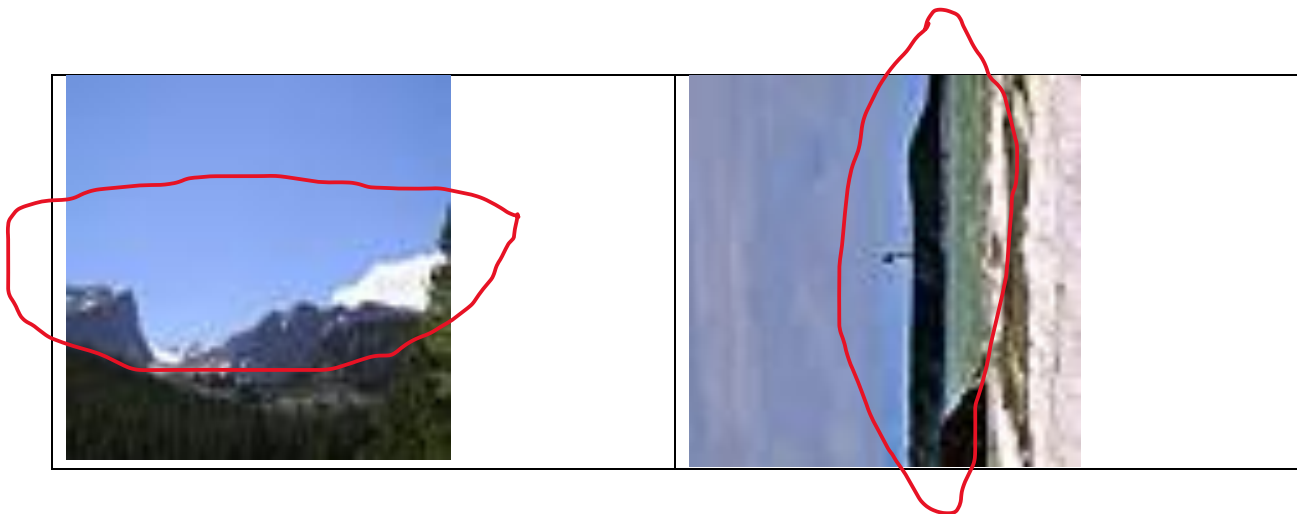
Images classified correctly:

test/10008707066.jpg		
test/10099910984.jpg		

test/10107730656.jpg		
test/10161556064.jpg		
test/102461489.jpg		

Observation:

We can see a clear pattern in the images that are classified correctly. The images which have two distinct components with separated colors such as sky and the land are classified correctly. Also these images do not have very dark colors and have less contrast.



The images classified correctly also have a clear boundary of separation between two distinct components in the image as shown in the two images above.

ADABOOST

We have taken 200 weak decision stumps

We have implemented one VS one in adaboost. The training data has been converted into 6 different combination,

1. Containing rows where label = 0 or 90
2. Containing rows where label = 0 or 180
3. Containing rows where label = 0 or 270
4. Containing rows where label = 90 or 180
5. Containing rows where label = 90 or 270
6. Containing rows where label = 180 or 270

Each of the 6 combinations calls the adaboost function where the actual algorithm is written

There are 192C2 number of possible stumps, we have chosen randomly 500 stumps as that was the one that gave us the highest accuracy.

The decision stumps have been taken as given in the pdf, A comparison is made between 2 features and then depending on the results we assign it to one of the two classes.

Initially we assign the weight $W=1/N$, where n = Num of training points

And $k=0$

The hypothesis function is called which does the comparison and returns a list containing +1 and -1 (-1 if the value on left side is greater, +1 if value on right is greater)

Now the value of hypothesis is compared with the labels of all the rows.(we assume value on left to always be class-1 and value on right to always be class 1, for first model 0-> -1 90 ->+1)

If the value of hypothesis does not match with the label, we update the error

$$\text{error}=\text{error}+w[j]$$

This is done for all the rows and the final error for the hypothesis is calculated

Now If the value of hypothesis and labels match we update the weights of the matched label (we reduce it)

$$w[m] = w[m]*(\text{error}/(1-\text{error}))$$

Finally we normalize the weights

update the value of a with $a[i]=\text{math.log}((1-\text{error})/\text{error})$

REPEAT the steps for all the hypothesis

The “hypothesis” along with value of “ a ” for all the 6 combinations are stored in a pickle file

For testing, we do one Vs one for all the 6 combinations

For the first combination, we assign the sign (0 or 90) to all the rows of the data depending in the value of $a(0 \rightarrow -1, 90 \rightarrow +1)$

$$h[j] += a[i] * \text{sign}[i][j]$$

if the final value of $h[j]$ is -1 we assign it to class 0 or if it +1 to class 90

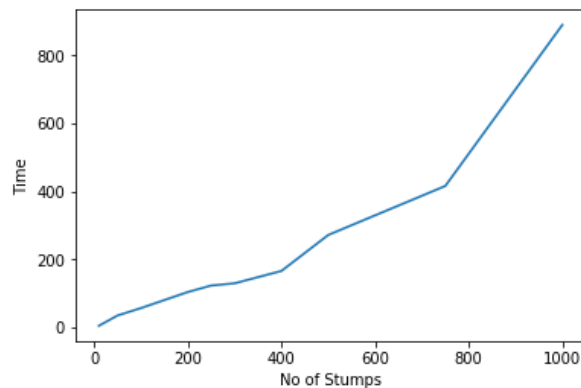
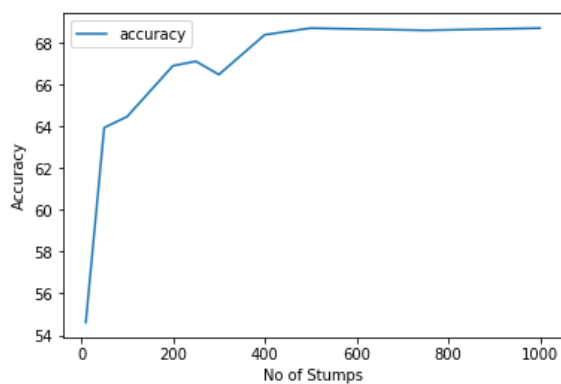
We do this for all the other combinations also.

Finally, we take a mode of the $h[j]$ for all the rows and depending on the mode we assign it to one of the 4 class 0,90,180,270.

We calculate the accuracy by comparing the predicted value with the actual labels. The percentage of properly classified rows gives the accuracy

1. Classification accuracy and running time as a function of decision stumps

No of decision stumps	Accuracy in percentage	Running time in seconds
10	54.613	4.080
50	63.945	34.412
100	64.475	56.228
200	66.914	103.557
250	67.126	122.718
300	66.489	129.716
400	68.399	165.643
500	68.717	271.693
750	68.611	416.065
1000	68.717	890.870

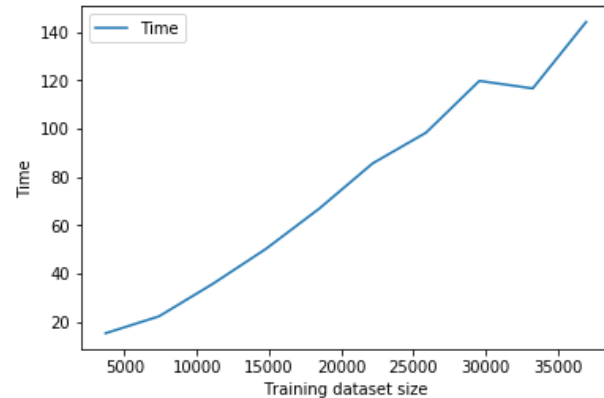
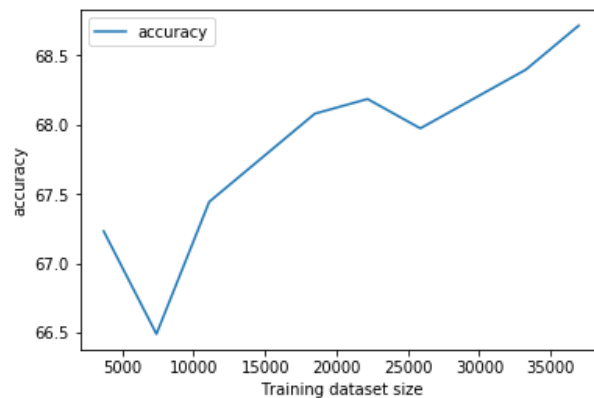


2. Performance depending upon the training dataset size when decision stumps=500

We have 36,976 entries or train vectors in our train-data.txt file which is being used to train the classifier. To determine classification accuracy as well as the running time with respect to the training data size we divide the training-data size into 10 intervals with each interval size greater than the previous interval by 3697 entries. We observe how the classification accuracy and running time is affected. We chose the best value of decision stumps=500.






The following observations are made:

Training data size	Accuracy in percentage	Time taken in minutes
3697	67.232	15.222
7394	66.489	22.222
11091	67.444	35.560
14788	67.762	50.093
18485	68.085	66.793
22128	68.187	85.515
25879	67.975	98.235
29576	68.187	119.693
33273	68.398	116.577
36970	68.717	144.115








Images Classified Correctly and incorrectly

Images Classified Correctly

test/10008707066.jpg			
test/10099910984.jpg			
test/10107730656.jpg			
test/10161556064.jpg			
test/10164298814.jpg			

Images Classified Incorrectly

test/10196604813.jpg			
test/ 10304005245.jpg			

test/10484444553.jpg			
test/10577249185.jpg			
test/ 10684428096.jpg			

Observation:

We can see a similar pattern in the images classified correctly, which is all the images have blue skies and a contrast color for the part of the image which is not sky. The random forest classifier is maybe able to classify the sky colors and able to assign it to the pixels at the top of image for correct classification.

The images which are not classified correctly are mainly greyscale image or images which do not have a diverse range of colors.

Random Forest

Reference for the program: <https://www.youtube.com/watch?v=LDRbO9a6XPU>

Random Forest Classification is a supervised learning algorithm. It is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Implementation-

Training-

1. After inputting the training data as a pandas dataframe, we convert it to a numpy array
2. For building a forest of decision trees, the inputs taken from the user are-
 - a. Max Depth: the depth to which the nodes are split
 - b. Sample size ratio: the ratio of the dataset which is used to build trees of the forest. This is selected randomly.
 - c. Number of trees: number of trees which are built for the forest
 - d. Number of features: the number of features from the full set of 192 features to consider building the tree. These are selected randomly
3. In the program, the features have been set as the best and optimum (running time is reasonable)
4. Before initializing a tree, subsample and features of the dataset are selected for which the tree must be built
5. Building a tree-
 - a. The subsample, subset of features is passed to the function for building a decision tree along with other parameters
 - b. Best split is found by checking the medians of all the columns considered (only the column median is considered to reduce the running time). Best split is calculated by calculating the **Information Gain** from the previous state to the state after the split. The information gain is the difference in the **Gini Impurity** from the initial state to the state after the split.
 - c. For all the branches split at the node a recursive tree is built until the maximum depth is reached or if the gain at the node after splitting is zero
 - d. The tree which is build is saved as reference to all the decision nodes and leaves
6. A forest is built by training number of decision trees. This is the number given by the user.

Testing-

1. For each row in the training dataset, the trees which are built in the training step and are traversed to give a decision at the leaf nodes. The class which is **maximum at the leaf node** is the predicted class for that tree. Similarly, all the trees are traversed for that row and the predicted class is obtained from all the trees. The final predicted class for the row is the class which is **voted majority** by all the trees.
2. This step is performed for all the rows to predict a class for all the rows.
3. Accuracy is calculated as number of predictions correct by total number of observations in the testing dataset

Accuracies and Testing time by parameters-

1. Max Depth

Accuracies and running time (train + test) is measured for different values of max depth keeping all the other parameters constant (number of trees = 10)

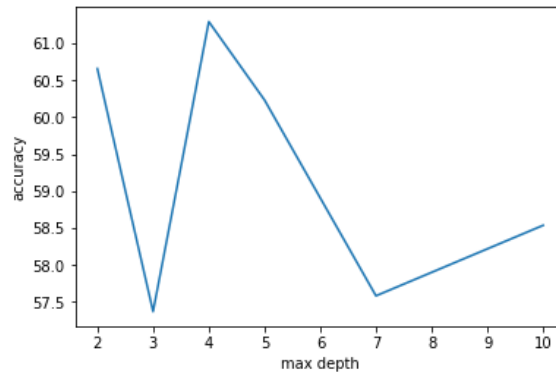


Fig. max depth vs accuracy

Table showing Max Depth vs Accuracy (in %) and Running Time(Train + Test)

Max Depth	Accuracy (in %)	Running Time (Train + Test)
2	60.65748	28.08492
3	57.3701	26.88442
4	61.29374	27.7947
5	60.2333	28.51769
7	57.58219	28.61156
10	58.53659	28.45792

Max accuracy is obtained for max depth = 4

2. Sample size ratio

Accuracies and running time (train + test) is measured for different values of sample size ration taken for building a tree in the forest keeping all the other parameters constant and max depth obtained for max accuracy.

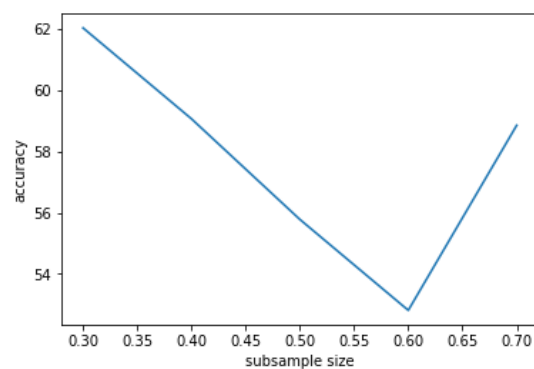


Table showing Sample size ratio vs Accuracy (in %) and Running Time (train + test)

Sample size ratio	Accuracy (in %)	Running Time (train + test)
0.3	62.03606	17.29324
0.4	59.06681	22.68735
0.5	55.77943	28.06913
0.6	52.81018	33.25314
0.7	58.85472	39.86064

Max Accuracy is obtained for Sample size ratio = 0.3

3. Number of trees

Accuracies and running time (train + test) is measured for different values of number of trees in the forest keeping all the other parameters constant and max depth and sample size ratio obtained for max accuracy. Please note that maximum accuracy will generally increase on increasing the number of trees but in the final submission, the number of trees is kept being 50 as accuracy increase very slowly after that point

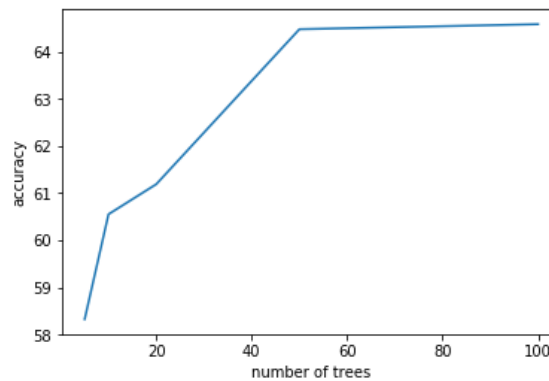


Fig. Number of trees vs accuracy

Table showing Number of trees in the forest vs Accuracy (in %) and Running Time (Train + Test)

Number of trees	Accuracy (in %)	Running time (train + test)
5	58.32	8.58
10	60.55	16.91
20	61.19	33.97
50	64.48	85.07
100	64.58	194.20

Maximum accuracy is obtained for Number of trees = 100

4. Number of features

Accuracies and running time (train + test) is measured for different values of number of features considered to build a tree keeping all the other parameters constant (number of trees = 50) and value of max depth and sample size ratio obtained for best accuracy

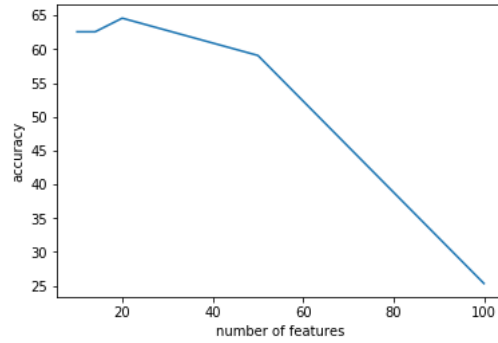


Fig. Number of features vs Accuracy

Table showing Number of features considered to build a tree vs Accuracy (in %) and Running Time (Train + Test)

Number of features	Accuracy (in %)	Running Time (Train + Test)
10	62.56628	27.05462
14	62.56628	34.80206
20	64.58112	47.64851
50	59.06681	109.7476
100	25.34465	199.6694

Maximum accuracy is obtained for Number of features = 20

5. Size of Dataset

Accuracies and Running time (train + test) is measured for different values of size of data set considered for training keeping all the features constant and equal to the value for which the maximum accuracy is obtained.

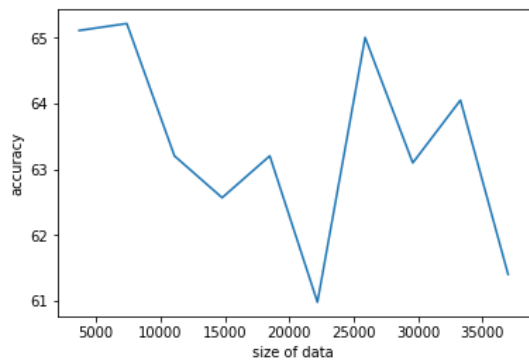


Fig. Size of data for training vs Accuracy

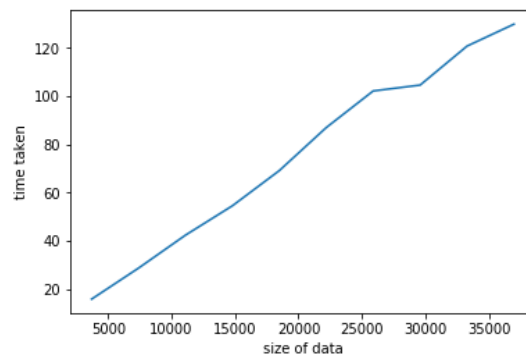


Fig. Size of data vs time taken

Table showing size of data considered for training vs Accuracy (in %) and Running Time (Train + Test)

Size of Data (number of rows)	Accuracy (in %)	Running Time (train + test)
3698	65.11	16.02
7396	65.22	28.10
11094	63.20	38.63
14792	62.57	56.98
18490	63.20	60.17
22188	60.98	71.56
25886	65.01	84.26
29584	63.10	94.98
33282	64.05	114.99
36980	61.40	124.80




Best Parameters-

These features are chosen for the final model, they are chosen as the right balance between running time and accuracy-

Max Depth	4
Sample size ratio	0.3
Number of trees	50
Number of features	20




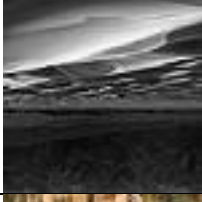

Images Classified Correctly and incorrectly

Images Classified Correctly

test/10008707066.jpg	
test/10099910984.jpg	
test/10107730656.jpg	

test/10161556064.jpg			
test/10164298814.jpg			

Images Classified Incorrectly

test/10196604813.jpg			
test/10351347465.jpg			
test/10484444553.jpg			
test/10577249185.jpg			
test/108172840.jpg			

Observation-

We can see a similar pattern in the images classified correctly, which is all the images have blue skies and a contrast color for the part of the image which is not sky. The random forest classifier is maybe able to classify the sky colors and able to assign it to the pixels at the top of image for correct classification.

The images which are not classified correctly are mainly greyscale image or images which do not have a diverse range of colors.

Recommended Model

Even though we are getting a higher accuracy with the knn-model which is ~71.2%, we would recommend Adaboost model to the clients with which we are getting an accuracy ~68.7% because of faster computation time.