Assignment 5 Report

KNN Classifier:

For KNN, few of the important parameters to be decided are-

- Value of k – how many nearest neighbors to consider for classifying a given image
- Number of pixels – whether to consider all pixels of the training data image or just a fraction of these (example, only the four edges of the training images), when comparing to a test data image
- Fraction of training data to consider

We experimented with different values for the above parameters and the results are as presented in the table below:

Row No.	k value	Pixels considered	Fraction of training data	Accuracy	Execution Time
1	1	Entire image	100%	67.23%	~ 45 minutes
2	200	Entire image	100%	71.16%	~ 50 minutes
3	192	Only considering the four edges	100%	72.43	~ 15 minutes
4	**200**	**Only considering the four edges**	**100%**	**72.53**	**~ 15 minutes**
5	369	Only considering the four edges	100%	72.43	~ 15 minutes
6	40	Only considering the four edges	400 randomly selected images (included all 4 orientations for these images, thus giving total 400*4 training images)	69.25%	~ 1 minute
7	60	Only considering the four edges	Same as above, 400 * 4 total training images	69.67%	~ 1 minute
8	100	Only considering the four edges	Same as above, 400 * 4 total training images	70.2%	~ 1 minute
9	150	Only considering the four edges	Same as above, 400 * 4 total training images	70.73%	~ 1 minute
10	40	Only considering the four edges	600 * 4 total training images	70.63%	~ 2 minutes
11	50	Only considering the four edges	600 * 4 total training images	71.9%	~ 2 minutes
12	100	Only considering the four edges	600 * 4 total training images	69.78%	~ 2 minutes

Case 1: Classifying the test images by using the entire training dataset and considering all the pixels takes longer, as shown in the table above.

Case 2: The execution time is reduced significantly if we compare the test images only with the edges of the training dataset images. Also, the accuracy is quite good with reduced comparisons. We then experimented by taking only a sample of training dataset.

Case 3: As shown in the table, the execution time was reduced to less than 2 minutes with lesser number of training images and the accuracy also was good enough. Since the training images were selected randomly, the accuracy results would differ with each run for the same k-value. However, on an average, the accuracy in this case was better than case 1 but less than case 2.

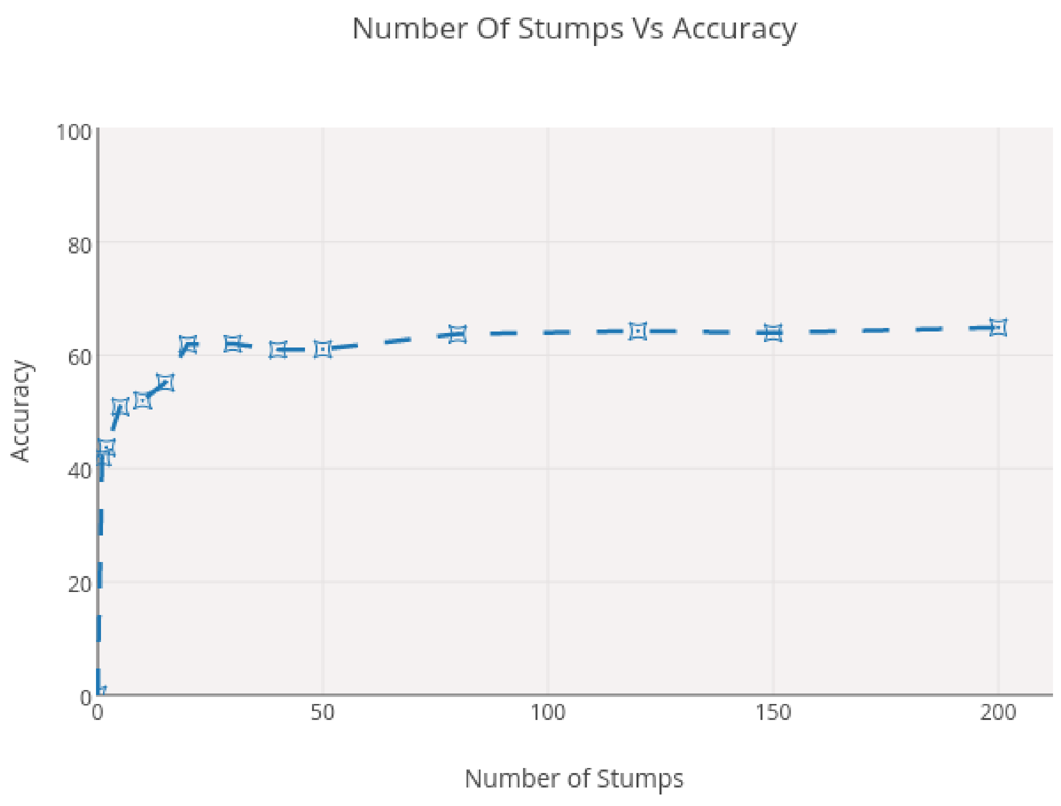
Thus, after experimenting with these parameters, we find that the best performance is given by comparing only the edges of all the images in the train dataset and finding the k = 200 nearest neighbors (highlighted in the table). Thus, if classification time is not a constraint, we can recommend KNN with these parameters to a potential client.

AdaBoost:

For our Adaboost the important parameters are:

- Combination of Two Pixel Indices – We compare the two pixel values at these indices and if they are greater then we go ahead and mark it as correctly classified, but otherwise we mark it as misclassified.

Here is a brief overview of the accuracy when tried over different values of stumps:

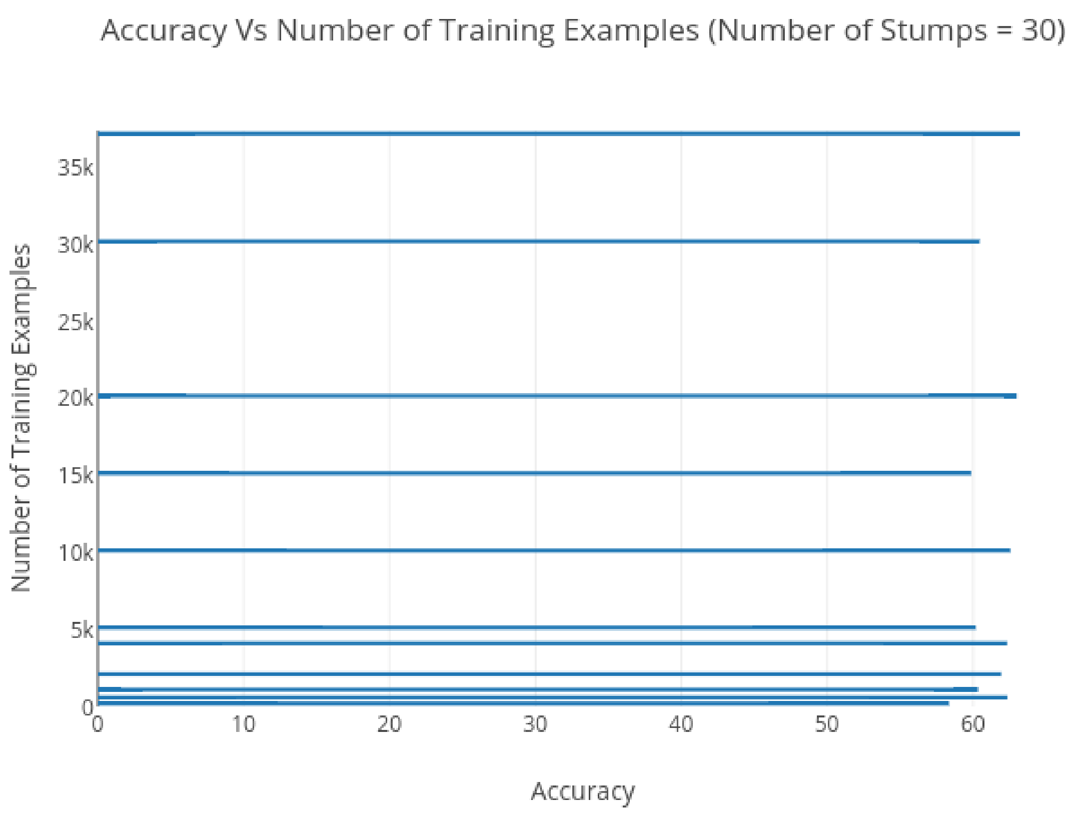


What does this infer?

The accuracy increases as we start to increase the number of stumps until a certain point and then remains fairly constant.

This is true if you think of it with basic intuition, when the number of classifiers are less the collective strength is also less and then when we increase the number of classifiers, we make a much stronger collection and then after a while it becomes constant. It becomes constant because our classifiers have only so much to learn, we would need to build a much more sophisticated classifier in order to improve our classifier beyond this point.

When we compare Accuracy with Number of Stumps we get this inference:



What do we infer?

We see the accuracy is fairly constant once we have about 500 Learning Examples or above. We tried, with 100 where the accuracy is about 58, but then once we hit 500, we see it is remains fairly constant. The variations we see are simply due to the randomness in our model, but otherwise the accuracy is pretty constant.

Which Parameters?

I would recommend them that they should have at least 50-100 stumps to get a consistent accuracy. The number 50 – 100 is given as a range, depending upon the size and time taken.

Time Taken (for 36976 records for training & testing): 5.22s

What are the observations?

There is a generic pattern that scenic images which have more contrast in their top portion vs their bottom portion are seen to be correctly classified.

Neural Network Classifier:

Neural Network we implemented had following open parameters that affected the overall model :

- Number of nodes in Hidden Layer (Hidden count)
- Alpha learning rate (Alpha)
- Number in iterations (I)

Hidden count	Learning Rate	Iterations	Time	Accuracy
3	10e-5	1000	2 min	25%
3	10e-7	10000	4 min	27%
3	10e-8	1000000	1 hour	42%
100	10e-7	1000000	2 hour	45%
150	10e-7	1000000	3 hour	69%
300	10e-7	1000000	5 hour	74%
350	10e-7	1000000	6 hour	74%
400	10e-7	1000000	8 hour	74%

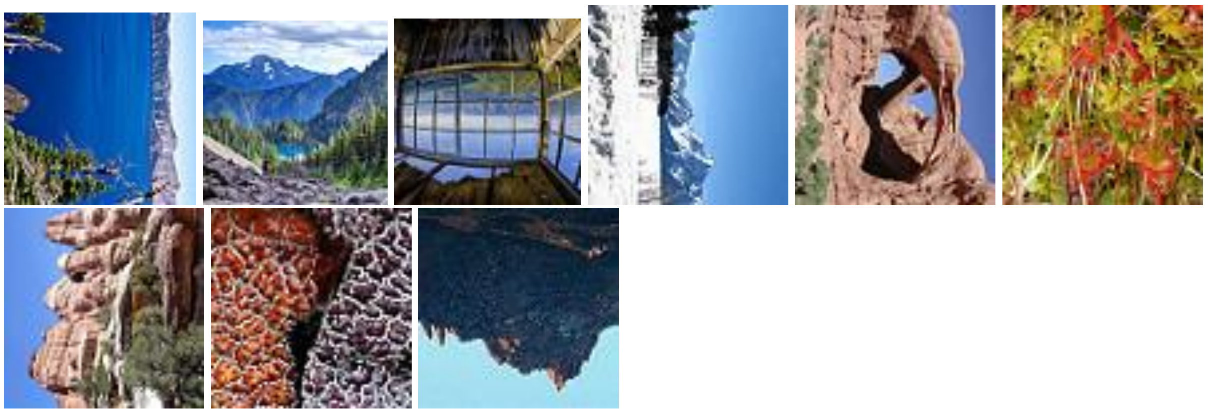
We could clearly see as we increase the complexity of the model with more iterations and hidden count (Nodes in hidden layer) the Accuracy of the classifier increases but after a certain threshold of hidden nodes (300) , Accuracy is constant around 74%.

Increasing the hidden count significantly increases the computation time.

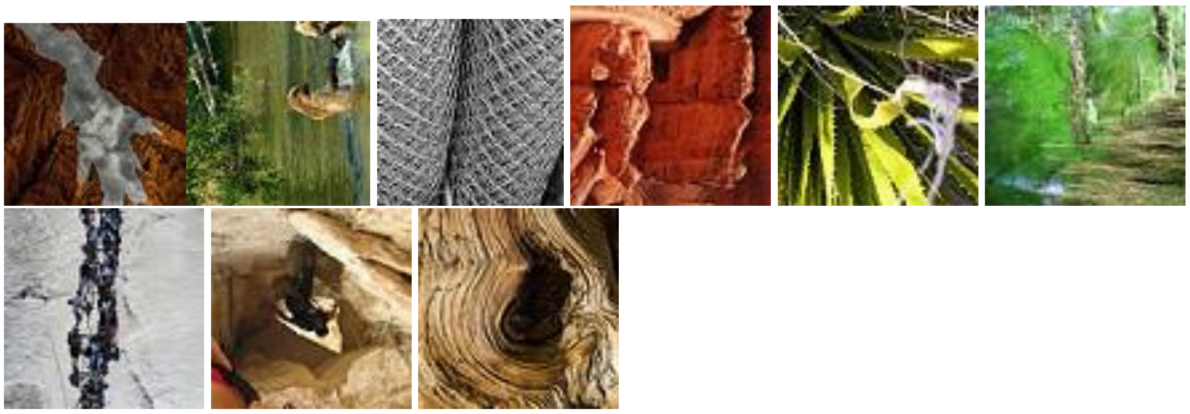
OUR INFERENCE AND RECOMMENDATIONS:

Sample Images

Correct:



Incorrect:



Observations:

It seems that the classifier is more likely to correctly identify images having more colors. As shown above, it was not able to classify the images which are mostly covered by a single color or shades of the same color or very few colors. Also, one interesting observation is that the classifier does pretty well for images where the sky can be clearly distinguished.

Recommendation to Client:

We would recommend **Neural Nets** to the client.

We’ll particularly recommend Neural Network model to the client as it can be trained on large historical data and the model can be saved for future classifications which is sufficiently fast.

We recommend the Neural Network with highest accuracy from the above table. This model is already saved in the file 'model-nn.p' and can be readily used for new images.