1. NEAREST NEIGHBOR

Methodology:

Approach:

During the training phase of the algorithm we only stored the feature vectors and class labels of the training samples. During testing, for getting the predicted class, we iterate over all the training data points and calculate the distance between test data and each row of training data. We find the k nearest neighbor and assign the label which gets matched with majority of the nearest label. We took Euclidean distance as our distance function.

Parameters:

To decide how many neighbors needs to be consider in determining the classifier, parameter K is used. Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee.

Results:

The performance ranges around 70 %. The best performance is at k= 45 which gives 71.47 % efficiency. As the value of k decreased, the performance is also decreased. K=1 , gives the least efficiency of 67.23%.

Table 1.1 k nearest neighbor

k	*Performance (%)*	*Time (s)*
1	67.232	397.91
5	69.141	391.75
10	70.307	401.46
20	70.519	857.13
25	71.049	741.84
30	70.201	368.85
35	70.413	407.72
40	71.049	408.81
45	71.474	403.99
50	71.261	399.95

2. ADABOOST:

Methodology:

Approach:

In some scenarios, we may not be able to build a single tree which would explain all the data without facing the problem of overfitting. Hence, it is ideal to come up with multiple hypotheses/ classifiers/ decision trees to represent an ensemble learning approach. Incase of Adaboost, we build multiple decision stumps (decision trees with level of just 1) to train our data and classify the label. Each time a new hypothesis is considered, we would like it to perform well on data that has been mis-classified by the previous hypotheses. In the end we make decision by taking weighted vote of all classifiers that we have come up with.

In our implementation of Adaboost for the image classification problem, we have considered classifiers as one Vs all. In other words we compare each orientation labels with all the other orientation variables present and marking that orientation. (E.g : {90} Vs {0,180,270}. Value 1 stored for {90} and 0 for {0,180,270}) For the hypothesis we are randomly choosing 2 features (out of 192) and comparing them (Feature1>=Feature2) to make a decision. The video lecture also discusses about training each decision tree for different datasets/splits. We have included a function to generate data subsets which gives 80% train data and further design aspects include how to choose this parameter. The parameter 80% is adjustable and can be increased or decreased depending on data size. If the data file is too large we can consider a relatively less portion of data. As we increase number of hypotheses, there would not be much difference in accuracy (as observed for K=100 for 80% and 100%). If the data file is relatively small so that we (client) need not worry about time constraint, then it could be 100% (all the training data) for maximum accuracy. While training, the attributes needed for testing are saved in model_file.txt to predict accuracy.

Parameters:

Here, there is only one parameter that is considered as a design decision. That is the number of decision stumps and we did few test runs to find out the best one.

Results:

Below are the results observed for training sample of 75% size :

Table 1.1 k nearest neighbor

Decision Stumps	*Runtime(secs)*	*Accuracy*
30	3	62.77
40	4	61.50
50	4	63.30
100	7	65.32
200	12	66.80
400	22	66.80
60	33	66.80

In the decision stumps range of 30-50 , we observed the accuracy is less for decision stumps of 40 (Accuracy 61.50) when compared to 30 decision stumps (Accuracy 62.77). The accuracy decreases by a margin if we compare 200 with 400 and 600 stumps.

Hence, for the best model we have chosen Adaboost.

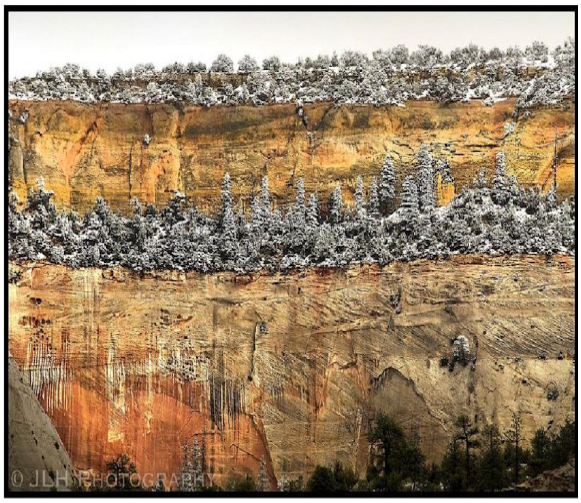
Testing Examples:

Incorrectly classified :

test/10196604813.jpg 90 - predicted as 0

Correctly classified:

test/10008707066.jpg 0



We think the angle from which the photo is taken plays a main role as decision factor for this problem. Also the training data must be lacking variety of pictures captured from different angles/viewpoints in our opinion.

3. RANDOM FOREST:

Methodology:

Approach:

Random forest is building trees based on multiple samples of the training data. Random Forest builds multiple decision trees and merges the results together to get a more accurate and stable prediction.

The Labels taken here are: 0, 90, 180, 270.

For every value of every feature, we calculate the gini index. Gini Index is one kind of splitting criterion. It tells us how good of a split it is. If Gini Index is close to 0 or 0, then it is considered as a good split. Else, it continues to find a good split.

We choose the feature and feature value of the split given by the gini index. We continue splitting until we reach the stopping criterion. The topping criterion are:

1. Maximum depth of a tree.
2. Minimum number of rows for node.

In this way, we build all the decision trees. Once we get the trees, we pass each test exemplar into the trees and predict our labels. We choose the maximum voted label of all the trees and assign it to that test exemplar.

Parameters:

The parameters used here are:

1. Number of decision trees to be built - 100
2. Maximum depth for each tree - 10
3. Minimum number of nodes for each node - 100