Homework4

Q1:

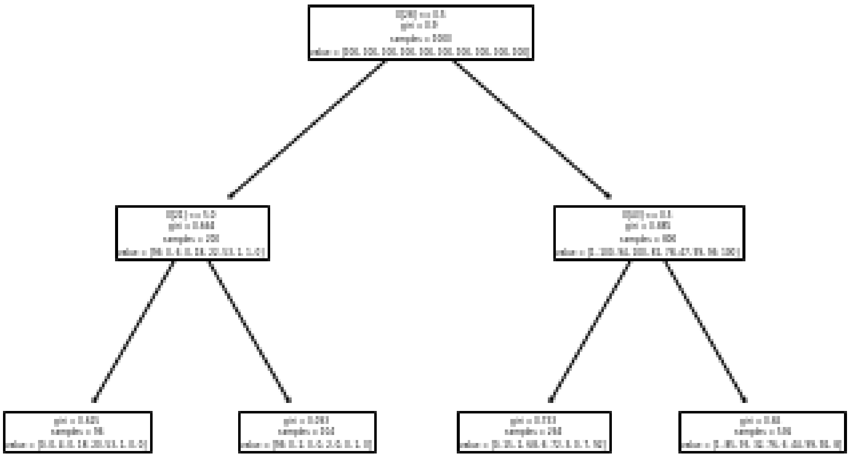
tree for depth = 2

'train_err_25_2': 0.712,

'test_err_25_2': 0.7545909849749582,

'train_err_100_2': 0.657,

'test_err_100_2': 0.6878130217028381



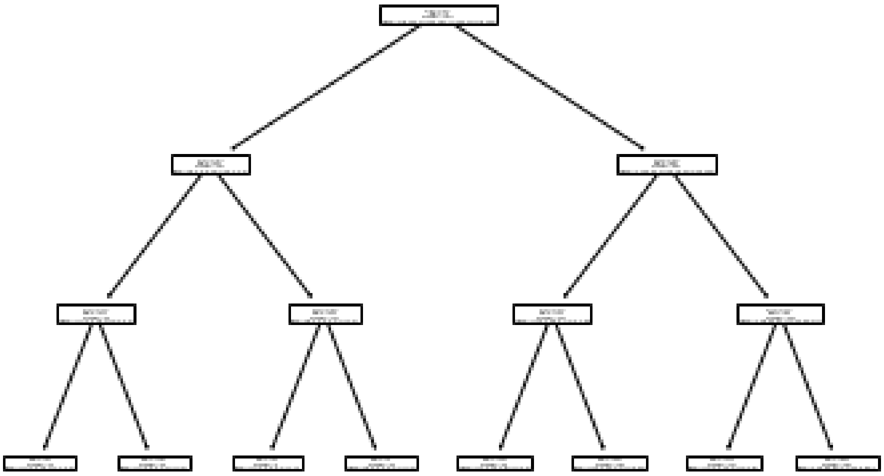
tree for depth = 3

'train_err_25_3': 0.536,

'test_err_25_3': 0.6104618809126321,

'train_err_100_3': 0.485,

'test_err_100_3': 0.5375626043405677



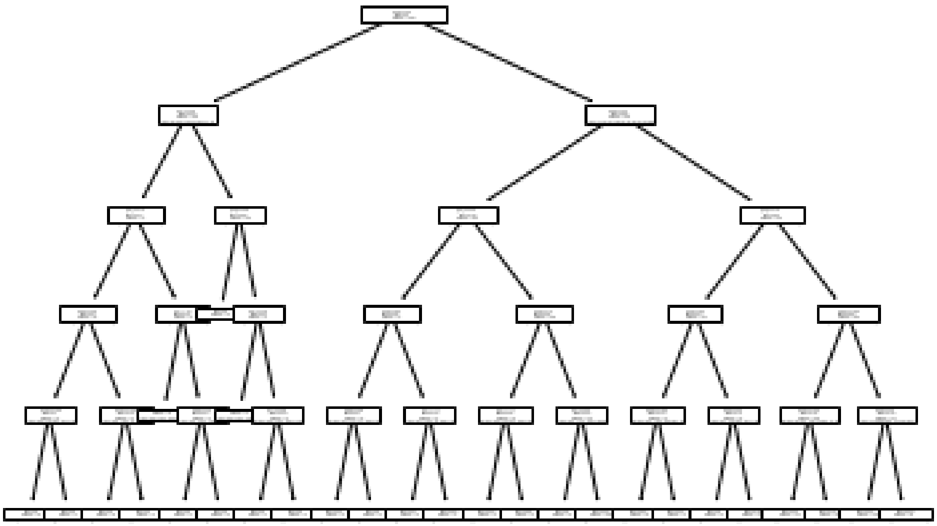
tree for depth = 5

'train_err_25_5': 0.19999999999999996,

'test_err_25_5': 0.38842515303283254,

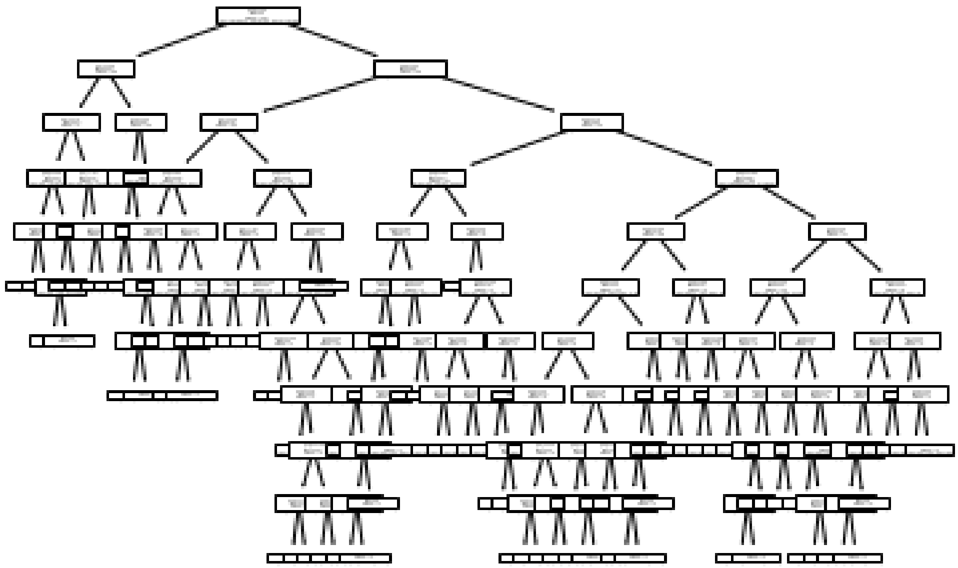
'train_err_100_5': 0.19699999999999995,

'test_err_100_5': 0.320534223706177



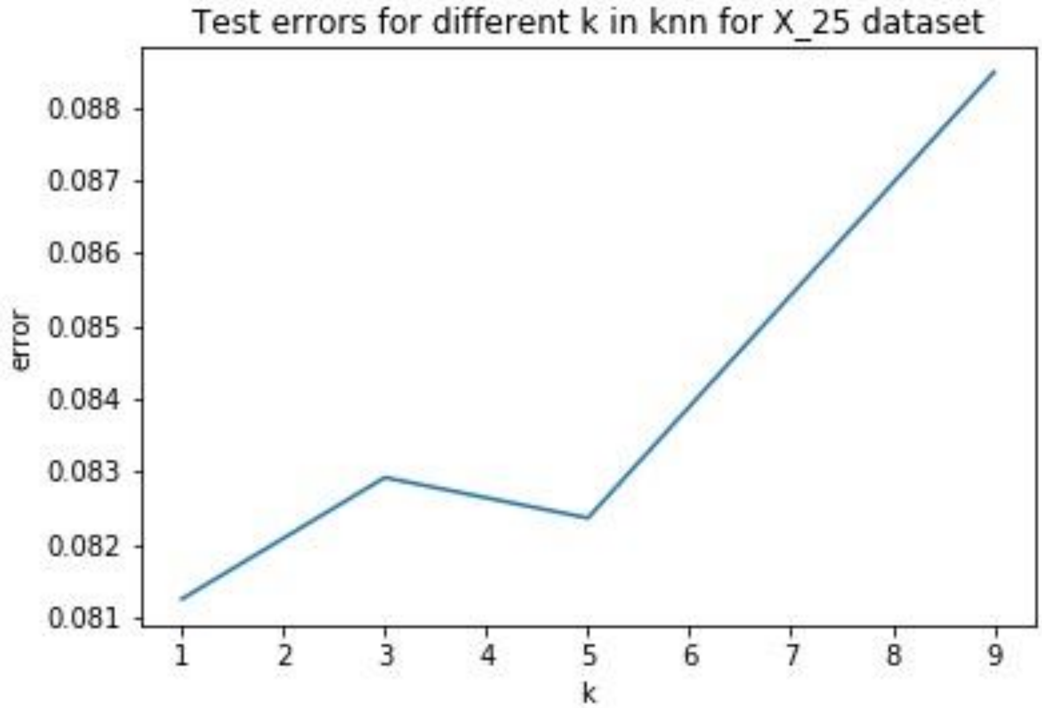
tree for depth = 10

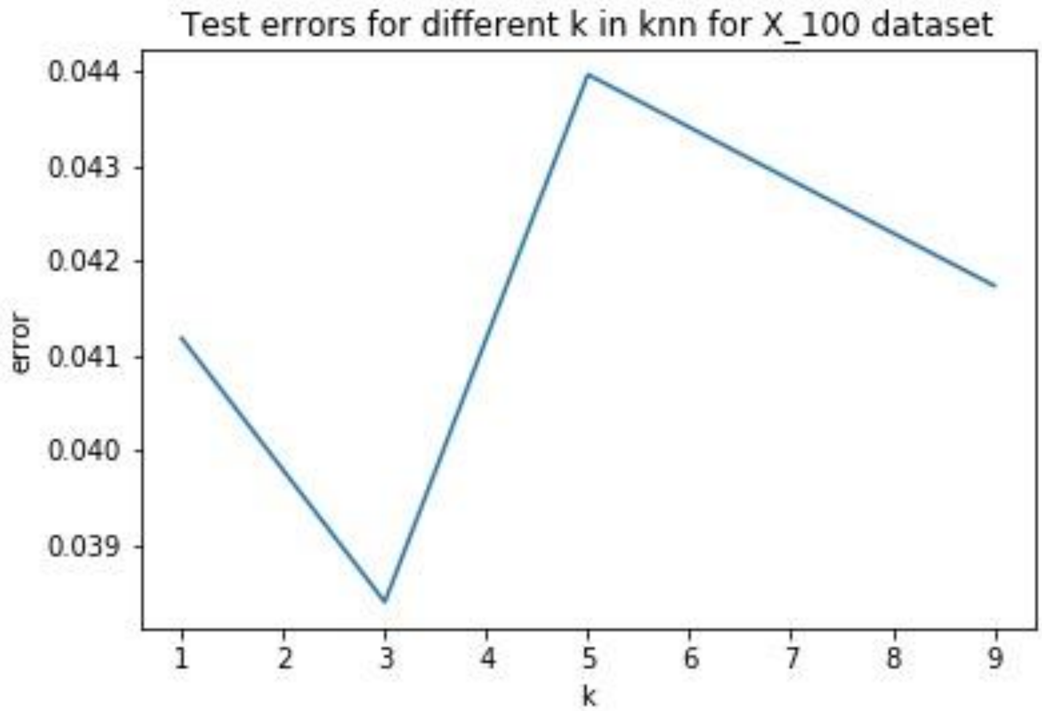
```
'train_err_25_10': 0.0,

'test_err_25_10': 0.29771841958820255,

'train_err_100_10': 0.007000000000000006,

'test_err_100_10': 0.2309404563160824
```



Q3: For X_25 data set k-nearest neighbor algorithm with neighbors K = 1 performs better. However, as the number of instances increases in X_100 dataset K = 3 performs better. In both the cases k =9 does not perform well and error shoots up. Also, K =1 performs

Please see the graph for test error with the models trained on X_25 and X_100 datasets respectively for k = [1,3,5,9]


Test errors for different k in knn for X_25 dataset
error
0.081
0.082
0.083
0.084
0.085
0.086
0.087
0.088
1
2
3
4
5
6
7
8
9
k


Test errors for different k in knn for X_100 dataset
error
0.039
0.040
0.041
0.042
0.043
0.044
1
2
3
4
5
6
7
8
9
k

Q4: Predicting pixels whose value is all zero gives 0 error and hence is the easiest to predict. Also, as we can see in the graph below, finding pixel 57 and 54 are with the minimal error which does not have all zero values.