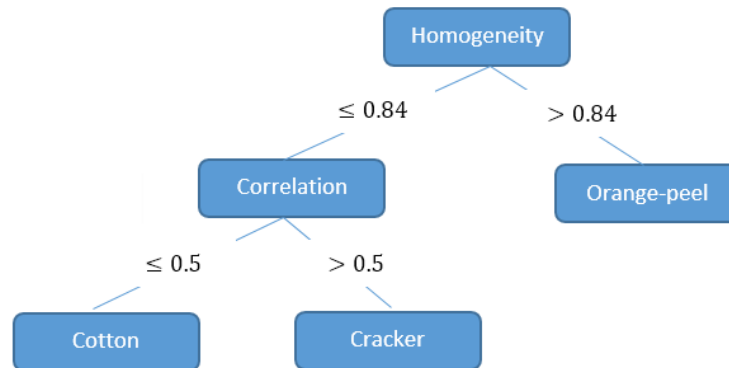


Part 1: Simple, hand-made classifier

We select three different textures classes from KTH-TIPS database. In order to extract the features from each of the images in each class, the gray-level co-occurrence matrix (GLCM) of the image is computed, Then the statistical properties(Contrast, Correlation , Energy, Homogeneity) of the GLCM of each image is calculated. These properties are the image features. Each class has 81 images taken at different conditions. The average result of features for each class is calculated and represented in the below table.




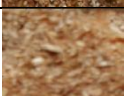

Feature Class	Cotton	Cracker	Orange-peel
Contrast	0.9318	1.0789	0.2055
Correlation	0.2315	0.7051	0.7333
Energy	0.2133	0.0815	0.3840
Homogeneity	0.7085	0.7186	0.9043

We build a handmade decision tree, which can classify a random image from any of the three classes. According to the data, it seems that Homogeneity can be a good choice for first splitting. Correlation can be a good choice for the second split.



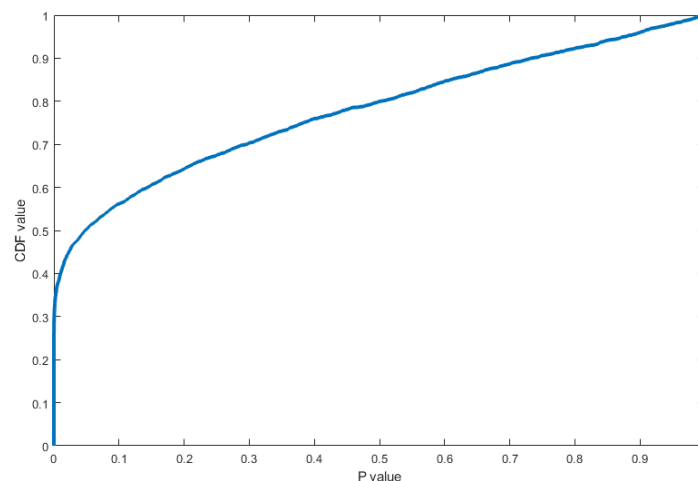
The follow table shows images that were selected randomly from the three classes and were classified correctly or incorrectly by the above decision tree.

Randomly selected image	Correct Class	Predicted Class	Was it classified correctly?
-------------------------	---------------	-----------------	------------------------------

			Cotton	Cotton	Yes
			Orange-peel	Orange-peel	yes
			Cracker	Cracker	yes
			Cracker	Orange-peel	no
			Cotton	Orange-peel	no

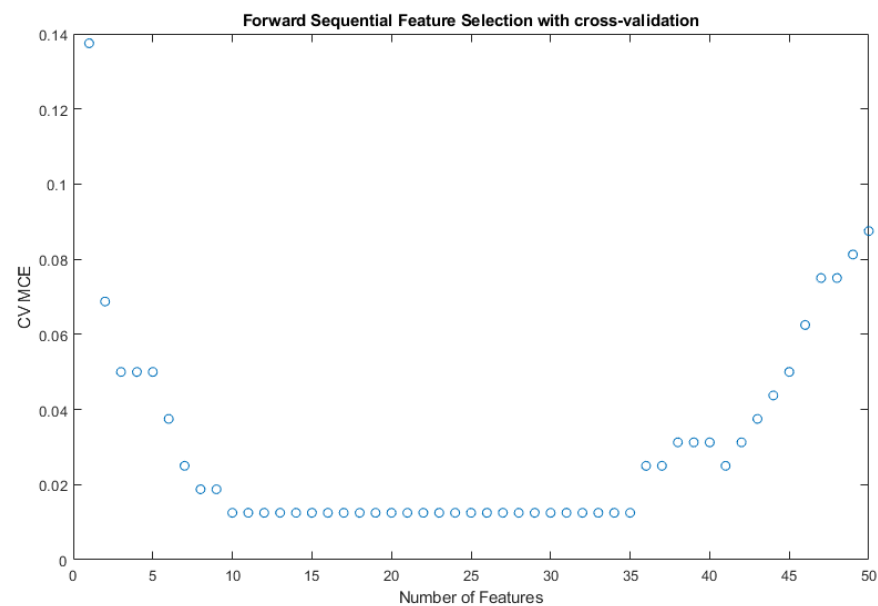
Part 2: Machine Learning workflow using MATLAB

In this part, we are working with a set of bioinformatics data that can be utilized to classify between normal and cancer. The file ovariancancer contains two arrays of grp and obs. There are 216 observations; the labels are stores in the file grp as cancer or normal, and their corresponding features are stored in the obs. Each observation has 4000 features. Using cvpartition command in matlab, we partition the dataset into 160 points for the training set and the remaining 56 observations for testing set. Since there are many features available, we use features selection methods to reduce the number of features. At First, the important features are selected using a filter-based method, which is based on the t-test. The following figure shows the empirical cumulative distribution function (CDF) of the p-values. As we can see, more than 50% of features have p-values close to zero.



We sort the features and select the first 150 features with the smallest p values. The feature indices are stored in fs1. Then we make a 10 fold cross validation of the training data, and define a function handler

as the misclassification error, which uses quadratic discriminant analysis (QDA) for classification. This function along with the cross validation training data are used in the function sequentialfs in order to do forward sequential feature selection on the 150 features. This feature selection method selects 3 features and the performance of the selected model with these three features is obtained from MCE (misclassification error) on the 56 test samples. The computed MCE is 7.14%. Since the sequential algorithm may have stopped early, we run the forward sequential feature selection algorithm for a range of features and plot the cross-validation MCE as a function of the number of features for up to 50 features.



From the above plot we see that the cross validation MCE reaches minimum for 10 features. We compute MCE using QDA and the 10 selected features for the test set. The resulting error is 3.57%, which shows improvement compared to only 3 features being selected.

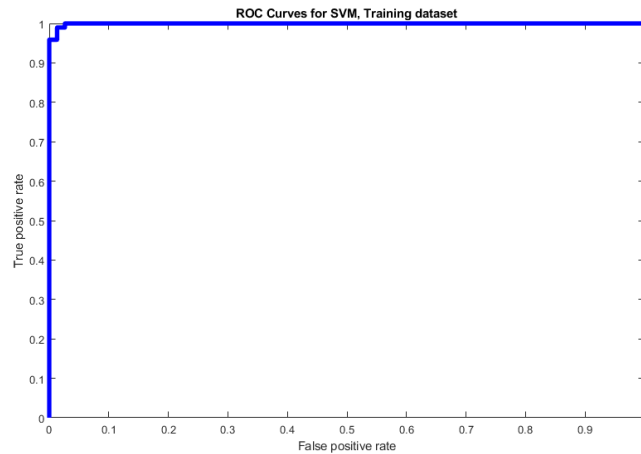
After feature selection, we build a SVM model classifier for the data with 10 selected features. We compute 10 fold cross-validation misclassification rate (loss), and accuracy for the built model. The misclassification rate is 9.3% and the accuracy is 91%. The confusion matrix for the 10 fold cross validation is shown below.

		Predicted	
		Cancer	Normal
True	Cancer	88	9
	Normal	7	69

PositivePredictiveValue = 0.9263

NegativePredictiveValue = 0.8846

The ROC plot is obtained for this SVM model using resubstitution in the training set and is shown below. The area under the curve (AUC) is 0.9993.



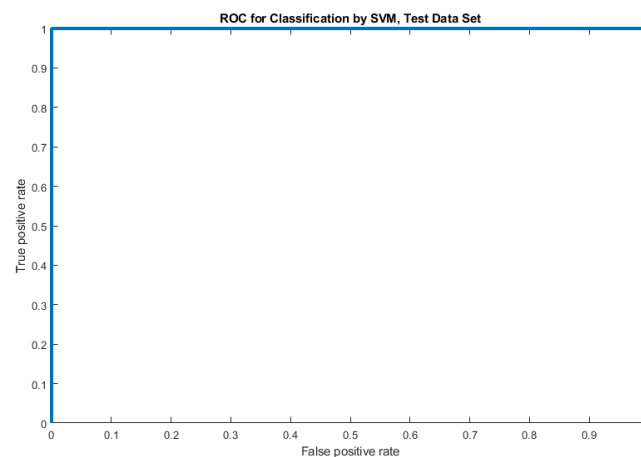
The accuracy of the model (SVM fitted for 10 features on the training set) on the test set was computed as 98%. The confusion matrix is

		Predicted	
		Cancer	Normal
True	Cancer	24	0
	Normal	1	18

PositivePredictiveValue = 0.9600

NegativePredictiveValue = 1

The ROC plot on test set is shown below which has an AUC of 1.

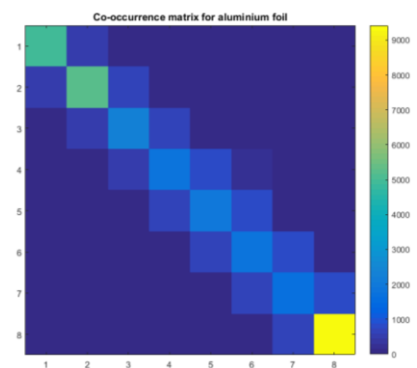


Part 3: Image-based classifier

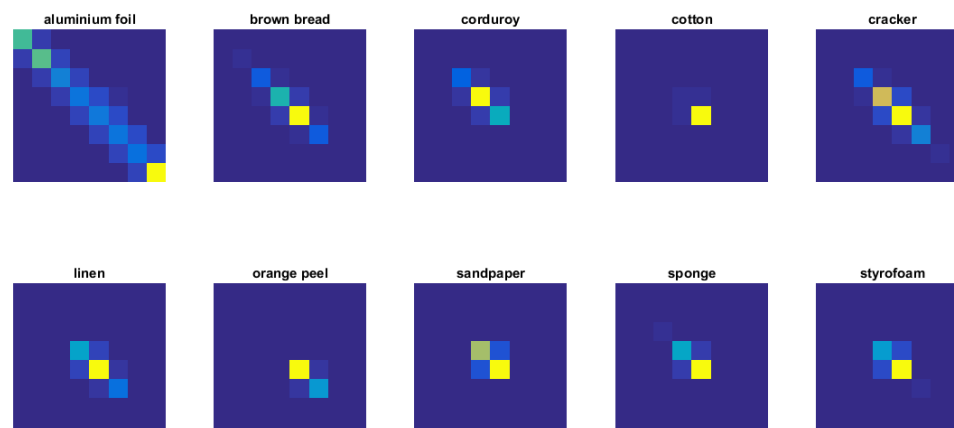
Here we build a 10-class texture classifier with the features that are elements of the gray-level co-occurrence matrix (glcm) of the images. A sample picture from each class is shown in figure below.



Each glcm has eight intensity levels with 64 elements, which are chosen as the features of the images. Figure below shows the heat map of the glcm for one image from aluminum-foil class.

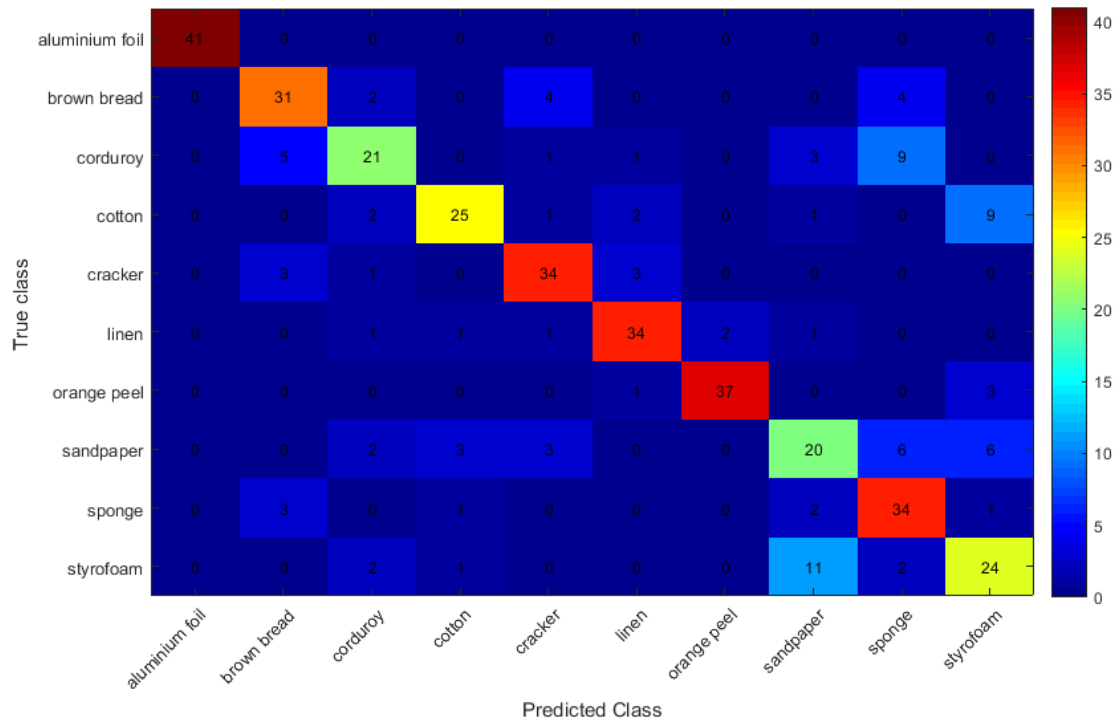


The following figure shows a sample from each class. It shows how various types of textures have different glcm.



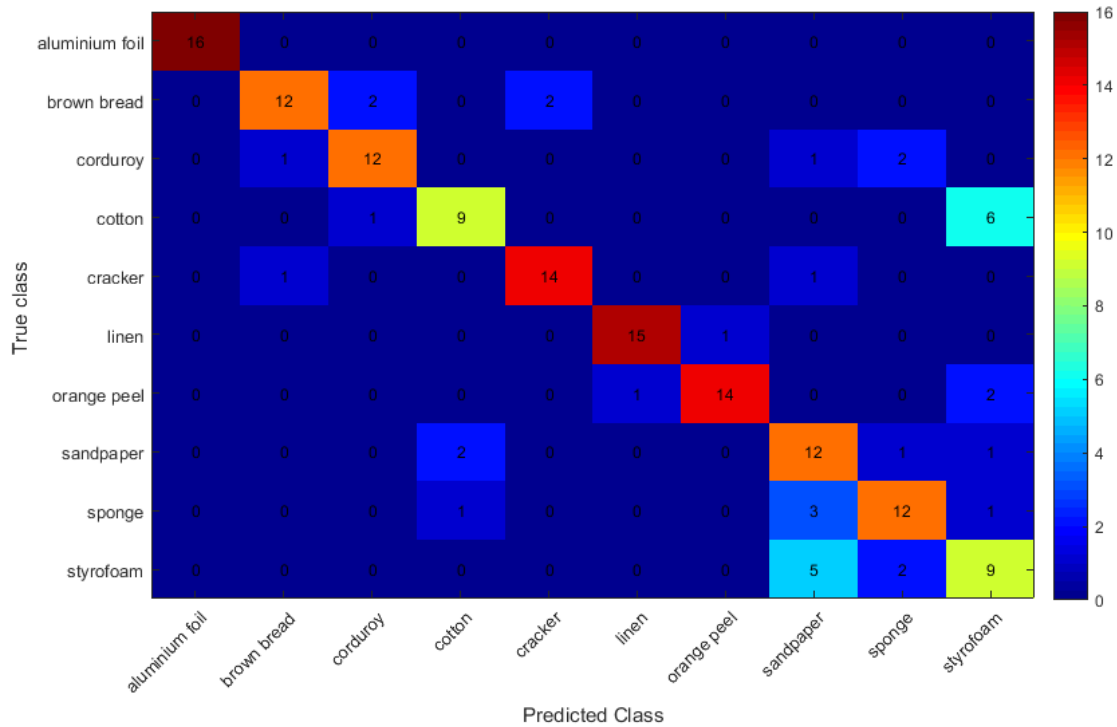
In this project, the dataset containing 81 different images from each class is divided into two equal sets of training and test. Then, k nearest neighbor algorithm (with $k=1$) is used as the classification algorithm.

The result of the classification is used as the baseline for comparison of other algorithms. The baseline accuracy was 74.321% and the following figure shows the confusion matrix for the baseline classification.



Using the starter code as an example, design a **10-class** classifier based on **different texture** features. Keep the kNN portion unchanged (i.e. use $k = 1$ and a 80/20 partition of the dataset). Record the accuracy (**modified texture-based accuracy**). Is it better than the **baseline accuracy**? Why (not)?

kNN algorithm was used again with different fractions of data for training and testing. This time, 80% of data was used for training and 20% for testing. No other parameters were changed in compare to baseline classification. The classification accuracy was 77.1605% which is better in comparison to baseline accuracy because we used a bigger training dataset, which produces a better classification model. The confusion matrix is shown in the following figure.

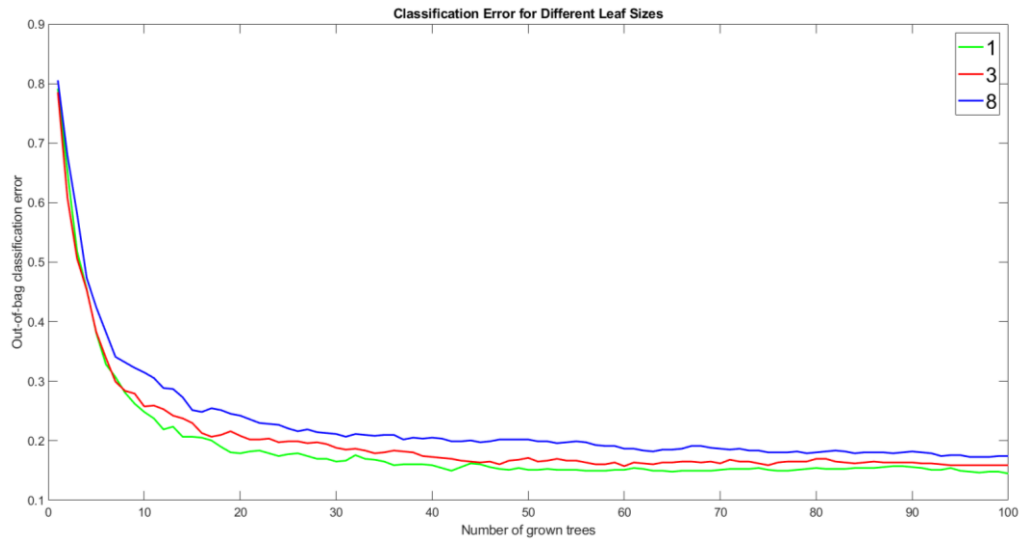


Investigate the **confusion matrices** for the two cases. Which classes are more likely to be confused? Do the results match your intuition?

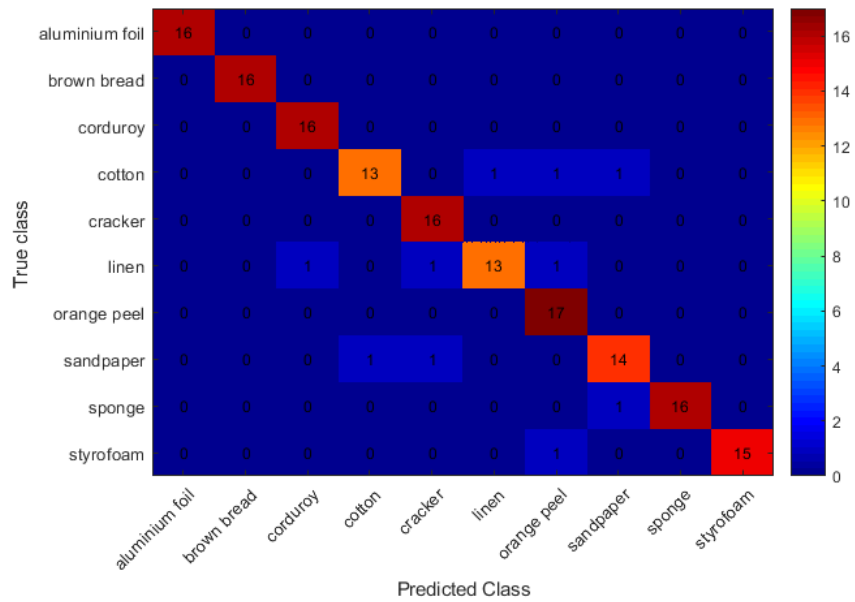
From the confusion matrices of the kNN classifications, we observe that it is very probable that Styrofoam be misclassified as sandpaper, and cotton be misclassified as Styrofoam. It can be seen that aluminum foil has the best prediction and was predicted always correctly. From figure of glcm we can see that aluminum has a distinguishing glcm compare to other textures glcm which makes it better to be classified. Also, we can see that styrofoam and sandpaper have similar glcm features, that's why they become confused with each other.

Repeat step 1, but this time using **a different classifier** and **making any other meaningful changes you might deem reasonable** (e.g., validation strategy, parameter optimization, etc.).

The data were divided into 80% for training and 20% for testing. Treebagger algorithm was used to create a bag of decision trees for classification using training set. The number of trees was set to 100 and the algorithm was utilized 3 times with different Minimum number of observations per tree leaf (MinLeaf). MinLeaf was set to 1, 3, and 8. The plot of out-of-bag (OOB) error versus the number of trees for each case is shown below. We can see that MinLeaf equal to 1 has slightly better results.



The Treebagger algorithm was performed again by setting MinLeaf to 1. The accuracy of the model on the test data set was 93.83%. The confusion matrix on the test data set is shown below.



The following table compares the performance of the above classification models on the test set.

Algorithm	Accuracy
kNN 50/50--baseline	74.321%
kNN 80/20	77.1605%
Treebagger80/20	93.83%