Assignment 2 (ML)

Akhil Babu Manam (927000968)

October 12, 2018

1 Conceptual

The answers for each question in this section have been written below each question. In addition, using the data given as part of question 3 of this assignment, the three models mentioned in parts 1 and 2 of this question are answered. The codes for these are present in q1.ipynb in the zip file.

1.1 Which of the three models with k predictors has the smallest training RSS?

It can generally be expected that the best subset selection will have the best training RSS as it covers all the subsets and is the optimal selection strategy. Also, since training RSS is considered, the model with more predictors generally works wells. The results obtained are in coherence with the results that can be observed from table 1.2 which contains the training error (RSS).

k	Best Subset	Stepwise Forward	Stepwise Backward	Predictors
1	118.73	118.73	118.73	5
2	77.20	77.20	77.20	1,5
3	69.35	69.35	69.35	5,1,0
4	64.87	64.87	64.87	5,1,0,7
5	**63.46**	63.57	63.57	0,1,3,5,7
6	**62.34**	62.78	62.78	0,1,2,3,5,7
7	**61.77**	62.68	62.68	0,1,2,3,5,6,7
8	61.70	61.70	61.70	0,1,2,3,5,6,7,8

In the table above, the model predictor encoding are as follows, 0 - Clump thickness 1 - Uniformity of cell size 2 - Uniformity of cell shape 3 - Marginal adhesion 4 - Single epithelial cell size 5 - Bare Nuclei 6 - Bland chromatin 7 - Normal Nucleoli 8 - Mitoses

It can be seen that the predictors were same till k = 4, but from k = 5 to k = 7, the predictors for the best subset approach were optimal and stepwise forward and backward approaches were only sub-optimal.

1.2 Which of the three models with k predictors has the smallest test RSS?

It can generally be expected that the best subset selection will have the best testing RSS as it covers all the subsets and is the optimal selection strategy. But, since testing RSS is considered, the model with more predictors may not be the optimal selection. The results obtained are in coherence with the results that can be observed from table 1.2 which contains the testing error (RSS).

k	Best Subset	Stepwise Forward	Stepwise Backward	Predictors
1	**50.00**	52.45	52.45	6
2	28.64	28.64	28.64	1,5
3	23.54	23.54	23.54	5,1,0
4	**20.88**	21.33	21.33	0,4,5,6
5	**19.38**	21.83	21.83	0,4,5,6,7
6	**19.39**	20.40	20.40	0,1,4,5,6,7
7	**19.72**	20.06	20.06	0,1,4,5,6,7,8
8	**20.23**	20.62	20.62	0,1,2,4,5,6,7,8

In the table above, the model predictor encoding are as follows, 0 - Clump thickness 1 - Uniformity of cell size 2 - Uniformity of cell shape 3 - Marginal adhesion 4 - Single epithelial cell size 5 - Bare Nuclei 6 - Bland chromatin 7 - Normal Nucleoli 8 - Mitoses

It can be seen that the best minimal test error was mostly obtained by best subset selection and it was similar with stepwise forward and backward approaches only when k = 2 and 3. In all other cases, test error was minimized for predictors that were not a part of model in stepwise strategies.

1.3 True or False

1.3.1 The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the k + 1-variable model identified by forward stepwise.

Answer : True

1.3.2 The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the k + 1-variable model identified by backward stepwise.

Answer : True

1.3.3 The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the k + 1-variable model by backward stepwise.

Answer : False
Explanation : Not necessarily true always.

1.3.4 The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the k + 1-variable model by forward stepwise.

Answer : False
Explanation : Not necessarily true always.

1.3.5 The predictors in the k-variable model identified by best subset are a subset of the predictors in the k + 1-variable model by best subset.

Answer : False

2 Ridge and Lasso

2.1 Write out the ridge regression optimization problem in this setting.

In the linear regression setting, we try to find the best fit for outcome y using,

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_p \tag{1}$$

where $x_1, x_2, ...x_p$ are the predictors or features.

Now, we generally try to reduce residual sum of squares (RSS) which can be written as,

$$RSS = \Sigma_{i=1}^{n}(y_i - \beta_0 - \Sigma_{j=1}^{p}\beta_j x_{ij})^2 \tag{2}$$

where n is the number of samples and p is the number of features (dimensions).

In this problem, it has been given that $n = 2$ and $p = 2$. Therefore, RSS can be written as

$$RSS = (y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2 \tag{3}$$

Substituting $x_{11} = x_{12}$, $x_{22} = x_{21}$ and $\hat{\beta}_0 = 0$ (ridge or lasso prediction of β_0) from the given,

$$RSS = (y_1 - (\beta_1 + \beta_2)x_{12})^2 + (y_2 - (\beta_1 + \beta_2)x_{21})^2 \tag{4}$$

Expanding eq.4,

$$RSS = y_1^2 + (\beta_1 + \beta_2)^2 x_{12}^2 - 2(\beta_1 + \beta_2)y_1 x_{12} + y_2^2 + (\beta_1 + \beta_2)^2 x_{21}^2 - 2(\beta_1 + \beta_2)y_2 x_{21} \tag{5}$$

From the question, it can be seen that $x_{21} = -x_{11} = -x_{12}$, substituting this in eq.5,

$$RSS = y_1^2 + y_2^2 + 2(\beta_1 + \beta_2)^2 x_{12}^2 - 2(\beta_1 + \beta_2)x_{12}(y_1 - y_2) \tag{6}$$

Expanding the RSS factor and adding the regularization term,

$$RidgeError = y_1^2+y_2^2+2\beta_1^2x_{12}^2+2\beta_2^2x_{12}^2+4\beta_1\beta_2x_{12}^2-2\beta_1x_{12}(y_1-y_2)+2\beta_2x_{12}(y_1-y_2)+\lambda(\beta_1^2+\beta_2^2) \tag{7}$$

2.2 Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$

From eq.7, the optimal values of β_1 and β_2 can be obtained by computing partial differentials and equating them to 0.

$$\frac{\partial RidgeError}{\partial \beta_2} = 4\beta_2 x_{12}^2 + 4\beta_1 x_{12}^2 - 2x_{12}(y_1 - y_2) + 2\lambda\beta_2 = 0 \tag{8}$$

$$\frac{\partial RidgeError}{\partial \beta_1} = 4\beta_1 x_{12}^2 + 4\beta_2 x_{12}^2 - 2x_{12}(y_1 - y_2) + 2\lambda\beta_1 = 0 \tag{9}$$

By substituting $y_1 = -y_2$ from the given in eq.8 and eq.9,

$$\beta_1(4x_{12}^2 + 2\lambda) + 4\beta_2 x_{12}^2 = -4x_{12}(y_1) \tag{10}$$

$$\beta_2(4x_{12}^2 + 2\lambda) + 4\beta_1 x_{12}^2 = -4x_{12}(y_1) \tag{11}$$

Eq.10 can be written as

$$4\beta_1 x_{12}^2 + 4\beta_2 x_{12}^2 = -4x_{12}y_1 - 2\lambda\beta_1 \tag{12}$$

Similarly, eq.11 can be written as

$$4\beta_1 x_{12}^2 + 4\beta_2 x_{12}^2 = -4x_{12}y_1 - 2\lambda\beta_2 \tag{13}$$

Since LHS of both the above equations are same, equating the RHS, $\beta_1 = \beta_2$.

2.3 Write out the lasso optimization problem in this setting.

The lasso optimization error would be

$$LassoError = RSS + \lambda(|\beta_0| + |\beta_1|) \tag{14}$$

By substituting eq.6 in eq.14,

$$LassoError = y_1^2+y_2^2+2\beta_1^2x_{12}^2+2\beta_2^2x_{12}^2+4\beta_1\beta_2x_{12}^2-2\beta_1x_{12}(y_1-y_2)+2\beta_2x_{12}(y_1-y_2)+\lambda|\beta_1|+\lambda|\beta_2| \tag{15}$$

Substituting $y_1 = -y_2$,

$$LassoError = 2y_1^2 + 2\beta_1^2x_{12}^2 + 2\beta_2^2x_{12}^2 + 4\beta_1\beta_2x_{12}^2 - 4\beta_1x_{12}y_1 + 4\beta_2x_{12}y_1 + \lambda|\beta_1| + \lambda|\beta_2| \tag{16}$$

The above equation is the Lasso optimization equation for this problem.

2.4 Argue that in this setting, the lasso coefficients β_1 and β_2 are not unique in other words, there are many possible solutions to the optimization problem. Describe these solutions.

From eq.16, to obtain the optimal values for β_1 and β_2 for minimizing the Lasso error,

$$\frac{\partial LassoError}{\partial \beta_1} = 4\beta_1 x_{12}^2 + 4\beta_2 x_{12}^2 - 4x_{12}y_1 + \lambda\frac{|\beta_1|}{\beta_1} = 0 \tag{17}$$

$$\frac{\partial LassoError}{\partial \beta_2} = 4\beta_1 x_{12}^2 + 4\beta_2 x_{12}^2 - 4x_{12}y_1 + \lambda\frac{|\beta_2|}{\beta_2} = 0 \tag{18}$$

It can be observed that eq.17 and eq.18 are similar. Hence, these equations can not be uniquely solvable and by substituting any values of β_1 and β_2, the equations remain the same. Hence, these variables β_1 and β_2 have many values that satisfy these optimization criteria.

3 Decision Tree

3.1

3.1.1 Step 1

As the training starts, the root node can be binary split in three ways, Hot vs. Cool, High vs. Low and Cloudy vs. Clear. Now let us compute the entropy at each node for all the three cases.

Hot vs. Cool
Samples at Hot node = 23 rainy, 17 not rainy
Samples at Cool node = 13 rainy, 27 not rainy

Entropy at Hot node = $-(\frac{23}{40}log(\frac{23}{40}) + \frac{17}{40}log(\frac{17}{40})) = 0.2961$

Entropy at Cool node = $-(\frac{13}{40}log(\frac{13}{40}) + \frac{27}{40}log(\frac{27}{40})) = 0.2739$

Entropy = $\frac{1}{2}(0.2961) + \frac{1}{2}(0.2739) = 0.2850$

High vs. Low
Samples at High node = 23 rainy, 17 not rainy
Samples at Low node = 13 rainy, 27 not rainy

Entropy at High node = $-(\frac{23}{40}log(\frac{23}{40}) + \frac{17}{40}log(\frac{17}{40})) = 0.2961$

Entropy at Low node = $-(\frac{13}{40}log(\frac{13}{40}) + \frac{27}{40}log(\frac{27}{40})) = 0.2739$

Entropy = $\frac{1}{2}(0.2961) + \frac{1}{2}(0.2739) = 0.2850$

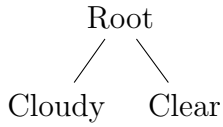
Cloudy vs. Clear Samples at Cloudy node = 25 rainy, 15 not rainy
Samples at Clear node = 11 rainy, 29 not rainy

Entropy at Cloudy node = $-(\frac{25}{40}log(\frac{25}{40}) + \frac{15}{40}log(\frac{15}{40})) = 0.2873$

Entropy at Clear node = $-(\frac{11}{40}log(\frac{11}{40}) + \frac{29}{40}log(\frac{29}{40})) = 0.2554$

Entropy = $\frac{1}{2}(0.2873) + \frac{1}{2}(0.2554) = 0.2713$

Since, Cloudy vs. Clear has the lowest conditional entropy, it has the highest information gain. Also, the accuracy of the tree after this step would be $(25 + 29)/80 = 0.675$. So the tree till here would be



3.1.2 Step2

Now let us repeat the process at Node Cloudy, **Hot vs. Cool (at Cloudy)** Samples at Hot = 15 rainy, 5 not rainy
Samples at Cool = 10 rainy, 10 not rainy

Entropy at Hot node = $-(\frac{15}{20}log(\frac{15}{20}) + \frac{5}{20}log(\frac{5}{20})) = 0.2442$

Entropy at Cool node = $-(\frac{10}{20}log(\frac{10}{20}) + \frac{10}{20}log(\frac{10}{20})) = 0.3010$

Entropy = $\frac{1}{2}(0.2442) + \frac{1}{2}(0.3010) = 0.2726$

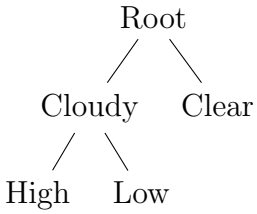
High vs. Low (at Cloudy) Samples at High = 17 rainy, 3 not rainy
Samples at Low = 9 rainy, 11 not rainy

Entropy at High node = $-(\frac{17}{20}log(\frac{17}{20}) + \frac{3}{20}log(\frac{3}{20})) = 0.1835$

Entropy at Low node = $-(\frac{9}{20}log(\frac{9}{20}) + \frac{11}{20}log(\frac{11}{20})) = 0.2988$

Entropy = $\frac{1}{2}(0.1835) + \frac{1}{2}(0.2988) = 0.2411$

Since, entropy is minimum for High vs. Low, it should be split accordingly. The accuracy after the split becomes $\frac{17+11+29}{80} = 0.712$



3.1.3 Step3

Now let us repeat the process at Node Clear, **Hot vs. Cool (at Clear)** Samples at Hot = 8 rainy, 12 not rainy
Samples at Cool = 3 rainy, 17 not rainy

Entropy at Hot node = $-(\frac{8}{20}log(\frac{8}{20}) + \frac{12}{20}log(\frac{12}{20})) = 0.2922$

Entropy at Cool node = $-(\frac{17}{20}log(\frac{17}{20}) + \frac{3}{20}log(\frac{3}{20})) = 0.1835$

Entropy = $\frac{1}{2}(0.2922) + \frac{1}{2}(0.1835) = 0.2378$

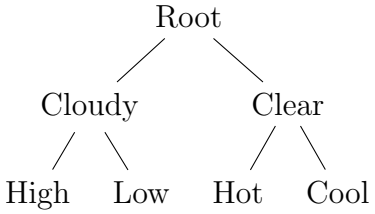
High vs. Low (at Clear) Samples at High = 7 rainy, 13 not rainy
Samples at Low = 4 rainy, 16 not rainy

Entropy at High node = $-(\frac{7}{20}log(\frac{7}{20}) + \frac{13}{20}log(\frac{13}{20})) = 0.2811$

Entropy at Low node = $-(\frac{4}{20}log(\frac{4}{20}) + \frac{16}{20}log(\frac{16}{20})) = 0.2173$

Entropy = $\frac{1}{2}(0.2811) + \frac{1}{2}(0.2173) = 0.2492$

Since, entropy is minimum for Hot vs. Cool, it should be split accordingly. The accuracy after the split remains $\frac{17+11+13+16}{80} = 0.712$. But, since the entropy $0.2492 < entropy(Clear) = 0.2554$, this split can be done.



3.1.4 Step4

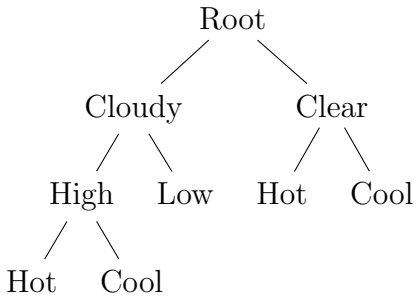
Now let us repeat the process at Node High,
Hot vs. Cool (at High) Samples at Hot = 9 rainy, 1 not rainy
Samples at Cool = 7 rainy, 3 not rainy

Entropy at Hot node = $-(\frac{9}{10}log(\frac{9}{10}) + \frac{1}{10}log(\frac{1}{10})) = 0.1411$

Entropy at Cool node = $-(\frac{3}{10}log(\frac{3}{10}) + \frac{7}{10}log(\frac{7}{10})) = 0.2652$

Entropy = $\frac{1}{2}(0.1411) + \frac{1}{2}(0.2652) = 0.2031$

The accuracy after the split becomes $\frac{9+7+11+29}{80} = 0.712$. Since entropy $0.2031 < entropy(High)$, we can split this node. The tree after split will be,



3.1.5 Step5

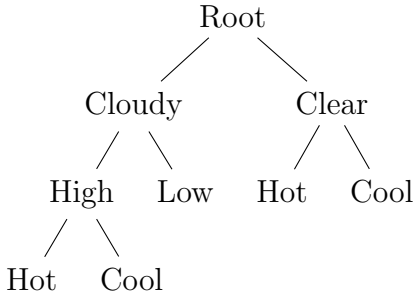
Now let us repeat the process at Node Low,
Hot vs. Cool (at Low) Samples at Hot = 6 rainy, 4 not rainy
Samples at Cool = 3 rainy, 7 not rainy

Entropy at Hot node = $-(\frac{6}{10}log(\frac{6}{10}) + \frac{4}{10}log(\frac{4}{10})) = 0.2922$

Entropy at Cool node = $-(\frac{3}{10}log(\frac{3}{10}) + \frac{7}{10}log(\frac{7}{10})) = 0.2652$

Entropy = $\frac{1}{2}(0.2922) + \frac{1}{2}(0.2652) = 0.2787$

Since entropy $0.2787 > entropy(Low)$, we need not implement this split. The tree after this step will remain same.



3.1.6 Step6

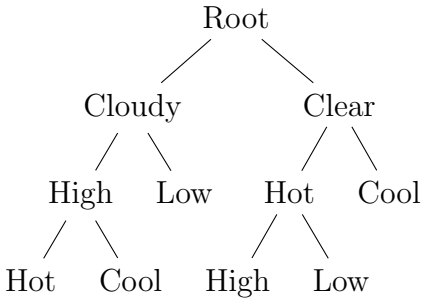
Now let us repeat the process at Node Hot,
High vs. Low (at Hot) Samples at High = 9 rainy, 1 not rainy
Samples at Low = 3 rainy, 7 not rainy

Entropy at Hot node = $-(\frac{9}{10}log(\frac{9}{10}) + \frac{1}{10}log(\frac{1}{10})) = 0.1411$

Entropy at Cool node = $-(\frac{3}{10}log(\frac{3}{10}) + \frac{7}{10}log(\frac{7}{10})) = 0.2652$

Entropy = $\frac{1}{2}(0.1411) + \frac{1}{2}(0.2652) = 0.2031$

Since entropy $0.2031 > entropy(Hot) = 0.2922$, we can split this node. The tree after this step becomes,



3.1.7 Step7

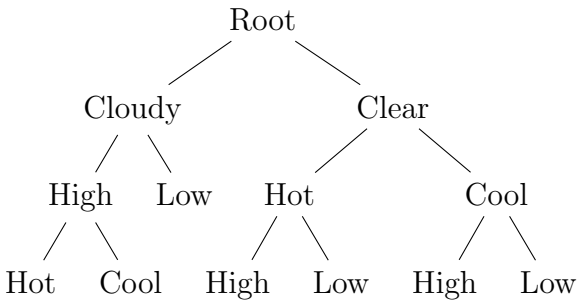
Now let us repeat the process at Node Cool,
High vs. Low (at Cool) Samples at High = 2 rainy, 8 not rainy
Samples at Low = 1 rainy, 9 not rainy

Entropy at High node = $-(\frac{2}{10}log(\frac{2}{10}) + \frac{8}{10}log(\frac{8}{10})) = 0.2173$

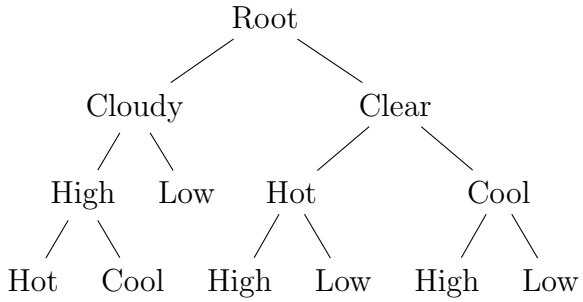
Entropy at Low node = $-(\frac{1}{10}log(\frac{1}{10}) + \frac{9}{10}log(\frac{9}{10})) = 0.1411$

Entropy = $\frac{1}{2}(0.2173) + \frac{1}{2}(0.1411) = 0.1792$

Since entropy $0.1792 > entropy(Cool)$, we can split this node. The tree after this step becomes,



Therefore, by maximizing the information gain, the final decision tree obtained is



3.2

3.2.1 c.i

In the dataset, there are 444 benign samples and 239 malignant samples, which is approximately 65% benign samples to 35% malignant samples. Hence, it can be seen that there is a class imbalance in the dataset. Hence, when splitting the dataset into training and testing data, it has been taken care such that 65% of training samples are benign and 35% malignant and the same with testing data.

3.2.2 c.ii

Two decision trees have been implemented using the training samples. One of the decision tree used gini index for splitting nodes and the other decision tree used entropy for splitting nodes. It has been observed that the decision tree trained using the gini index has performed better in most of the cases (when run multiple times) on the testing samples. Whereas the decision tree with entropy as criteria has performed better most of the time on the training data. It has also been observed that testing accuracy tends to increase as the depth of the tree is increased initially but after a certain depth, the testing accuracy drops as would be expected due to the model getting over-fit on the training samples. These results can be seen in fig.1. In this plot, x-axis is the depth of the decision tree and y-axis is the accuracy of the model on training and testing samples respectively.

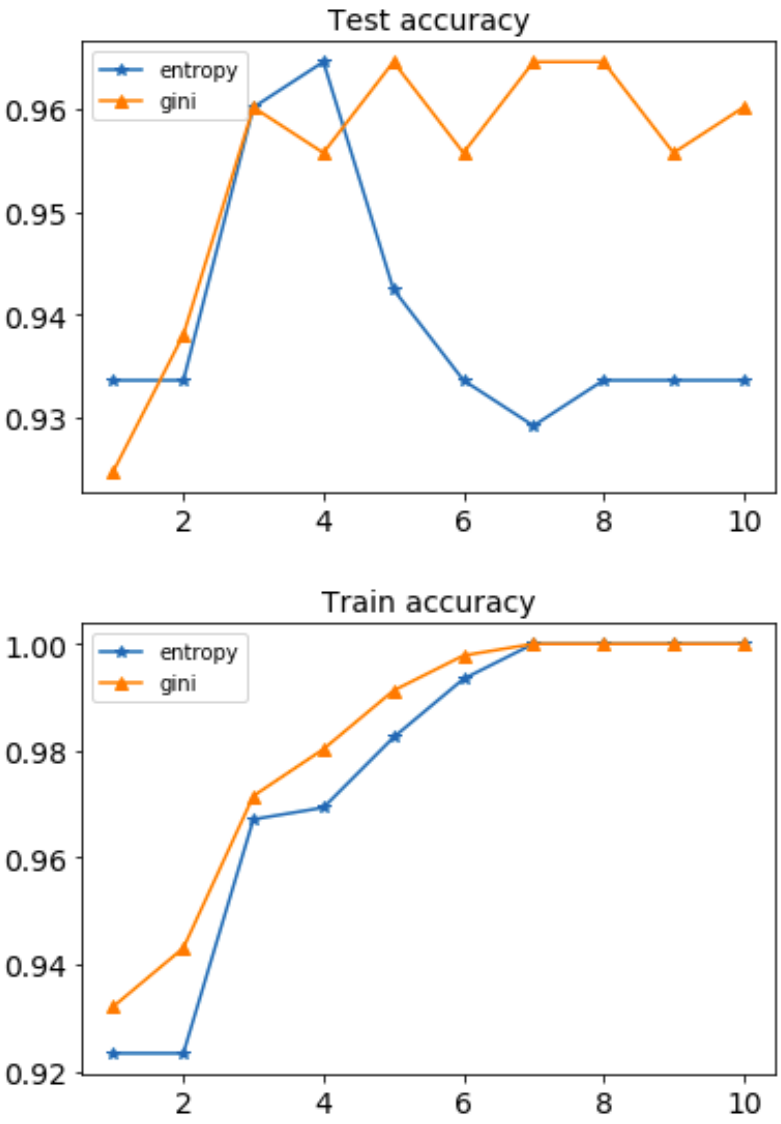


Figure 1: Polynomial regression curves

3.3 c.iii

As mentioned in the question, a 10-fold stratified cross-validation is performed to model a random forest and another 10-fold cross-validation is being performed in each fold of the main 10 fold cross validation to choose the hyper parameters (depth of the tree and the number of trees). Once, these optimal hyperparameters are obtained, they are used for building the main random forest model. The ranking of each feature in the model is obtained using the built-in functions and the importance of each feature is represented by mean and variance from the data collected over the 10-folds. The feature importances are

Feature	Mean	Variance
Clump thickness	0.05	0.0001
Uniformity of cell size	0.28	0.0102
Uniformity of cell shape	0.21	0.0052
Marginal adhesion	0.02	0.0003
Single epithelial cell size	0.11	0.0040
Bare nuclei	0.17	0.0101
Bland chromatin	0.09	0.0027
Normal nuceoli	0.04	0.0006
Mitoses	0.007	0

It can be said that while some features contribute more to the random forest classifier some contribute very less like Mitoses, Normal nuclei etc. It is better to keep all the features in random forest because it could be possible only a few models have these features as the depth of the tree has been controlled, hence keeping these variables does not harm the results that much. The final models list of features can however be a portion of these features that contribute more in order to reduce overfitting the training data.

4 Appendix

4.1 Code for 3.2.i

```
from sklearn import tree
from sklearn import ensemble
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

df = pd.read_csv('hw2_question1.csv')
data = np.array(df.values.tolist())

ind = [i for i in range(9)]
#np.random.seed(1234)
n_benign = np.sum(data[:,9]==2)
n_malignant = np.sum(data[:,9]==4)
print 'Before split'
print '# Benign' + str(n_benign)
print '# Malignant' + str(n_malignant)
print 'Percent Benign ' + str(float(n_benign)/(float(n_benign) +
    float(n_malignant)))
```

```
train_data, test_data, train_labels, test_labels =
    train_test_split(data[:,ind], data[:,9], test_size=0.33)
train_benign = np.sum(data[:,9]==2)
train_malignant = np.sum(data[:,9]==4)
test_benign = np.sum(data[:,9]==2)
test_malignant = np.sum(data[:,9]==4)
print ''
print 'After split'
print 'Percent Train Benign ' +
    str(float(train_benign)/(float(train_benign) + float(train_malignant)))
print 'Percent Test Benign ' + str(float(test_benign)/(float(test_benign)
    + float(test_malignant)))
```

4.2 Code for 3.2.ii

```
gini_te = []
entr_te = []
gini_tr = []
entr_tr = []
for i in range(10):
    i += 1
    clf1 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=i)
    clf2 = tree.DecisionTreeClassifier(criterion='gini', max_depth=i)
    clf1 = clf1.fit(train_data, train_labels)
    clf2 = clf2.fit(train_data, train_labels)
    pred_1 = clf1.predict(test_data)
    pred_2 = clf2.predict(test_data)
    pred_3 = clf1.predict(train_data)
    pred_4 = clf2.predict(train_data)
    entr_te.append(accuracy_score(test_labels, pred_1))
    gini_te.append(accuracy_score(test_labels, pred_2))
    entr_tr.append(accuracy_score(train_labels, pred_3))
    gini_tr.append(accuracy_score(train_labels, pred_4))
    print str(i) + ' ' + str(accuracy_score(test_labels, pred_1)) + ' ' +
        str(accuracy_score(test_labels, pred_2))
plt.plot([i+1 for i in range(10)], entr_te, '*-', label='entropy')
plt.plot([i+1 for i in range(10)], gini_te, '^-', label='gini')
plt.title('Test accuracy', size = 14)
plt.tick_params(labelsize = 14)
plt.legend()
plt.show()

plt.plot([i+1 for i in range(10)], entr_tr, '*-', label='entropy')
plt.plot([i+1 for i in range(10)], gini_tr, '^-', label='gini')
plt.title('Train accuracy', size = 14)
plt.tick_params(labelsize = 14)
```

```
plt.legend()
plt.show()
```

4.3 Code for 3.2.iii

```
df = pd.read_csv('hw2_question1.csv')
data = np.array(df.values.tolist())

ind = [i for i in range(9)]
skf = StratifiedKFold(n_splits=10)
skf.get_n_splits(data[:,ind], data[:,9])

labels = data[:,9]
data = data[:,ind]

feat_imp = []
for train_ind, test_ind in skf.split(data, labels):
    train_data, test_data = data[train_ind], data[test_ind]
    train_labels, test_labels = labels[train_ind], labels[test_ind]
    # perform CV to choose hyperparameter
    best_score = 0
    for max_depth in range(10):
        max_depth += 1
        for n_estimators in xrange(90, 110):
            kf = KFold(n_splits=10)
            kf.get_n_splits(train_data, train_labels)

            accuracy = []
            for train_index, val_index in kf.split(train_data,
                train_labels):
                tr_data, v_data = train_data[train_index],
                    train_data[val_index]
                tr_labels, v_labels = train_labels[train_index],
                    train_labels[val_index]

                clft = ensemble.RandomForestClassifier(criterion='gini',
                    max_depth=max_depth)
                clft = clft.fit(tr_data, tr_labels)
                pred = clft.predict(v_data)
                accuracy.append(accuracy_score(v_labels, pred))
                del clft, tr_data, v_data

            if np.mean(np.array(accuracy)) > best_score:
                best_depth = max_depth
                best_ntrees = n_estimators
```

```
        print best_depth
        print best_ntrees
        clf = ensemble.RandomForestClassifier(criterion='gini',
            max_depth=best_depth)
        clf.fit(train_data, train_labels)
        feat_imp.append(clf.feature_importances_)

        del clf, train_data, test_data, train_labels, test_labels
    feat0_imp = np.mean([i[0] for i in feat_imp])
    feat1_imp = np.mean([i[1] for i in feat_imp])
    feat2_imp = np.mean([i[2] for i in feat_imp])
    feat3_imp = np.mean([i[3] for i in feat_imp])
    feat4_imp = np.mean([i[4] for i in feat_imp])
    feat5_imp = np.mean([i[5] for i in feat_imp])
    feat6_imp = np.mean([i[6] for i in feat_imp])
    feat7_imp = np.mean([i[7] for i in feat_imp])
    feat8_imp = np.mean([i[8] for i in feat_imp])

    print np.var([i[0] for i in feat_imp])
    print np.var([i[1] for i in feat_imp])
    print np.var([i[2] for i in feat_imp])
    print np.var([i[3] for i in feat_imp])
    print np.var([i[4] for i in feat_imp])
    print np.var([i[5] for i in feat_imp])
    print np.var([i[6] for i in feat_imp])
    print np.var([i[7] for i in feat_imp])
    print np.var([i[8] for i in feat_imp])

    print feat0_imp
    print feat1_imp
    print feat2_imp
    print feat3_imp
    print feat4_imp
    print feat5_imp
    print feat6_imp
    print feat7_imp
    print feat8_imp
```