Supervised Learning: Assignment 1

Tiffany Woodley

April 2019

Contents

List of Figures

List of Tables

1 Question 1

The objective of question 1 was to generate a model which is able to accurately predict the median house value of the areas within the greater Boston area.

1.1 Linear Regression

The initial investigation looked into a linear model for the purpose of predicting the median house values. The first model created included all of the predictor variables as to generate a base model to improve upon.

1.1.1 Linear Model fit and test statistics

The printed summary of the fitted model gives us an indication of the performance. The residual standard error (RSE) 4.68 (which translates to 4 680 dollars). This provides an estimate of the standard deviation of the irreducible error.

The R^2 represents how much of the variance in the dependent-variable can be explained by the model, the remaining percentage (1-R^2) indicates towards the irreducible error inherent in the model. The adjusted R^2 takes into account the R^2 value and penalises the value on the basis of having more predictor variables, giving a more accurate representation of the R^2 value.

The fitted model returned an adjusted R^2 value of 0.7168. In 1.1.2 we look into interaction terms in order to try and increase the R^2 value.

The F-statistic is fairly high and the p-value close to zero, this indicates that we can reject the null hypothesis and assume their is a high chance that their is a relationship between the median house price and the predictor variables

Model Metric	Value
RSE	4.68
Multiple R^2	0.729
Adjusted R^2	0.7168
F-statistic	59.84
p-value	$<2.2e^{-16}$

Table 1: Base linear model summary values

The below table represents the statistical information about the fitted model's predictors variables. This representation shows us the most significant variables are chas ,nox ,rm ,dis ,ptratio and lstat. These variables are seen as significant as they have higher t values(which tell us how many standard deviations the estimated coefficient lies away from zero) and very low p values(the probability that the t value will occur for the regression coefficient)

Coefficient	Estimate	Std.Error	t value	p value	significance
Intercept	34.22978	6.91806	4.948	1.33e-06	***
crim	-0.09109	0.05724	-1.591	0.112713	
zn	0.03679	0.02046	1.799	0.073196	.
indus	0.03279	0.08706	0.377	0.706748	
chas	4.58229	1.35201	3.389	0.000807	***
nox	-23.53052	5.03766	-4.671	4.76e-06	***
rm	4.92549	0.64221	7.670	3.23e-13	***
age	-0.00742	0.01822	-0.407	0.684085	
dis	-1.43437	0.27058	-5.301	2.42e-07	***
rad	0.28298	0.08709	3.249	0.001305	**
tax	-0.01124	0.00504	-2.229	0.026646	*
ptratio	-0.92835	0.17452	-5.320	2.20e-07	***
lstat	-0.43609	0.07507	-5.809	1.78e-08	***

Table 2: Base linear model variable statistics

When performing cross validation, the mean squared error (MSE) averaged at 23.68. The base model returned a MSE of 27.27 when applied to the test set, this indicates how well the model generalizes, and is the error score to beat.

1.1.2 Multi-collinearity

Multi-collinearity occurs when predictor variables are highly correlated and they interfere with each other.This in turn increases the standard errors of the coefficients. This leads to increased variability of the beta values when fitting the model.

The table below shows the Variable Inflation Factor (VIF) scores (an indication the multi-collinearity) of the model:

The variables with high multi-collinearity were removed, and the model was refitted in order to check the significance of the predictor variables without large effects of multi-collinearity.

The new adjusted R^2 value is 0.7077 - which is very similar to the first R^2 value we got.
This just goes to prove that even though the predictors we removed were considered significant in the first model, they were not necessarily adding anymore useful information to the fitted model.
As was expected, the standard errors of the variables have decreased. The model now would suggest that the only variables of significance are chas ,nox ,rm ,dis

Coefficient	VIF
crim	2.12
zn	2.51
indus	4.46
chas	1.18
nox	4.35
rm	2.17
age	3.11
dis	4.14
rad	7.35
tax	9.42
ptratio	1.86
lstat	3.32

Coefficient	VIF
crim	3.69
zn	2.35
indus	3.48
chas	1.13
nox	4.60
rm	2.44
age	3.40
dis	4.03
rad	-
tax	-
ptratio	1.53
lstat	3.84

Table 3: Left: VIF values of all variables Right: VIF values after removing correlated variables

,ptratio and lstat.

Although these variables were removed here for the purpose of investigating variable significance,they were left in for variable selection. As they may become useful when adding in interaction terms due to the hierarchical principle which states that if you want to include an interaction you have to include the main effects.

Coefficient	Estimate	Std.Error	t value	p value	significance
(Intercept)	26.24654	6.33696	4.142	4.62e-05	***
crim	-0.02574	0.05154	-0.500	0.617822	
zn	0.03132	0.01997	1.568	0.117972	
indus	-0.04168	0.07234	-0.576	0.565026	
chas	4.67346	1.36647	3.420	0.000723	***
nox	-19.74694	4.79994	-4.114	5.17e-05	***
rm	5.31045	0.63322	8.386	2.85e-15	***
age	-0.01413	0.01834	-0.770	0.441706	
dis	-1.43867	0.27484	-5.235	3.34e-07	***
ptratio	-0.79001	0.15973	-4.946	1.34e-06	***
lstat	-0.42206	0.07557	-5.585	5.72e-08	***

Table 4: Variable statistics with less affect of multi-collinearity

1.2 Improving the linear model

1.2.1 Data Exploration

Before variable selection, the data was explored in order to see if any relationships could be identified. A pair scatter plot/correlation matrix was plotted in order to help visualise these relationships. The blue dots represent the points at which the house median value is such that it is above the mean value; the green dots are representative of the converse.

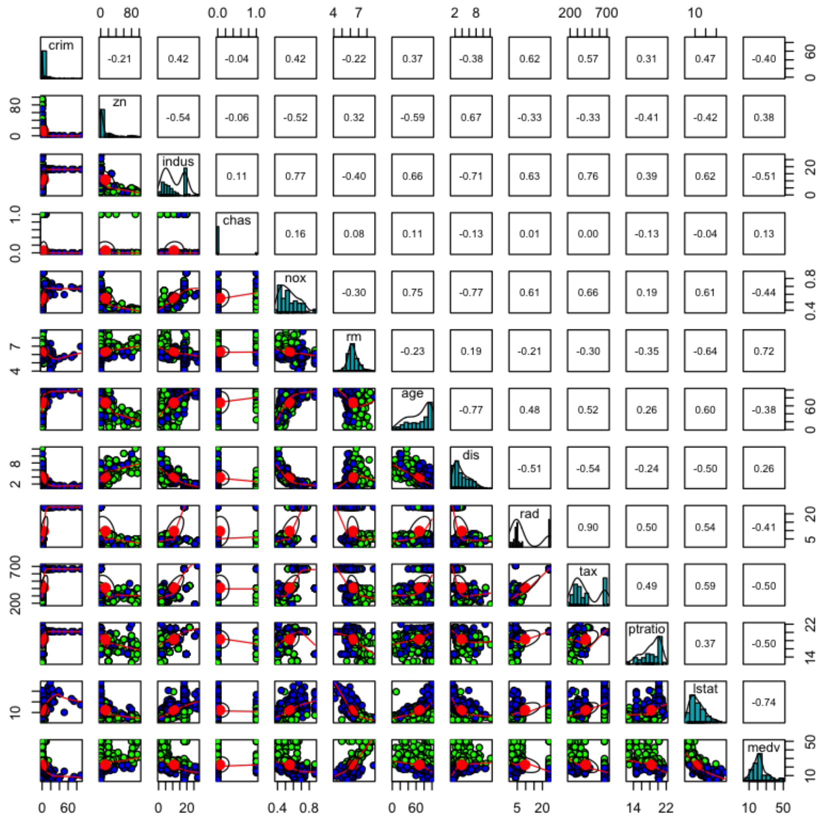


Figure 1: Pair scatter-plot/correlation matrix of variables

Some of the noticeable elements of the scatter plots are:

- The predictor variables - lstat, indus and crim - have an exponential relationship with the median house value.
- The predictor variables that have the highest correlation with the median house value - as shown by a linear relationship - are those likely to be

the variables of the most significance in the model. The variables with the strongest visual linear relationships are lstat and rm. This is substantiated by the correlation values of in order -0.74 and 0.72. The fact that these relationships have a distinct separation between the blue and green dots also helps us to see that these variables will help with predicting the house value.

- The predictor variables which have relationships between themselves, can help to determine which interaction terms could be beneficial to the model. For instance an exponential relationship between nox and dis can be seen, this is expected as the closer you are to a freeway the more likely there is to be higher NO_2 emissions.
- The correlation values also help to understand the VIF scores we saw earlier, for example rad and tax are 90 percent correlated. They are both also highly correlated with indus. This can also be visualised through their distributions, as they take on similar shapes.

In the below figure, we can see the three predictor variables which have an exponential relationship with the median house value. The log of the variables transforms them in order to become more linear, as can be seen by the correlation values in the top right of the plots.
These variables were transformed before variable selection, as they will now be better predictors for the model.

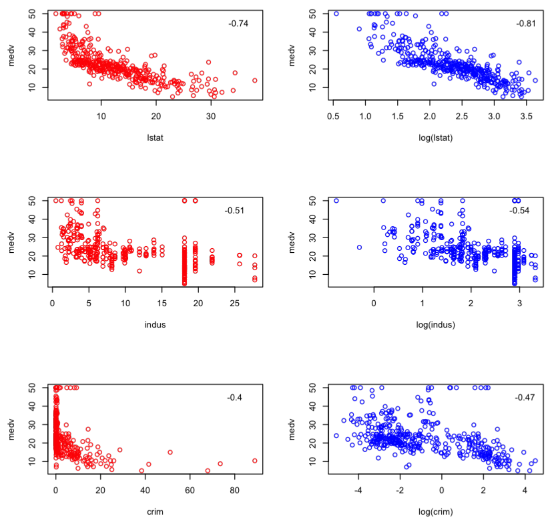


Figure 2: Log transforms of the exponential variables

Some of the interaction terms which were thought to have the potential of being good interaction terms were

- **indus and dis**
  - The relationship between the proportion of non-retail business acres per land and the distance to employment centres - the further you are away from an employment centre the less likely it is that the land will be used for retail use.
- **tax and lstat**
  - The relationship between the tax per 1000 dollars and the lower status of the population - The lower status areas are more likely to pay less tax.
- **ptratio and lstat**
  - The relationship between the pupil to teacher ratio of town and the lower status of the population - The lower status areas are more likely to have a higher pupil to teacher ratio
- **nox and rad**
  - The relationship between the amount of NO_2 concentration and how close the area is to highways - The closer you are to a highway the greater the chances are that you are exposed to higher emmisions from the traffic.
- **crim and lstat**
  - The relationship between crime in area and the lower status of the population - The lower status areas are more likely to have higher crime rates due to circumstance
- **chas and dis**
  - The relationship between if the area is bound by a river and the per capita crime rate of the town - The closer you are to a river the more vulnerable you would be to crime.

The interaction plots represented in the below figure show an indication of how strongly the variables interact with one another. A linear model was created using only two predictor variables, which were thought to have had a causal relationship.
Subsequently, the fitted linear relationship with the median house value was plotted whilst one of the variables was held constant for two distinct values (red = 0.1 and blue = 0.9).
If the line gradients differ, it indicates that the linear relationship with one of the variables changes with different values of the other variable.

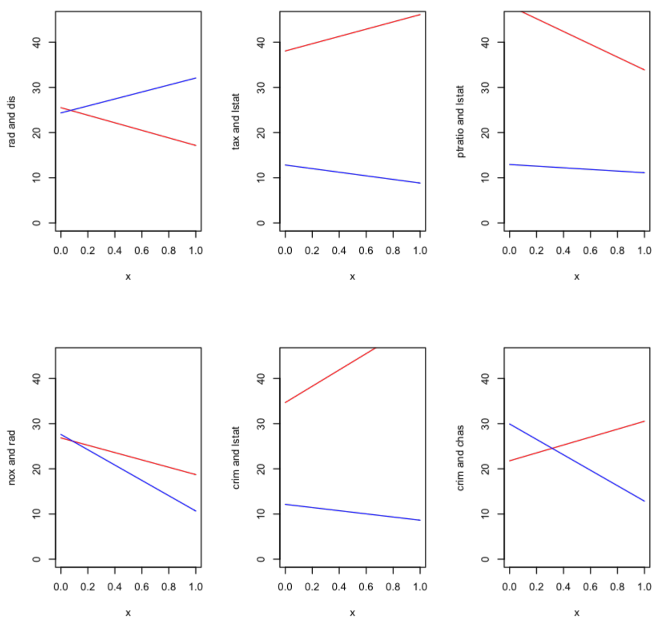


Figure 3: Interaction Plots

None of these plots have parallel lines which means they all have the potential to be interaction terms and will be left in for variable selection.

1.2.2 Variable selection methods

After the interaction terms were added, the process of variable selection began. The different techniques applied were best subset selection(exhaustive method), forward selection, backwards elimination, a combination of forwards and backwards elimination, and lasso regression. These were then evaluated based on their cross validated MSE. The results of these methods can be viewed in the table below.

Since we are using interaction terms all the input variables have been scaled otherwise the one part of the interaction term may have overpowered the other.

Selection method	CV MSE	No.of vars	Variables removed
exhaustive method	19.357	11	crim, chas, age, inter 1, inter 3, inter 5
forward selection	19.311	13	crim, chas,age,inter 1
backward elimination	19.366	13	crim, chas, age, inter 1
Combination	18.86	16	chas
lasso regression	20.69	16	chas

Table 5: Linear variable selection results

The lowest cross validation (CV) error was obtained by using the combined method of forward selection and backward elimination; with the fitted model using 16 out of the 17 variables. Even though this indicated towards the best model, the final model was actually fitted with the 11 variables that the exhaustive method had returned. This results in a much simpler model, for only a slight increase in CV error.

The exhaustive method would have run through all the model possibilities the forward, backward and both methods would have fitted. The fact that they are returning lower CV error results is an indicator to the variability of the MSE due to the small number of observations in the data set.

The model returned by the exhaustive method was subsequently fitted to the entire training set, and examined as the final model on the test set.

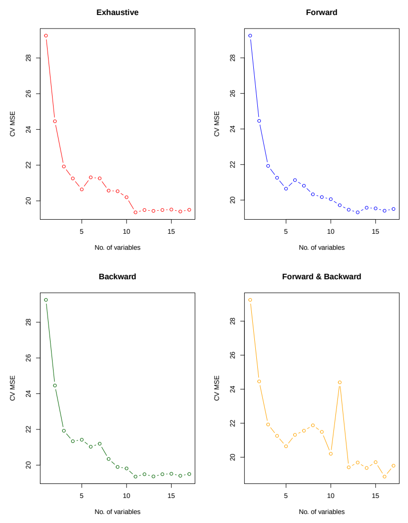


Figure 4: Step method cross validation results

1.2.3 Final model and evaluation on the test set

The final model applied in effect brought down the test MSE by 2.79, the residuals plot as shown below indicates that the final model captured the underlying function well. The adjusted R^2 value has increased to 0.7921 highlighting the fact that the interaction terms have brought more useful information to the model.
The residuals have no distinct pattern, and are evenly distributed above and below the zero line, with only one serious outlier.
The QQ plot shows the residual "noise" is not quite a normal distribution, which is an assumption we made when we chose to fit a linear model.
Since the residuals look like random noise, I don't think it would be necessary to attempt to use a more flexible method. The only way it could improve the fit would be to start over-fitting to the noise which would decrease the bias slightly but increase the variance more substantially.

	Train MSE	CV MSE	Test MSE
Base Model	21.51958	23.68	27.27
Final Model	17.48668	19.357	24.48

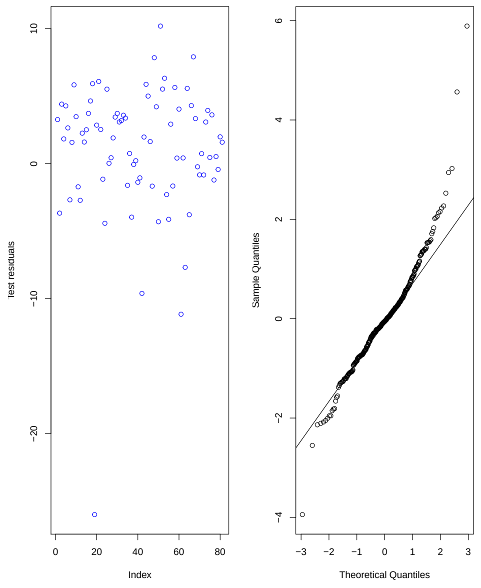


Figure 5: Left: Final linear model residual plot Right: Final linear model QQ plot

1.3 Regression Tree

In this section we looked into the application of regression trees as a tool in order to try and solve the same problem:

1.3.1 Growing a large tree

The first step taken was to relax the stopping criteria, this was achieved by setting the minimum deviance to 0.001. As a result a very deep tree was grown - having 25 terminal nodes. This large tree allowed over fitting to occur, as seen by the train MSE being a lot lower than the test MSE highlighting the fact that the tree will not generalize well.

Train MSE	Test MSE
11.44893	29.90289

The predictor variables the model made use of are

- rm
- lstat
- crim
- rad
- tax
- dis
- nox
- indus

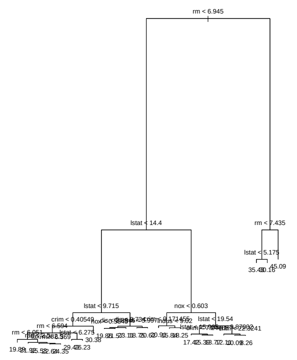


Figure 6: Tree fitted with relaxed criteria

1.3.2 Finding the alpha values

The plot below shows the alpha values and their associated deviance values. Each alpha values has a tree size (number of terminal nodes) that it will force the tree down too in subsequently creating a subtree. These alpha values were used in the next step to find the best subtree using cross validation to decide between them.

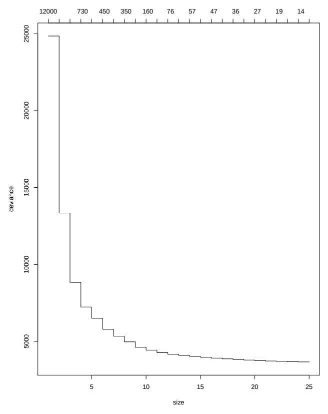


Figure 7: Regression tree: alphas vs deviation

1.3.3 Pruning the tree using cross validation

From the figure below we can see that the cross validated deviance plateaus at the lowest value a when the tree has a size of 6 terminal nodes and so this is the value we used to prune the tree. This associates with the alpha value of 448.27831

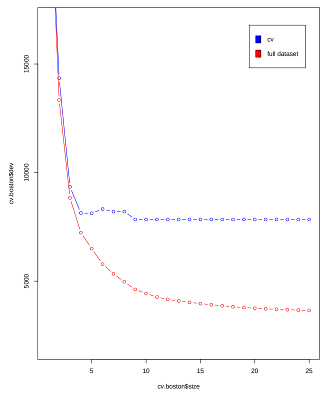


Figure 8: Cross validation for different tree sizes

1.3.4 Testing the final model

The newly pruned tree is a lot more simple - only having 6 terminal nodes, as seen below. It only includes the predictor variables

- rm
- lstat
- nox

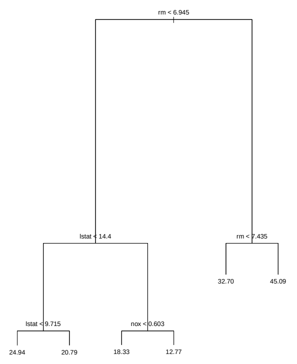


Figure 9: Final tree after pruning

The newly pruned tree was used to predict both the training and the test set. Although the train MSE has come up substantially the test MSE has come down by 3.41 indicating a better model for generalization

Train MSE	Test MSE	
20.12368	26.49217	

The residuals look random around the zero line indicating the model captured the underlying function. The predictions can take on one of 6 values depicted by the mean of the data at the different terminal nodes. This is visualised by the 6 horizontal lines running across the linear line.

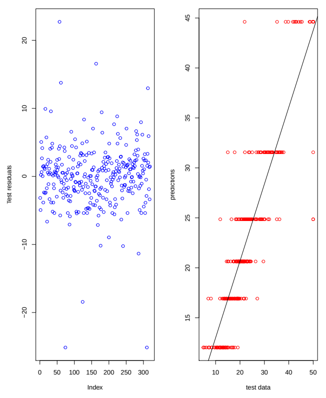


Figure 10: Left:Regression tree residual plot Right: Regression tree fitted vs predicted

1.4 Comparison

The linear regression out performed the regression tree slightly, this points to the data having a linear shape. If the data was non linear the decision tree would have done a better job capturing the non-linearity of the data than linear regression. It would have achieved this by dividing the space up into smaller subspaces.
The overlaid residual plot show that on this data they are pretty evenly matched. It should also be considered that with it being such a small data set the MSE has quite high variance and might be slightly biased.

	Train MSE	Test MSE
Linear Regression	17.48	24.48
Regression Tree	20.12	26.49

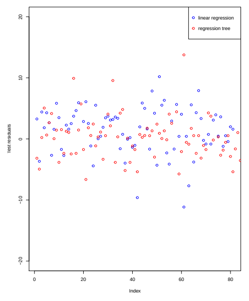


Figure 11: Linear and regression tree residual plot comparison

2 Question 2

In this question the goal is to create a model which can accurately predicts whether an email is spam or not based on a large number of predictor variables. The data set has 2141 email entries and 1460 spam entries, so the model has to beat a 59 percent correct classification rate for it to be an improvement. For this question more emphasis is not going to be put onto either spam and email and so the area under the ROC curve (AUC) will be used as the metric to rate and compare the different models with.

2.1 Data Exploration

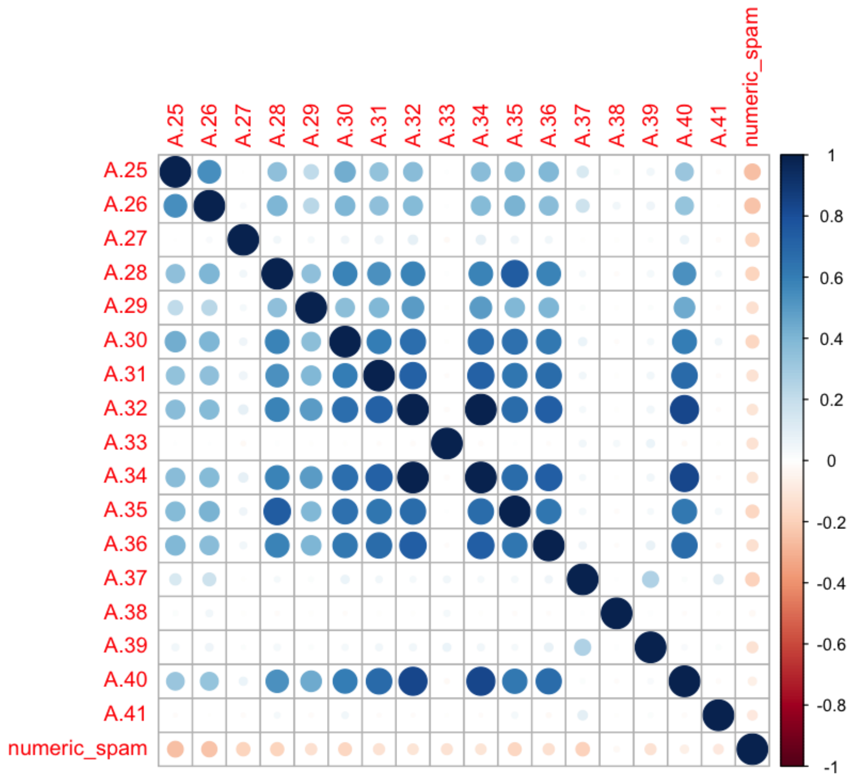


Figure 12: Spam data correlation matrix

The data was explored to see which would be potential good predictor variables. To analyse how effective the variable selection techniques were. First the correlation between the predictor variables was explored to see if all the variables were actually needed. Most of the correlation happened between the middle variables as seen in the matrix above, the rest of the variables were cut off as they didn't have significant correlation. The highest correlation resulted between A.32 and A.34(can be seen visually in figure 13). Since we preforming variable selection I have left in all the variables to compare the different variable selection processes and see if they help get rid of this phenomenon.

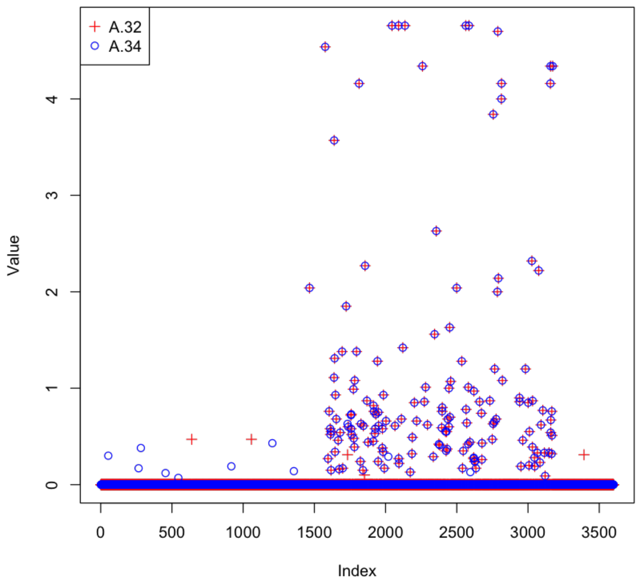


Figure 13: A.32 and A.34 comparison

The next aspect that was looked into was box and whisker plots as seen in the figure below. Only the first 8 variables are plotted and explored here as there are 57 variables. From these plots A.4 looks like it will not be a good predictor variable - There is very little variability in the values with the large majority of the points being zero for both email and spam. A.7 looks like it would be a good variable as there is a difference in the variance between the email and the spam. So it shows if the variable is above zero it is most likely to indicate towards spam.

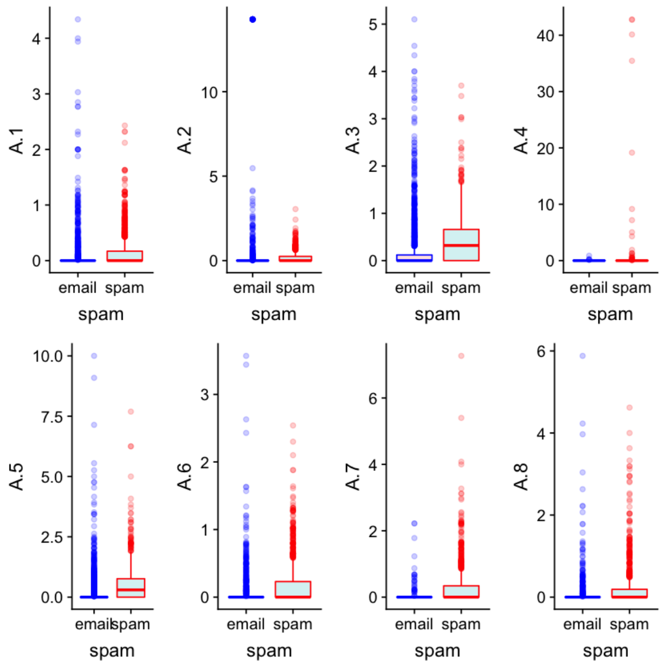
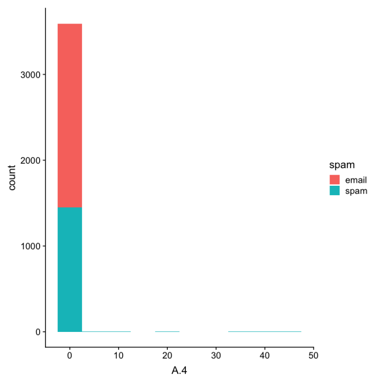
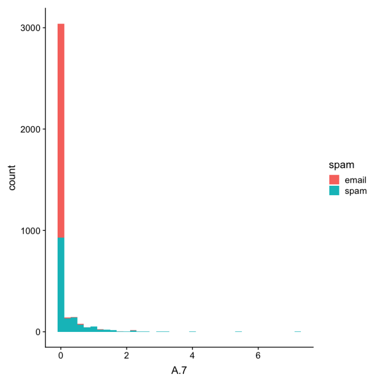


Figure 14: Variance plots for first 8 predictor variables

The stacked histogram of A.4 reiterates the fact that the variable will not be good at predicting, will mostly all the points for both email and spam being 0. The histogram for A.7 points to the fact that if the variable is above zero it is probably spam.



(a) A.4



(b) A.7

Figure 15: Stacked histograms

Since we will be using multiple variable selection techniques and this will be a good exercise to compare them, all the variables will be left in.

2.2 Logistic Regression

2.2.1 Initial model

The first logistic regression model was fitted to all the parameters, below you can see the summary results of the model. The deviance residuals have a median of zero and the spread looks similar on both sides which gives us a good indication we have captured the underlying function of the data. The residual deviance is lower than the null deviance which is a good sign as is means the model is performing better than if there was no model and so is adding value. The AIC value is higher than the residual deviance as it has been adjusted for the number of predictors in the model. From the fisher scoring iterations we can see the model converges to the estimates quite quickly.

The log likelihood with having no model($ll.null$) is $Nulldeviance/-2 = -2431.25$ The log likelihood with having a mode($ll.model$)l is $Residualdeviance/-2 = -1188.3$ Psuedo R^2 is $ll.null - ll.model/ll.null = 0.511$

Metric	Value
Deviance Residual-min	-4.2235
Deviance Residual-1Q	-0.1646
Deviance Residual-median	0.0000
Deviance Residual-3Q	0.1110
Deviance Residual-max	5.0170
Null deviance	4862.5
Residual deviance	1377.0
AIC	1493
Fisher Scoring iterations	13

Table 6: Logistic regression summary values

The significance of the predictors based on their p-values can be seen in the graph below. The longer the line the smaller the significance. It is interesting to see that A.7 a variable found to be a good predictor variable through data exploration is found to be the most significant, A.4 which was found to not be a good variable through data exploration had a large p value of 0.474.

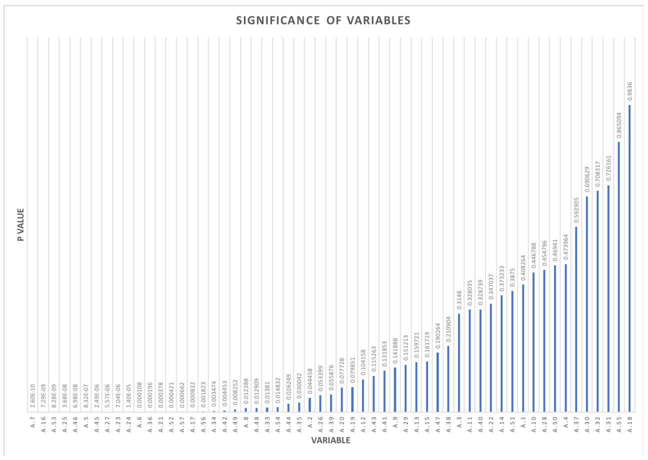
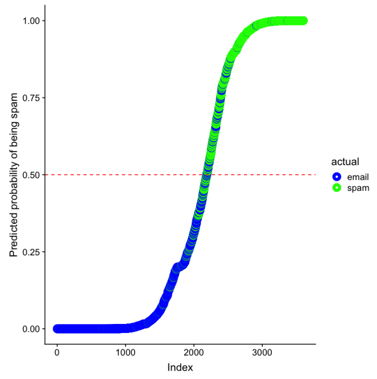


Figure 16: Logistic regression variable significance

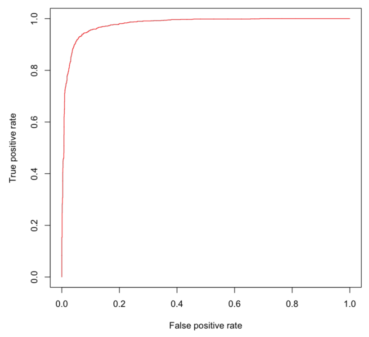
The table below shows you the in sample errors of the model. This gives us a classification rate of 93.31 percent Which is an improvement when compared to the classification rate if we had no model of 57%

	Email	Spam
Email	2044	144
Spam	97	1316

Below we can see the the prediction curve of the in sample points and can see the model is doing a pretty good job at distinguishing them with it separating the colors fairly well. The in sample ROC curve is also shown; having an AUC of 0.9789984. The cross-validation AUC of this model is 0.9726633



(a) Probabilities associated with different classes

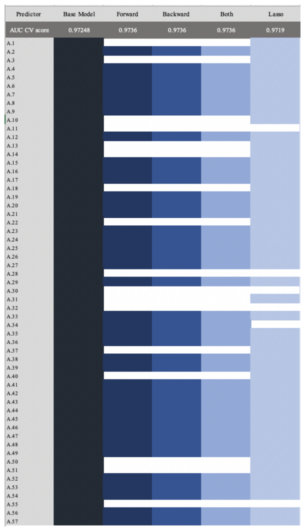


(b) In sample ROC curve

Figure 17: Prediction accuracy graphs for logistic regression

2.2.2 Variable selection

The graphic below gives an overview of how the different techniques performed(the variables chosen are coloured for each technique). It is interesting to see that all techniques removed A.32 which gives indication to the fact that they do help with multi-collinearity. The variable A.4 which was seen to be not a very good predictive variable was left in by all the techniques, which just goes to show they are not perfect. The technique which returned the best model was all the stepwise techniques(as they all ended up returning the same model) having the highest cross validation AUC score. And so this was chosen to be the final model for logistic regression.



Predictor	Base Model	Forward	Backward	Both	Lasso
AUC CV score	0.97248	0.9736	0.9736	0.9736	0.9719
A.1	■				■
A.2	■	■	■	■	■
A.3	■				■
A.4	■	■	■	■	■
A.5	■	■	■	■	■
A.6	■	■	■	■	■
A.7	■	■	■	■	■
A.8	■	■	■	■	■
A.9	■	■	■	■	■
A.10	■				■
A.11	■				
A.12	■	■	■	■	■
A.13	■				■
A.14	■				■
A.15	■	■	■	■	■
A.16	■	■	■	■	■
A.17	■	■	■	■	■
A.18	■				■
A.19	■	■	■	■	■
A.20	■	■	■	■	■
A.21	■	■	■	■	■
A.22	■				■
A.23	■	■	■	■	■
A.24	■	■	■	■	■
A.25	■	■	■	■	■
A.26	■	■	■	■	■
A.27	■	■	■	■	■
A.28	■				
A.29	■	■	■	■	■
A.30	■				
A.31	■				■
A.32	■				
A.33	■	■	■	■	■
A.34	■	■	■	■	
A.35	■	■	■	■	■
A.36	■	■	■	■	■
A.37	■				■
A.38	■	■	■	■	■
A.39	■	■	■	■	■
A.40	■				■
A.41	■	■	■	■	■
A.42	■	■	■	■	■
A.43	■	■	■	■	■
A.44	■	■	■	■	■
A.45	■	■	■	■	■
A.46	■	■	■	■	■
A.47	■	■	■	■	■
A.48	■	■	■	■	■
A.49	■	■	■	■	■
A.50	■				■
A.51	■				■
A.52	■	■	■	■	■
A.53	■	■	■	■	■
A.54	■	■	■	■	■
A.55	■				
A.56	■	■	■	■	■
A.57	■	■	■	■	■

Figure 18: Logistic variable selection results

2.2.3 Final model

The final model performed well on the test set. The test, CV and train AUC are close together giving a good indication that the model will generalize well.

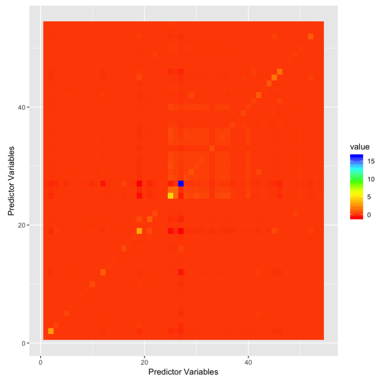
CV AUC	0.973
test AUC	0.967
train AUC	0.978

2.3 Discriminant Analysis Assumptions

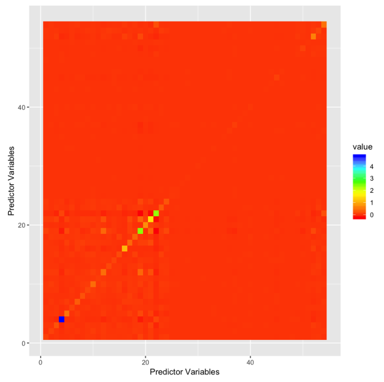
The assumptions that need to be met for discriminant analysis are

- The predictor variables are independent from each other
- The predictor variables have a normal distribution
- The predictor variables for the different classes have the same covariance matrix - for LDA, when using QDA this assumption is relaxed
- The predictor variables for the different classes have different mean values

The heat plots below show the covariance matrices for the separate spam and email classes, it is clear to see they are not equal to each other and look quite different. This breaks the LDA assumption that the classes have equal covariance matrices.



(a) Email



(b) Spam

Figure 19: Covariance Matrices

Test	p value	meaning
BoxM	$2.2e^{-}16$	multivariate normality does not hold
Avg. Levene	0.018	covariances are not equal

Table 7: Discriminant analysis assumption tests

The qq plot shows that the multivariate normality does not hold. The line the data produces is far off the normal line. This was also indicated by the BoxM test.

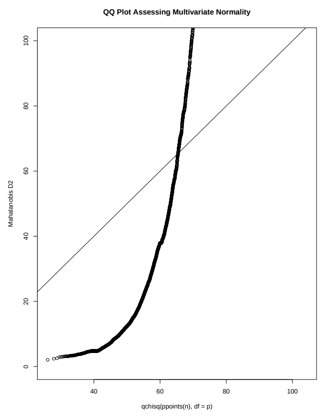


Figure 20: Multivariate QQ plot

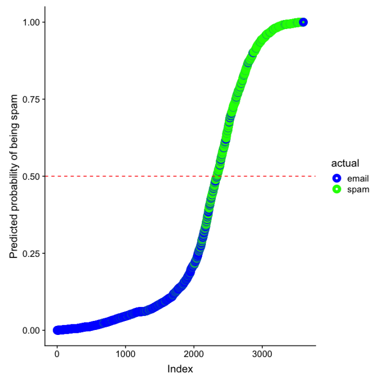
2.4 LDA

2.4.1 Initial Model

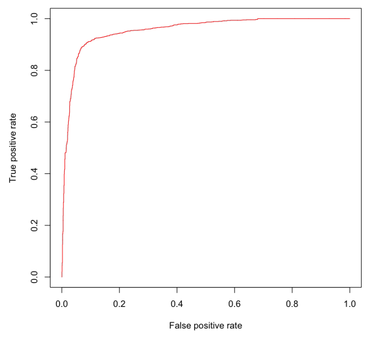
The table below shows you the in sample errors of the model. This gives us an classification rate 89.28 percent

	Email	Spam
Email	2046	291
Spam	95	1169

Below we can see the the prediction curve of the in sample points and can see the model is doing a pretty good job at distinguishing them with it separating the colors fairly well. Although it seems dropping the decision boundary slightly could be beneficial. The in sample ROC curve is also shown; having an AUC of 0.9549887 The cross-validation AUC of this model is 0.9519



(a) Probabilities associated with different classes

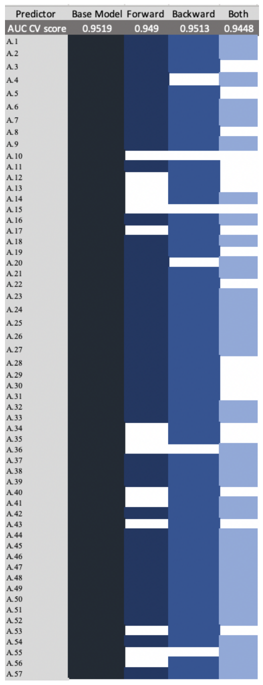


(b) In sample ROC curve

Figure 21: LDA prediction accuracy graphs

2.4.2 Variable Selection

The stepwise functions all returned different models. None of them removed A.32 or A.34 indicating they do not work as well with removing multicollinearity in the LDA setting. The best model based on the cross validated AUC is the base model. This also gives indication to these methods not working as well for LDA. The lasso function came back with a cross validated AUC score of 87.57



Predictor	Base Model	Forward	Backward	Both
AUC CV score	0.9519	0.949	0.9513	0.9448
A.1	■	■	■	■
A.2	■	■	■	■
A.3	■	■	■	
A.4	■	■		■
A.5	■	■	■	
A.6	■	■	■	■
A.7	■	■	■	■
A.8	■	■	■	
A.9	■	■	■	■
A.10	■			
A.11	■	■	■	
A.12	■		■	
A.13	■		■	
A.14	■		■	■
A.15	■			
A.16	■	■	■	■
A.17	■		■	
A.18	■	■	■	■
A.19	■	■	■	
A.20	■	■		■
A.21	■	■	■	■
A.22	■	■	■	
A.23	■	■	■	■
A.24	■	■	■	■
A.25	■	■	■	■
A.26	■	■	■	■
A.27	■	■	■	■
A.28	■	■	■	
A.29	■	■	■	
A.30	■	■	■	
A.31	■	■	■	
A.32	■	■	■	■
A.33	■	■	■	■
A.34	■		■	
A.35	■		■	
A.36	■			■
A.37	■	■	■	■
A.38	■	■	■	■
A.39	■	■	■	■
A.40	■		■	
A.41	■		■	■
A.42	■	■	■	■
A.43	■		■	
A.44	■	■	■	■
A.45	■	■	■	■
A.46	■	■	■	■
A.47	■	■	■	■
A.48	■	■	■	■
A.49	■	■	■	■
A.50	■	■	■	■
A.51	■	■	■	■
A.52	■	■	■	■
A.53	■		■	
A.54	■	■	■	■
A.55	■			■
A.56	■		■	■
A.57	■	■	■	■

Figure 22: LDA variable selection results

2.4.3 Final Model

The final model didn't perform as well as logistic regression. This is probably because non of the assumptions for LDA held. The predictor variable distributions for different classes probably had quite a bit of overlap resulting in a lower performance.

CV AUC	0.9519
test AUC	0.946
train AUC	0.955

The distributions of the two classes have quite a large overlap with means that are fairly close together.

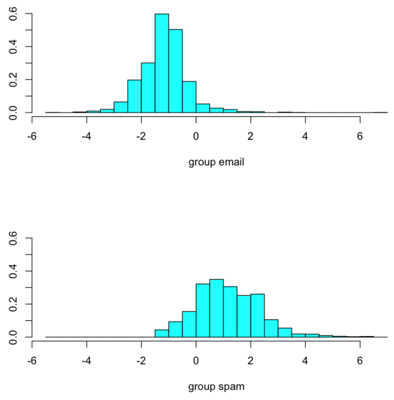


Figure 23: LDA class distributions

Since the assumptions do not hold true for discriminate analysis and the fact that QDA performed worse than LDA when fitted with all the variables, the technique was not explored.

2.5 Random Forest

A random forest was fitted to the data and a grid search was done to find the best model which generalised well. The results of the grid search came back with the best model being

no. of trees	125
number of variables per model	4
max depth of trees	9

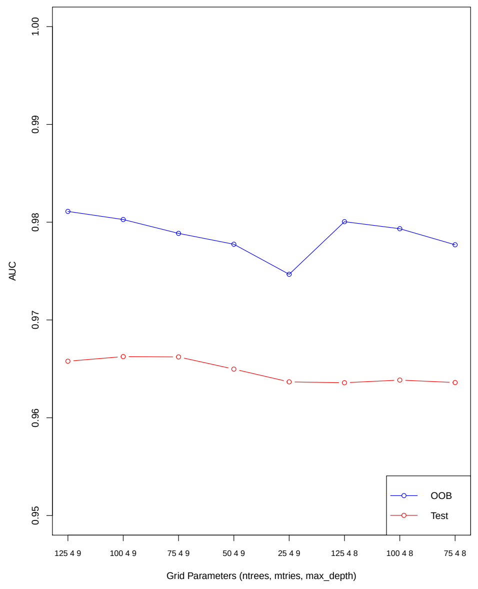


Figure 24: Random forest grid search results

The plot below shows the variable importance of the top 10 variables, it came back with A.7 being the most significant, which is something we also saw in the logistic model. The variable importance score is calculated based on the variable being selected to split on whilst growing the trees and how much of the total MSE improved as a consequence.

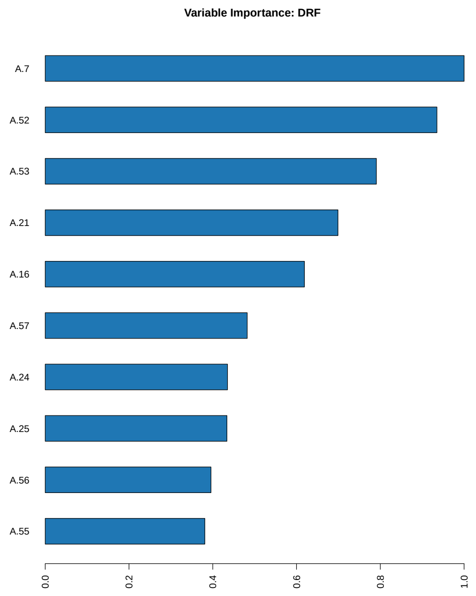
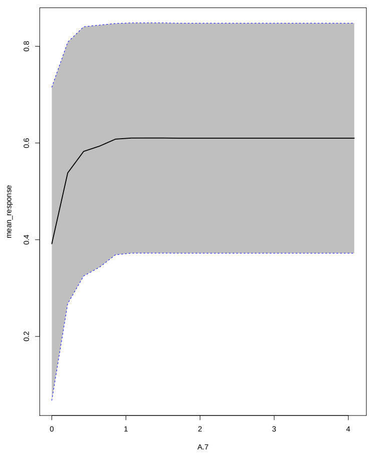
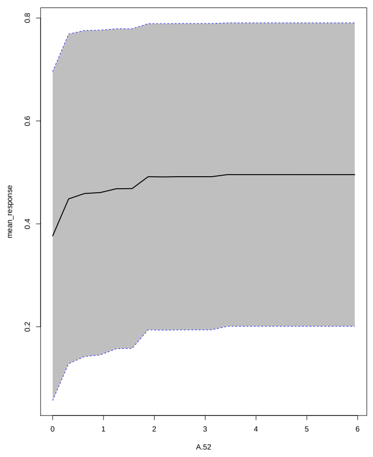


Figure 25: Random forest variable importance

The variable importance is explored through partial dependence plots, A.7 shows that the difference between being zero and one has a substantial jump in the mean response and evens out after that, this links back to our data exploration, if A.7 is above 1 it is most likely to be spam. A.52 which was seen as less important has a smaller effect on the mean response as it's value changes but the change happens over the larger range of the variable.



(a) A.7



(b) A.52

Figure 26: Partial dependence plots

The final model performed extremely well on the test set, also having a training and test AUC close together indicating the model has not over or under fit.

AUC on training set	0.9879104
AUC on test set	0.9859394

2.6 Boosted Trees

A boosted tree model was fitted to the data and a grid search was done to find the best model which generalised well. The results of the grid search came back with the best model being

no. of trees	100
max. depth of trees	6
learning rate	0.5

The grid search favoured a bigger learning rate, highlighing the fact that slow learning results in a better model.

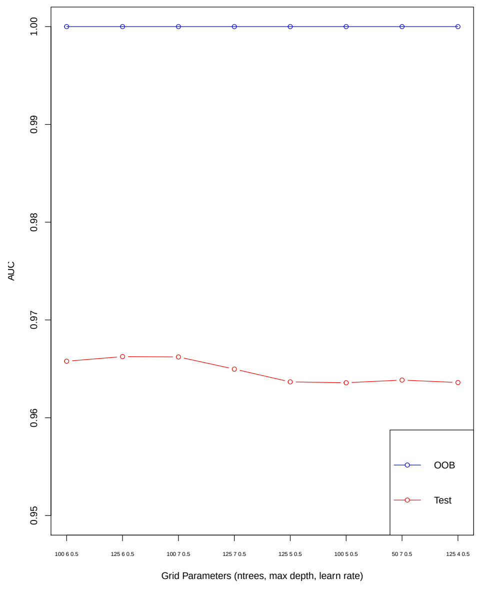


Figure 27: Boosted tree grid search results

The plot below shows the variable importance of the top 10 variables, it came back with A.52 being the most significant, which is different from the random forest, showing the difference between the two models.

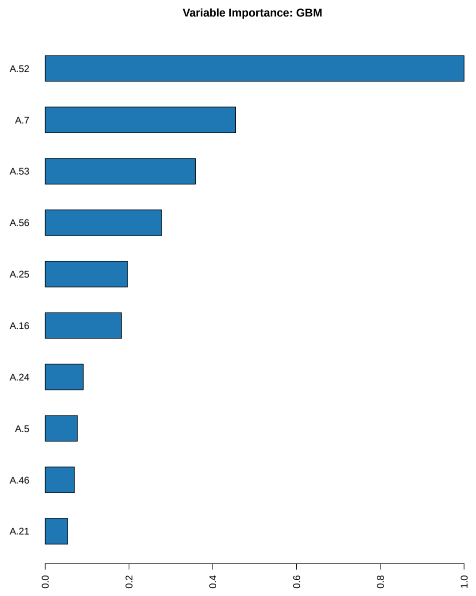
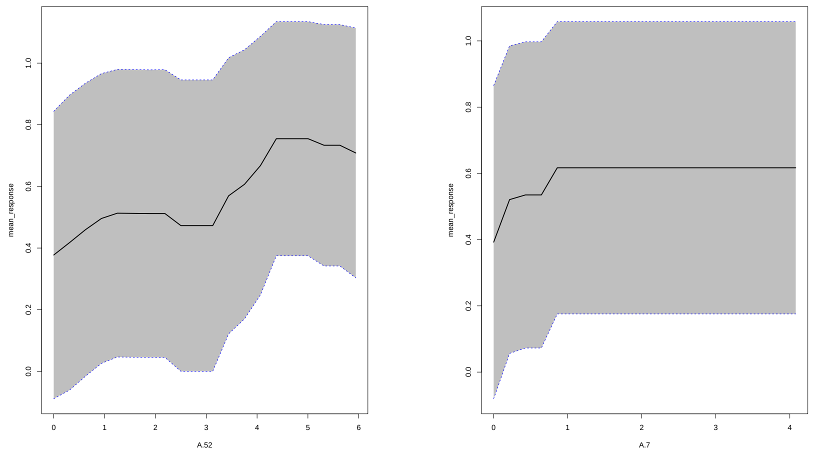


Figure 28: Boosted tree variable importance

A.52 has a much bigger change in mean response over a greater range, whilst A.7 has a smaller change only between 0 and 1. The boosted trees are much more reactive to changes in A.52.



(a) A.52

(b) A.7

Figure 29: Partial dependence plots

The final model performed extremely well on the test set, also having a training and test AUC close together indicating the model has not over or under fit.

AUC on training set	0.9999997
AUC on test set	0.9894595

2.7 Comparison

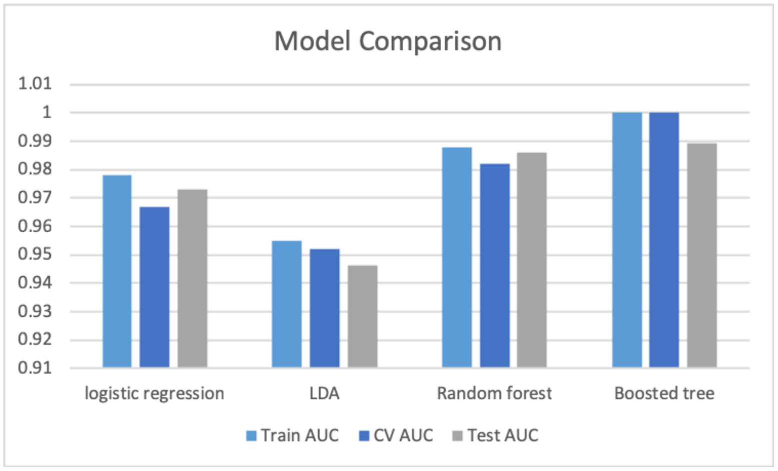


Figure 30: Model Comparisons

	Misclassification %	Precision %	Recall %
Logistic Regression	6.5	90.3	91.1
LDA	10.8	78.5	89.6
Random Forest	4.0	91.1	94.4
Boosted Tree	2.8	95.7	94.7

The best model came back as the boosted trees model as it outperformed all the other models. This is probably due to the fact that is continues to try explain the unexplained variance as it learns. LDA did the worst, this is likely due to the fact that the assumptions for the model didn't hold.
All the models except for LDA have pretty even precision and recall values and therefore good at predicting both spam and email evenly. LDA's recall was about 10 % higher meaning it was better at predicting emails.
The random forest and boosted trees model had similar train, out of bag and test AUC values, pointing towards the fact the models will generalize well.

3 Appendix:

3.1 R Code: Question 1 a and b

```
rm(list=ls())
set.seed(108)

## reading in data

fulldata <- read.csv("my_boston.csv")

## splitting into train and test set

split.function <- function(data)
{
  rownames(data) <- NULL
  sampled <- sample(seq(0,400,1), 400*0.8)
  train <- data[sampled,]
  test <- data[-sampled,]
  return(list(train = train, test = test))
}

split.data <- split.function(fulldata)

## fitting full linear model

linear_model <- lm(medv ~ ., data =(split.data$train[,-1]))
summary(linear_model)

linear_predictions <- predict.lm(linear_model, split.data$
    train[,-1])
mean((linear_predictions - split.data$train[,14])^2)

## cross validation for initial model

k=10
set.seed(2)
folds=sample(1:k,nrow(split.data$train),replace=TRUE)
cv.errors=matrix(NA,k,1)

for(j in 1:k){
  fit = lm(medv~.,data=split.data$train[,-1][folds!=j,])
  pred = predict.lm(fit,split.data$train[,-c(1,14)][folds==j
      ,],id=i)
  cv.errors[j]=mean( (split.data$train[,14][folds==j]-pred)
      ^2)
}

mean.cv.errors=apply(cv.errors,2,mean)
```

```
print(mean.cv.errors)

## log transforms of variables

split.data$train$lstat <- log(split.data$train$lstat)
split.data$train$indus <- log(split.data$train$indus)
split.data$train$crim <- log(split.data$train$crim)

split.data$test$lstat <- log(split.data$test$lstat)
split.data$test$indus <- log(split.data$test$indus)
split.data$test$crim <- log(split.data$test$crim)

## scaling inputs

test_normalised <- as.data.frame(apply(split.data$test[,-c
    (1,14)],2, scale))
train_normalised <- as.data.frame(apply(split.data$train[,-c
    (1,14)],2, scale))

train_normalised$medv <- split.data$train[,14]
y.two <- split.data$test[,14]

## adding in interaction terms

test_normalised$interaction1 <- test_normalised$dis*test_
    normalised$rad
train_normalised$interaction1 <- train_normalised$dis*train_
    normalised$rad
test_normalised$interaction2 <- test_normalised$tax*test_
    normalised$lstat
train_normalised$interaction2 <- train_normalised$tax*train_
    normalised$lstat
test_normalised$interaction3 <- test_normalised$ptratio*test
    _normalised$lstat
train_normalised$interaction3 <- train_normalised$ptratio*
    train_normalised$lstat
test_normalised$interaction4 <- test_normalised$nox*test_
    normalised$rad
train_normalised$interaction4 <- train_normalised$nox*train_
    normalised$rad
test_normalised$interaction5 <- test_normalised$crim*test_
    normalised$lstat
train_normalised$interaction5 <- train_normalised$crim*train
    _normalised$lstat
test_normalised$interaction5 <- test_normalised$crim*test_
    normalised$chas
train_normalised$interaction5 <- train_normalised$crim*train
    _normalised$chas

## exhaustive subset selection with cross validation
```

```
library(leaps)

predict.regsubsets <- function(object, newdata, id,...) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[,xvars]%*%coefi
}

y <- train_normalised$medv

k=10
set.seed(1)
folds=sample(1:k,nrow(train_normalised),replace=TRUE)
cv.errors=matrix(NA,k,17, dimnames=list(NULL, paste(1:17)))

for(j in 1:k){
  best.fit=regsubsets(medv~.,data=train_normalised[folds!=j
      ,],nvmax=17)
  for(i in 1:17){
    pred=predict.regsubsets(best.fit,train_normalised[folds
        ==j,],id=i)
    cv.errors[j,i]=mean( (y[folds==j]-pred)^2)
  }
}

mean.cv.errors=apply(cv.errors,2,mean)
print(mean.cv.errors)

par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
lines(res.sum$rss, type = 'b', col = 'red')


reg.best=regsubsets(medv~.,data= train_normalised, nvmax=
    17)
coef(reg.best,11)

exhaustive = mean.cv.errors
exh.coef = coef(reg.best,11)

## forward selection

k=10
set.seed(1)
folds=sample(1:k,nrow(train_normalised),replace=TRUE)
cv.errors=matrix(NA,k,17, dimnames=list(NULL, paste(1:17)))
```

```r
for(j in 1:k){
  best.fit=regsubsets(medv~.,data=train_normalised[folds!=j
      ,],nvmax=17, method = "forward")
  for(i in 1:17){
    pred=predict.regsubsets(best.fit,train_normalised[folds
        ==j,],id=i)
    cv.errors[j,i]=mean( (y[folds==j]-pred)^2)
  }
}

mean.cv.errors=apply(cv.errors,2,mean)
print(mean.cv.errors)

par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
lines(res.sum$rss, type = 'b', col = 'red')


reg.best=regsubsets(medv~.,data= train_normalised, nvmax=
    17)
coef(reg.best,13)

forward = mean.cv.errors
for.coeff = coef(reg.best,13)

## backward elimination

k=10
set.seed(1)
folds=sample(1:k,nrow(train_normalised),replace=TRUE)
cv.errors=matrix(NA,k,17, dimnames=list(NULL, paste(1:17)))

for(j in 1:k){
  best.fit=regsubsets(medv~.,data=train_normalised[folds!=j
      ,],nvmax=17, method = "backward")
  for(i in 1:17){
    pred=predict.regsubsets(best.fit,train_normalised[folds
        ==j,],id=i)
    cv.errors[j,i]=mean( (y[folds==j]-pred)^2)
  }
}

mean.cv.errors=apply(cv.errors,2,mean)
print(mean.cv.errors)

par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
lines(res.sum$rss, type = 'b', col = 'red')
```

```
reg.best=regsubsets(medv~.,data= train_normalised, nvmax=
    17)
coef(reg.best,13)

backward = mean.cv.errors
back.coeff = coef(reg.best,13)

## combination of forward and backward

k=10
set.seed(1)
folds=sample(1:k,nrow(train_normalised),replace=TRUE)
cv.errors=matrix(NA,k,17, dimnames=list(NULL, paste(1:17)))

for(j in 1:k){
  best.fit=regsubsets(medv~.,data=train_normalised[folds!=j
      ,],nvmax=17, method = "seqrep")
  for(i in 1:17){
    pred=predict.regsubsets(best.fit,train_normalised[folds
        ==j,],id=i)
    cv.errors[j,i]=mean( (y[folds==j]-pred)^2)
  }
}

mean.cv.errors=apply(cv.errors,2,mean)
print(mean.cv.errors)

par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
lines(res.sum$rss, type = 'b', col = 'red')


reg.best=regsubsets(medv~.,data= train_normalised, nvmax=
    17)
coef(reg.best,16)

both = mean.cv.errors
both.coeff = coef(reg.best,16)

## combined plot

par(mfrow = c(2,2))
plot(exhaustive,type='b', main = 'Exhaustive', ylab = 'CV␣
    MSE', xlab = 'No.␣of␣variables', col = 'red')
plot(forward,type='b', main = 'Forward', ylab = 'CV␣MSE',
    xlab = 'No.␣of␣variables', col = 'blue')
plot(backward,type='b', main = 'Backward', ylab = 'CV␣MSE',
    xlab = 'No.␣of␣variables', col = 'darkgreen')
plot(both,type='b', main = 'Forward␣&␣Backward', ylab = 'CV␣
    MSE', xlab = 'No.␣of␣variables', col = 'orange')
```

```
## lasso regression

library(glmnet)

par(mfrow = c(1,2))

y <- train_normalised$medv
x <- as.matrix(train_normalised[ ,-13])
y.two <- test_normalised$medv
x_test <- test_normalised[ ,-13]
train = train_normalised

lasso.mod = glmnet(x, y, alpha = 1)
plot(lasso.mod)

cv.out = cv.glmnet(x, y, alpha = 1)
plot(cv.out)
bestlam = cv.out$lambda.min
print(cv.out)
print(cv.out$cvm[78])
lasso.pred = predict(lasso.mod, s = bestlam, newx = as.
    matrix(test_normalised))

out = glmnet(x, y, alpha = 1)
lasso.coef = predict(out, type = "coefficients", s = bestlam
    )

print(lasso.coef)
print(lasso.coef[lasso.coef != 0])

print(mean((lasso.pred-split.data$test[,14])^2))

plot( (lasso.pred - split.data$test[,14]), ylab= 'Test␣
    residuals', col = 'blue')

## fitting final model

final_model <- lm(medv ~ . - chas - crim-age-interaction1-
    interaction3-interaction5, data =train_normalised)
summary(final_model)
y = split.data$test[,14]
linear_predictions <- predict.lm(final_model, test_
    normalised)
mean((linear_predictions - y)^2)

## residual plots of final model

par(mfrow = c(1,1))
plot( (linear_predictions - split.data$test[,14]), ylab= '
    Test␣residuals', col = 'blue', ylim = c(-20,20))
lines( (yhat- boston.test), ylab= 'Test␣residuals', col = '
    red', type = 'p')
legend("topright", c('linear␣regression', 'regression␣tree')
    , col = c('blue', 'red'), pch = c(1,1))

res = rstandard(final_model)
qqnorm(res, main = "")
qqline(res)
```

3.2 R Code: Question 1 c

```
library(tree)

set.seed(108)

par(mfrow = c(1,1))

rm(list=ls())

## reading in data

fulldata <- read.csv("my_boston.csv")

## splitting into test/train set

split.function <- function(data)
{
  rownames(data) <- NULL
  sampled <- sample(seq(0,400,1), 400*0.8)
  train <- data[sampled,]
  test <- data[-sampled,]
  return(list(train = sampled, test = test))
}

split.data = split.function(fulldata)

## fitting initial tree with relaxed stopping criteria

tree_model <- tree(medv ~ . , fulldata[,-1], subset = split.
    data$train, control=tree.control(length(split.data$train)
    ,minsize = 20, mincut = 10, mindev = 0.0001))
yhat = predict(tree_model, newdata = fulldata[-split.data$
    train,-1])
boston.test = fulldata[-split.data$train, "medv"]
print(mean((yhat-boston.test)^2))
print(tree_model)
summary(tree_model)

## Initial prediction
```

```r
yhat = predict(tree_model, newdata = fulldata[split.data$
    train,-1])
boston.test = fulldata[split.data$train, "medv"]
plot(yhat,boston.test)
print(mean((yhat-boston.test)^2))

plot(tree_model)
text(tree_model, pretty = 0)

plot(prune.tree(tree_model))

## alphas and the deviance associated for the subtrees

alphas = prune.tree(tree_model)$k
print(alphas)

cv.boston = cv.tree(tree_model, K =10)
print(cv.boston)

plot(cv.boston$size, cv.boston$dev, type = 'b', col = 'blue'
    , ylim = c(2000,17000))
lines(prune.tree(tree_model)$size, prune.tree(tree_model)$
    dev, col = "red", type = 'b')
legend("topright", inset = .05, c("cv", "full␣dataset"),
    fill = c("blue","red"))

## pruning the tree

final.boston = prune.tree(tree_model, best = 6)

plot(final.boston)
text(final.boston, pretty = 0)

## final predictions test set

yhat = predict(final.boston, newdata = fulldata[-split.data$
    train,-1])
boston.test = fulldata[-split.data$train, "medv"]
plot(yhat,boston.test)
print(mean((yhat-boston.test)^2))

## final predicitions train set

yhat = predict(final.boston, newdata = fulldata[split.data$
    train,-1])
boston.test = fulldata[split.data$train, "medv"]
plot(yhat,boston.test)
print(mean((yhat-boston.test)^2))
```

```
## residual plots

par(mfrow = c(1,2))
plot( (yhat- boston.test), ylab= 'Test␣residuals', col = '
    blue')

plot(yhat ~ boston.test, ylab = "predictions", xlab= "test␣
    data", col = "red")
abline(lm(yhat ~ boston.test))
```

3.3 R Code: Question 2a

```
rm(list=ls())
library(aod)
library(ggplot2)
library(MASS)

# reading in data
spamdata <- read.csv("spam_data.csv")
spamdata$spam = factor(spamdata$spam)

plot <- list()

## plotting boxplots

box_variables <- colnames(spamdata[,2:9])
for(i in box_variables) {
  plot[[i]] <- ggplot(spamdata, aes_string(x = "spam", y = i
      , col = "spam", fill = "spam")) +
    geom_boxplot(alpha = 0.2) +
    theme(legend.position = "none") +
    scale_color_manual(values = c("blue", "red"))
  scale_fill_manual(values = c("blue", "red"))
}

do.call(grid.arrange, c(plot, nrow = 2))

## plotting stacked histogram

library(ggplot2)
plot4 =   ggplot(spamdata, aes(x = A.4, fill = spam)) +
  geom_histogram(binwidth = 5, position = 'stack')
print(plot4)

## classification rate based on balance

print(nrow(spamdata[spamdata$spam == "spam",]))
print(nrow(spamdata[spamdata$spam == "email",]))

print(2141/(1460+2141))
```

```
## correlation plot

numeric_mydata <- spamdata[,c(26:42)]
numeric_spam = as.numeric(spamdata$spam)- 1
numeric_mydata = cbind(numeric_mydata, numeric_spam)
#str(numeric_mydata)
library(corrplot)
M <- cor(numeric_mydata)
corrplot(M, method="circle")

par(mfrow = c(1,1))
plot(spamdata[,33], col = 'red', pch = 3, ylab = 'Value')
lines(spamdata[,35], col = 'blue', type = 'p')
legend('topleft', c('A.32', 'A.34'), pch = c(3,1), col = c('
    red', 'blue'))

## fitting full logistic model

logistic.model = glm(spam~ . ,data = spamdata[,-1], family =
     binomial)
summary(logistic.model)

## predicting on training set

glm.probs = predict(logistic.model, type = "response")
glm.probs[1:10]

glm.pred = rep('email', length(spamdata$spam))
glm.pred[glm.probs>0.5] = 'spam'

table(glm.pred, spamdata$spam)
mean(glm.pred == spamdata$spam)

## plotting probabilities of the classses

predicted.data = data.frame(prob = glm.probs, actual =
    spamdata$spam)
predicted.data = predicted.data[order(predicted.data$prob,
    decreasing = FALSE),]
predicted.data$rank = 1:nrow(predicted.data)
library(ggplot2)
library(cowplot)

plot = ggplot(data = predicted.data, aes(x = rank, y = prob)
    ) +
  geom_point(aes(color = actual),alpha = 1,shape = 1,stroke
     = 3)+
  xlab("Index")+
  ylab("Predicted␣probability␣of␣being␣spam")+
  scale_color_manual(values=c("blue", "green"))+
  geom_hline(yintercept=0.5, linetype="dashed", color = "red
     ")

print(plot)

## plotting ROC curve

library(ROCR)
pred1 <- prediction(glm.probs, spamdata$spam)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1, col= 'red')
auc = performance(pred1, measure = "auc")@y.values[[1]]
print(auc)

## stepwise selection processes

null.model =glm(spam~ 1 ,data = spamdata[,-1], family =
    binomial)
step = stepAIC(null.model, trace = FALSE, direction = "both"
    , scope = formula(logistic.model))

forward = stepAIC(null.model, direction = "forward", scope =
     formula(logistic.model), trace = FALSE)

backward = stepAIC(logistic.model, direction = "backward",
    trace = FALSE)

formula(backward)

## lasso regression

library(glmnet)

xfactors <- model.matrix(spam ~ ., data = spamdata)[, -1]
x        <- as.matrix(spamdata[ , - length(spamdata)])[ ,-1]

# Note alpha=1 for lasso only and can blend with ridge
    penalty down to
# alpha=0 ridge only.
glmmod <- glmnet(x, y=spamdata$spam, alpha=1, family="
    binomial")

# Plot variable coefficients vs. shrinkage parameter lambda.
plot(glmmod, xvar="lambda")

cv.glmmod <- cv.glmnet(x, y=spamdata$spam, alpha=1, family =
     "binomial", type.measure = 'auc')
plot(cv.glmmod)
```

```
## cross validation on initial model

df                     <- spamdata[,-c(1,length(spamdata))]
df$Y                   <- spamdata$spam


df      <- df[sample(nrow(spamdata)),]
folds   <- cut(seq(1,nrow(spamdata)),breaks=10,labels=FALSE)
result  <- list()

cv.actual = rep(NA,10)

for(i in 1:10){
  testIndexes <- which(folds==i,arr.ind=TRUE)
  testData    <- df[testIndexes, ]
  trainData   <- df[-testIndexes, ]
  model       <- glm(Y~ . ,family=binomial,data=trainData)
  result[[i]] <- predict(model, testData, type = "response")
  actual = spamdata[as.numeric(names(result[[i]])),length(
      spamdata)]
  actual = ifelse(actual == 'spam', 1, 0)
  pred_cv <- prediction(result[[i]], actual)
  cv.actual[i] = performance(pred_cv, measure = "auc")@y.
      values[[1]]
}
print(mean(cv.actual))

## cross validation on final model

df                     <- spamdata[,-c(1,length(spamdata))]
df$spam                   <- spamdata$spam


df      <- df[sample(nrow(spamdata)),]
folds   <- cut(seq(1,nrow(spamdata)),breaks=10,labels=FALSE)
result  <- list()

cv.actual = rep(NA,10)

for(i in 1:10){
  testIndexes <- which(folds==i,arr.ind=TRUE)
  testData    <- df[testIndexes, ]
  trainData   <- df[-testIndexes, ]
  model       <- glm(formula(backward) ,family=binomial,data
      =trainData)
  result[[i]] <- predict(model, testData, type = "response")
  actual = spamdata[as.numeric(names(result[[i]])),length(
      spamdata)]
  actual = ifelse(actual == 'spam', 1, 0)
  pred_cv <- prediction(result[[i]], actual)
  cv.actual[i] = performance(pred_cv, measure = "auc")@y.
      values[[1]]
}
print(mean(cv.actual))

## in sample ROC curve of final model

model        <- glm(formula(backward) ,family=binomial,data=
    spamdata[,-1])
result <- predict(model, spamdata[, -c(1, length(spamdata))
    ], type = "response")
actual = spamdata[,length(spamdata)]
pred_cv <- prediction(result, actual)
auc = performance(pred_cv, measure = "auc")@y.values[[1]]
print(auc)

## loading in test set

spamtest <- read.csv("spam_test.csv")
#summary(spamdata)
spamtest$spam = factor(spamtest$spam)

## using final model to predict

final.model = glm(formula(backward) ,family=binomial,data=
    spamdata[,-1])

glm.probs = predict(final.model, spamtest[,-c(1,length(
    spamtest))], type = "response")

glm.pred = rep('email', length(spamtest$spam))
glm.pred[glm.probs>0.5] = 'spam'
table(glm.pred, spamtest$spam)

pred1 <- prediction(glm.probs, spamtest$spam)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1, col= 'red')
auc = performance(pred1, measure = "auc")@y.values[[1]]
print(auc)
```

3.4 R Code: Question 2b

```
library(MASS)

## loading in data
spamdata <- read.csv("spam_data.csv")
#summary(spamdata)
spamdata$spam = factor(spamdata$spam)

## fitting lda on all variables
```

```r
lda.fit = lda(spam ~ . , data = spamdata[,-1])
lda.fit

## cross validation of initial model

df  <- spamdata[,-c(1,length(spamdata))]
df$spam <- ifelse(spamdata$spam == 'spam', 2,1)


df     <- df[sample(nrow(spamdata)),]
folds  <- cut(seq(1,nrow(spamdata)),breaks=10,labels=FALSE)
result <- list()

cv.actual = rep(NA,10)

dim(df)

for(i in 1:10){
  testIndexes <- which(folds==i,arr.ind=TRUE)
  testData    <- df[testIndexes, ]
  trainData   <- df[-testIndexes, ]
  model       <- lda(both$formula , data = trainData)
  result[[i]] <- predict(model, testData[,-length(testData)
      ])
  pred_cv <- prediction(result[[i]]$posterior[,2], testData$
      spam)
  cv.actual[i] = performance(pred_cv, measure = "auc")@y.
      values[[1]]
}
print(mean(cv.actual))

## stepwise selection methods

library(klaR)

both = stepclass(spam ~ ., data = spamdata[,-1], method = "
    lda", start.vars = "A.1", maxvar = 57)
backward = stepclass(spam ~ ., data = spamdata[,-1], method
    = "lda", direction = "backward")
forward = stepclass(spam ~ ., data = spamdata[,-1], method =
     "lda", direction = "forward", start.vars = "A.1", maxvar
     = 58)

## using final model to predict test set

lda.pred.test = predict(lda.fit, spamtest[,-c(1, length(
    spamdata))])
lda.pred.test
```

```
par(mfrow = c(1,1))

lda = prediction(lda.pred.test$posterior[,2], spamtest$spam)
perf = performance(lda,measure = "tpr", x.measure = "fpr")
plot(perf, col = "red")

auc = performance(lda, measure = "auc")@y.values
print(auc)

## lasso for LDA

library(penalizedLDA)

y = ifelse(spamdata$spam == "spam",2,1)

lambdas = exp(seq(-8,-3,length = 20))

cv.out <- PenalizedLDA.cv(spamdata[,-c(1,length(spamdata))],
    y,lambdas=lambdas, nfold = 10)
print(cv.out)

par(mfrow = c(1,1))
plot(cv.out$errs/(length(spamdata)/10) ~ log(cv.out$lambdas)
    , col = "blue")

print(cv.out$errs/(length(spamdata)/10))

folds = cv.out$folds

print(folds)

df  <- spamdata[,-c(1,length(spamdata))]
df$y <- ifelse(spamdata$spam == 'spam', 2,1)


df     <- df[sample(nrow(spamdata)),-c(1, length(spamdata))]
folds  <- cut(seq(1,nrow(spamdata)),breaks=10,labels=FALSE)
result <- list()

cv.actual = rep(NA,10)

dim(df)

for(i in 1:10){
  testIndexes <- which(folds==i,arr.ind=TRUE)
  testData    <- df[testIndexes, ]
  trainData   <- df[-testIndexes, ]
  model       <- PenalizedLDA(trainData[,-length(trainData)
      ], trainData$y, xte=NULL, type = "standard", lambda =
      cv.out$bestlambda, K = 1, chrom =
                                NULL, lambda2 = NULL,
                                    standardized = FALSE,
                                    wcsd.x = NULL, ymat =
                                    NULL,
                              maxiter = 20, trace=FALSE)
  result[[i]] <- predict(model, testData[,-length(testData)
      ], type = "response")
  pred_cv <- prediction(result[[i]], testData$y)
  cv.actual[i] = performance(pred_cv, measure = "auc")@y.
      values[[1]]
}
print(mean(cv.actual))

par(mfrow = c(1,1))

lda = prediction(lda.pred$posterior[,2], spamdata$spam)
perf = performance(lda,measure = "tpr", x.measure = "fpr")

auc = performance(lda, measure = "auc")@y.values
print(auc)

## comparing covariance matrices

CV1 = cov(XX[spamdata$spam == "spam",])
CV2 = cov(XX[spamdata$spam == "email",])

rownames(CV1) <- as.character(seq(1, 54, 1))
colnames(CV1) <- as.character(seq(1, 54, 1))
library(reshape2)
library(ggplot2)
ggplot(data = melt(t((CV1))), aes(x=Var1, y=Var2)) +
  geom_tile(aes(fill = value)) +
  scale_fill_gradientn(colours = c("red", "orange", "yellow"
      , "green", "cyan", "blue")) +
  xlab("N") + ylab("Q")

## assumption tests

library(car)

column.names = colnames(spamdata[, -c(1,length(spamdata))])
p.sum = 0

## test for equality variance covariance

for (i in column.names)
{

  formula.val = formula(paste(paste(i,'~␣'),i))
  print(formula.val)
  #test = bartlett.test(formula.val, data=spamdata)
  test = leveneTest(formula.val, data = spamdata)
  p.sum = p.sum + test$‘Pr(>F)‘[1]
  #p.sum = p.sum + test$p.value
}

avg.p = p.sum/length(column.names)
print(avg.p)


## multivariate normality

x <- as.matrix(spamdata[,-c(1,length(spamdata))]) # n x p
   numeric matrix
center <- colMeans(x) # centroid
n <- nrow(x); p <- ncol(x); cov <- cov(x);
d <- mahalanobis(x,center,cov) # distances
qqplot(qchisq(ppoints(n),df=p),d,
       main="QQ␣Plot␣Assessing␣Multivariate␣Normality",
       ylab="Mahalanobis␣D2", ylim = c(0,100))
abline(a=0,b=1)
```

3.5 R Code: Question 2 c and d

```
library(h2o)
library(ggplot2)
library(reshape2)

## initializing h2o instance

localH2O = h2o.init(ip = "localhost", port = 54321, startH2O
     = TRUE,min_mem_size = "1g")

## reading in data

spamdata <- read.csv("spam_data.csv")
#summary(spamdata)
spamdata$spam = factor(spamdata$spam)

## creating validation set

split.function <- function(data)
{
  rownames(data) <- NULL
  sampled <- sample(seq(0,length(data[,1]),1), length(data
      [,1])*0.8)
  train <- data[sampled,]
  test <- data[-sampled,]
  return(list(train = train, test = test))
}
```

```
split.data <- split.function(spamdata)

## changing into h2o variables

spam_h2o = as.h2o(split.data$train)
test_h2o = as.h2o(split.data$test)

## initializing variables for grid search

mtries <- c(2,3,4)
ntrees = c(25,50,75,100,125)
max_depth = c(4,5,6,7,8,9)

hyper_params <- list(mtries = mtries, ntrees = ntrees, max_
    depth = max_depth)

## running grid search

rf1_grid <- h2o.grid(algorithm = "randomForest",
                     hyper_params = hyper_params,
                     x = 2:58, y = 59,
                     training_frame = spam_h2o,
                     #validation_frame = datTest_h2o,
                     nfolds = 0,
                     seed = 1)

summary(rf1_grid)

## choosing best model

model1 = h2o.getModel(rf1_grid@model_ids[[1]])
h2o.auc(model1)

## plotting grid search results

models=list()
for(i in 1:8){
  models[i] = h2o.getModel(rf1_grid@model_ids[[i]])
}

# OOB
xlabel = NULL
for(i in 1:8){
  xlabel[i] = paste(models[[i]]@allparameters$ntrees, models
      [[i]]@allparameters$mtries, models[[i]]@allparameters$
      max_depth)
}

res_devOOB = NULL
```

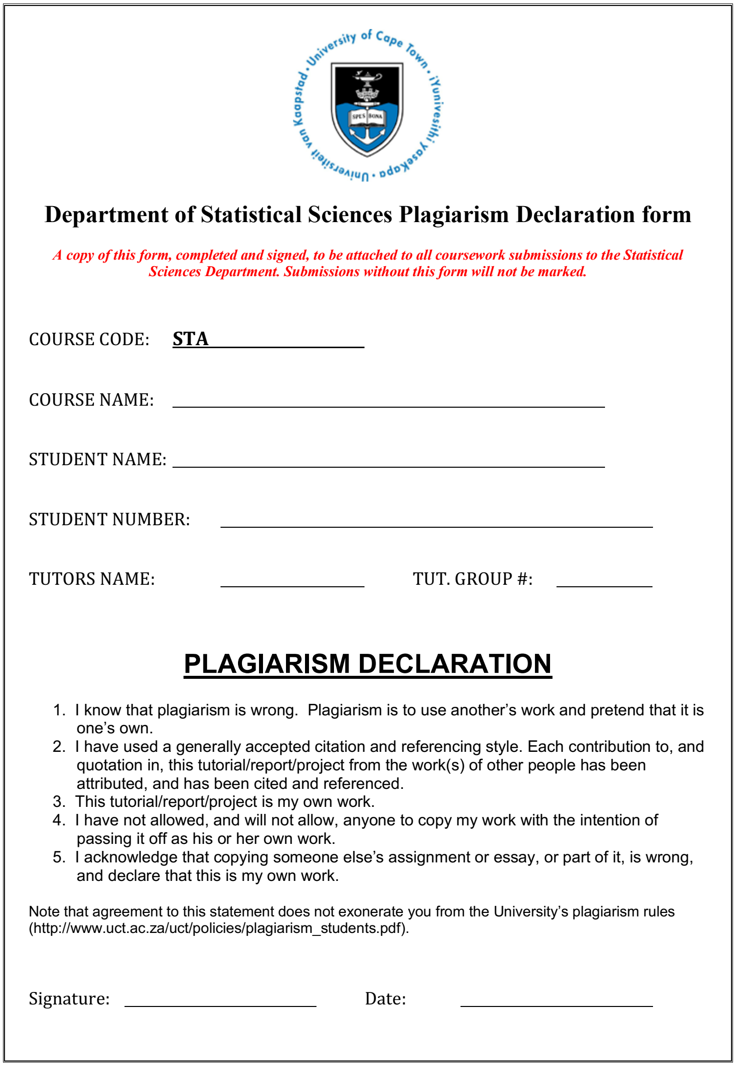
```
for(i in 1:8){
  res_devOOB[i] = h2o.auc(models[[i]])
}

# TEST SET
res_devTest = NULL
for (i in 1:8) {
  res_devTest[i] <- h2o.auc(h2o.performance(h2o.getModel(rf1
      _grid@model_ids[[i]]), newdata = test_h2o))
}

par(mfrow = c(1,1))

plot(res_devTest[1:8], xaxt = "n", xlab='Grid␣Parameters␣(
    ntrees,␣mtries,␣max_depth)', type = "b",col ="red",
      ylim = c(0.95,1), ylab = 'AUC')
axis(1, at=1:8, labels=xlabel[1:8], cex.axis = 0.8)
lines(res_devOOB[1:8], type = "o", col ="blue")
legend("bottomright",c("OOB","Test"),pch = 21, pt.bg = "
    white", lty = 1, col = c("blue", "red"))


best_rf <- h2o.getModel(rf1_grid@model_ids[[1]])
perf_rf <- h2o.performance(best_rf, newdata = spam_h2o)
auc.val <- h2o.auc(best_rf)
perf_rf
auc.val

## variable importance plot

h2o.varimp_plot(best_rf)

predictionsTrain <- h2o.predict(best_rf, spam_h2o)
predictionsTrain

## predictions

yhatTrain = (as.matrix(predictionsTrain$spam))
#yhatTrain = as.factor(as.matrix(predictionsTrain$pr))

predictionsTest <- h2o.predict(best_rf, test_h2o)
predictionsTest

yhatTest = (as.matrix(predictionsTest$spam))

library(ROCR)
pred1 <- prediction(yhatTrain, split.data$train$spam)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1, col= 'red')
auc = performance(pred1, measure = "auc")@y.values[[1]]
```

```
print(auc)


pred1 <- prediction(yhatTest, split.data$test$spam)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1, col= 'red')
auc = performance(pred1, measure = "auc")@y.values[[1]]
print(auc)

## partial dependence plots

h2o.partialPlot(object = best_rf, data = test_h2o, cols = c(
    "A.7", "A.52"))

table(ifelse(yhatTest >0.5, "spam", "email"), split.data$
    test$spam)


## boosted trees

ntrees = c(50, 100, 125)
max_depth = c(4,5,6,7)
learn_rate = c(0.001,0.01, 0.1, 0.5)
learn_rate_annealing = 0.99

hyper_params <- list(ntrees = ntrees, max_depth = max_depth,
     learn_rate = learn_rate)

## running a grid search

gbm_grid <- h2o.grid(algorithm = "gbm",
                     hyper_params = hyper_params,
                     x = 2:58, y = 59,
                     training_frame = spam_h2o,
                     #validation_frame = datTest_h2o,
                     nfolds = 0,
                     seed = 1)
summary(gbm_grid)

model1 = h2o.getModel(gbm_grid@model_ids[[1]])
h2o.auc(model1)

## plotting grid search results

models=list()
for(i in 1:8){
  models[i] = h2o.getModel(gbm_grid@model_ids[[i]])
}

# OOB
```

```
xlabel = NULL
for(i in 1:8){
  xlabel[i] = paste(models[[i]]@allparameters$ntrees, models
      [[i]]@allparameters$max_depth, models[[i]]
      @allparameters$learn_rate)
}

res_devOOB = NULL
for(i in 1:8){
  res_devOOB[i] = h2o.auc(models[[i]])
}

# TEST SET
res_devTest = NULL
for (i in 1:8) {
  res_devTest[i] <- h2o.auc(h2o.performance(h2o.getModel(rf1
      _grid@model_ids[[i]]), newdata = test_h2o))
}

par(mfrow = c(1,1))

plot(res_devTest[1:8], xaxt = "n", xlab='Grid␣Parameters␣(
    ntrees,␣max␣depth,␣learn␣rate)', type = "b",col ="red",
     ylim = c(0.95,1), ylab = 'AUC')
axis(1, at=1:8, labels=xlabel[1:8], cex.axis = 0.6)
lines(res_devOOB[1:8], type = "o", col ="blue")
legend("bottomright",c("OOB","Test"),pch = 21, pt.bg = "
    white", lty = 1, col = c("blue", "red"))


best_gbm <- h2o.getModel(gbm_grid@model_ids[[1]])
perf_gbm <- h2o.performance(best_gbm, newdata = datTest_h2o)

## variable importance plot

h2o.varimp_plot(best_gbm)

## predictions

predictionsTrain <- h2o.predict(best_gbm, spam_h2o)
predictionsTrain

yhatTrain = (as.matrix(predictionsTrain$spam))

predictionsTest <- h2o.predict(best_gbm, test_h2o)
predictionsTest

yhatTest = (as.matrix(predictionsTest$spam))

library(ROCR)
```

```
pred1 <- prediction(yhatTrain, split.data$train$spam)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1, col= 'red')
auc = performance(pred1, measure = "auc")@y.values[[1]]
print(auc)


pred1 <- prediction(yhatTest, split.data$test$spam)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1, col= 'red')
auc = performance(pred1, measure = "auc")@y.values[[1]]
print(auc)

## partial dependance plots

h2o.partialPlot(object = best_gbm, data = test_h2o, cols = c
    ("A.52", "A.7"))

table(ifelse(yhatTest >0.5, "spam", "email"), split.data$
    test$spam)
```

3.6 Plagiarism Declaration



Department of Statistical Sciences Plagiarism Declaration form

A copy of this form, completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department. Submissions without this form will not be marked.

COURSE CODE:	**STA** ______________		
COURSE NAME:	______________________________		
STUDENT NAME:	______________________________		
STUDENT NUMBER:	______________________________		
TUTORS NAME:	______________	TUT. GROUP #:	__________

PLAGIARISM DECLARATION

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed, and has been cited and referenced.
3. This tutorial/report/project is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
5. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Note that agreement to this statement does not exonerate you from the University's plagiarism rules (http://www.uct.ac.za/uct/policies/plagiarism_students.pdf).

Signature: ______________________ Date: ______________________