Predict 454: Advanced Machine Learning

Assignment 3

Predictive Modeling For Binary Classification

Jack Doig

Introduction and Problem Statement

This paper outlines the results from an exercise in exploratory data analysis (EDA), statistical graphics, and predictive modeling exercise for a Email Spam data set. The goal of the study is to classify emails as "Spam" or otherwise.

The Data originate from a 1998 study at AT&T Labs-Research, at a time when "Spam" was an emerging problem in email communication. The researchers responsible for the data are Lorrie Faith Cranor and Brian A. LaMacchia.

"Spam" refers to unsolicited email, and often contains advertising, sexual content, requests for funding, or dubious financial opportunities. These emails become annoying and time consuming to filter through as a recipient's email becomes increasingly known by "Spam" propagators.

This paper firstly does a simple data quality check, followed by basic EDA, and then some more advanced model-driven EDA. The goal at this stage is to first find interesting relationships and fully understand the structure of the data with a view to optimizing decisions during modeling.

The performance metric for the problem is defined, and the study then moves into predictive modeling in order to choose the best model for this classification problem.

Part 1: Data Quality Check

1.1 Basic Data understanding

The table below describes the structure of the data set.

Metric	Value
Number of observations	4601
Type of Target Variable	Categorical
Categorical Independent Variables	0
Continuous Independent Variables	57
Total Number of Variables	58
Size of Data File	709.2kB

Figure 1.1: basic data set metrics

The table above shows that this data set is made up of exclusively continuous variables, with the exception of the target variable. It is also a relatively small data set, with only 4601 observations, so there should be no computational challenges in this study, however this may have a downstream impact on the decision of validation and training set size during modeling.

The table below describes each of the variables. It is important to invest time in gaining a real-world understanding of the data so as to understand the impact, or credibility of, of outliers, patterns and distributions, and possibly get clues so as to optimize the EDA, modeling process or generate ideas about variable interaction.

The target variable is "Spam", which is a categorical variable, representing whether an email meets the definition of Spam as outlined above.

Variable	Type	Number of variables in category	Description
Spam	Categorical	1	The Target Variable, indicates whether or not the email is designated as spam or not
word_freq_*	Continuous	48	48 key words have been counted in the email, and these variable represent the percentage of those words against the total number of words. Examples are “credit”, “Technology”, and “meeting”
char_freq_*	Continuous	6	6 characters has been counted in each email, and these variables represent the precentages of those characters againsts the total number of characters in each email. Examples are “!”, “(“, “[“ and “#”
capital_run_length_average	Continuous	1	The average length of the contiguous capital letter sequences in the email
capital_run_length_longest	Continuous	1	The longest run of capital letters in the email
capital_run_length_total	Continuous	1	The total length of capital letter sequences in the email

Figure 1.2: basic description of variables

1.2 Basic Descriptive Statistics

The table below outlines basic statistics for a sample of notable independent variables. This table highlights the extreme descriptive statistics in the data set, so as to understand the boundaries of the data.

	char_freq_ exclamationMark	word_freq_3d	word_freq_you	char_freq_ openSquareBracket	capital_run_ length_average	capital_run_ length_longest	capital_run_ length_total
nobs	4601	4601	4601	4601	4601	4601	4601
NAs	0	0	0	0	0	0	0
Minimum	0	0	0	0	1	1	1
Maximum	32.478	42.81	18.75	4.081	1102.5	9989	15841
1. Quartile	0	0	0	0	1.588	6	35
3. Quartile	0.315	0	2.64	0	3.706	43	266
Mean	0.27	0.07	1.66	0.02	5.19	52.17	283.29
Median	0.00	0.00	1.31	0.00	2.28	15.00	95.00
Variance	0.67	1.95	3.15	0.01	1006.76	37982.62	367657.72
Stdev	0.82	1.40	1.78	0.11	31.73	194.89	606.35
Skewness	18.65	26.21	1.59	21.07	23.75	30.74	8.70
Kurtosis	606.53	725.34	5.25	617.53	669.35	1478.39	145.61
NonZeroCount	2258	47	3227	529	4601	4601	4601
PercentNonZeros	49.08%	1.02%	70.14%	11.50%	100.00%	100.00%	100.00%

Figure 1.3: basic statistics of a sample of continuous variables

The following observations are notable and can be made about the basic statistics of the data set:

1. There are no missing observations
2. The maximum word_freq percentage is for the word “3d”, with a maximum of 42% of the words of an email, and the maximum char_freq is “!” with 32% of the characters.
3. There is a high proportion of zeros in every field which represents a word count. The maximum is the word “you” which appears in over 70% of the emails, the minimum is “3d” which appears in just 1% of the emails
4. The average rate of words appearing is 17% across all emails
5. There is a high proportion of zeros in every field which represents a char count. The maximum is “!” which appears in 49% of the emails, the minimum is “[” which appears in just 11.5% of the emails
6. The average rate of chars appearing is 31% across all emails
7. As a result, the columns are both skewed and peaked for all word and char counts

1.3 Unconditional Distributions of continuous variables

A sample of the distributions of the continuous variables is shown in the image below, one image for a word appearance, and one for a char appearance. These distribution are typical for the words and chars, in that there is a high proportion of zeros, and the distribution quickly degrades with a high number of outliers.

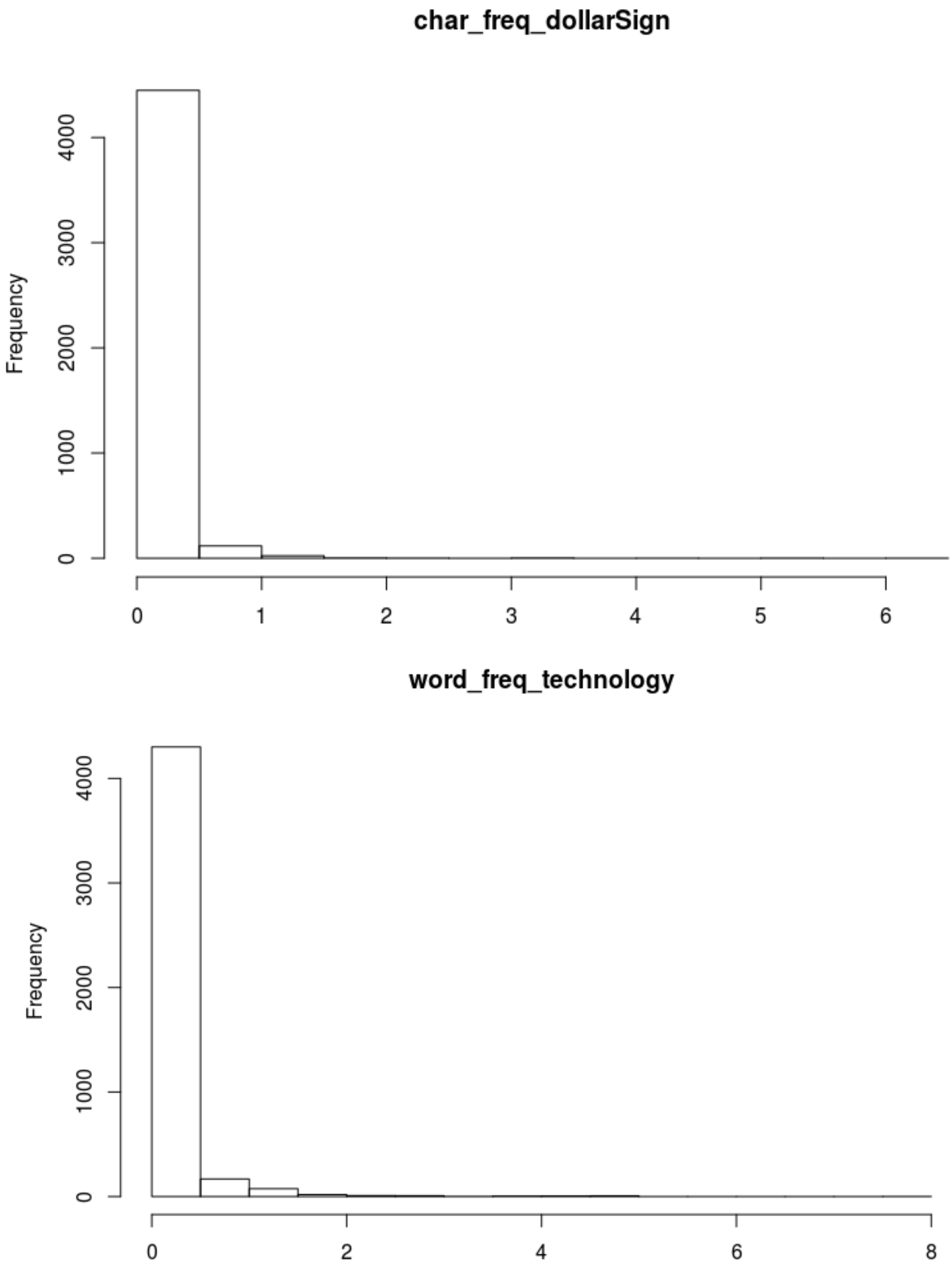


Figure 1.4: Continuous variable distributions: word and char count examples

The distributions of the capital letter metrics follow a similar pattern, although there a no zero entries. The observations of these distributions is that there is a high proportion of zeros, the independent variables are all rare-event type measures

1.4 Distribution of the target variables

The target variable is distributed in the sample as follows

	Count	Percentage
1 (Spam)	1813	39.40%
0 (Not Spam)	2788	60.60%
Total	**4601**	**100.00%**

Figure 1.5: Distribution of the target variable

This indicates there is a strong representation of “Spam” type emails in the data set, so there is sufficient data so as to create a credible model.

1.5 Outlier detection – continuous variables

As the scaled boxplot shows below, there are a high number of outlying observations – this is true across all the predictor variables.

This is natural given the nature of the data as word and char count percentages. There is no reason to suspect any of this data as being erroneous.

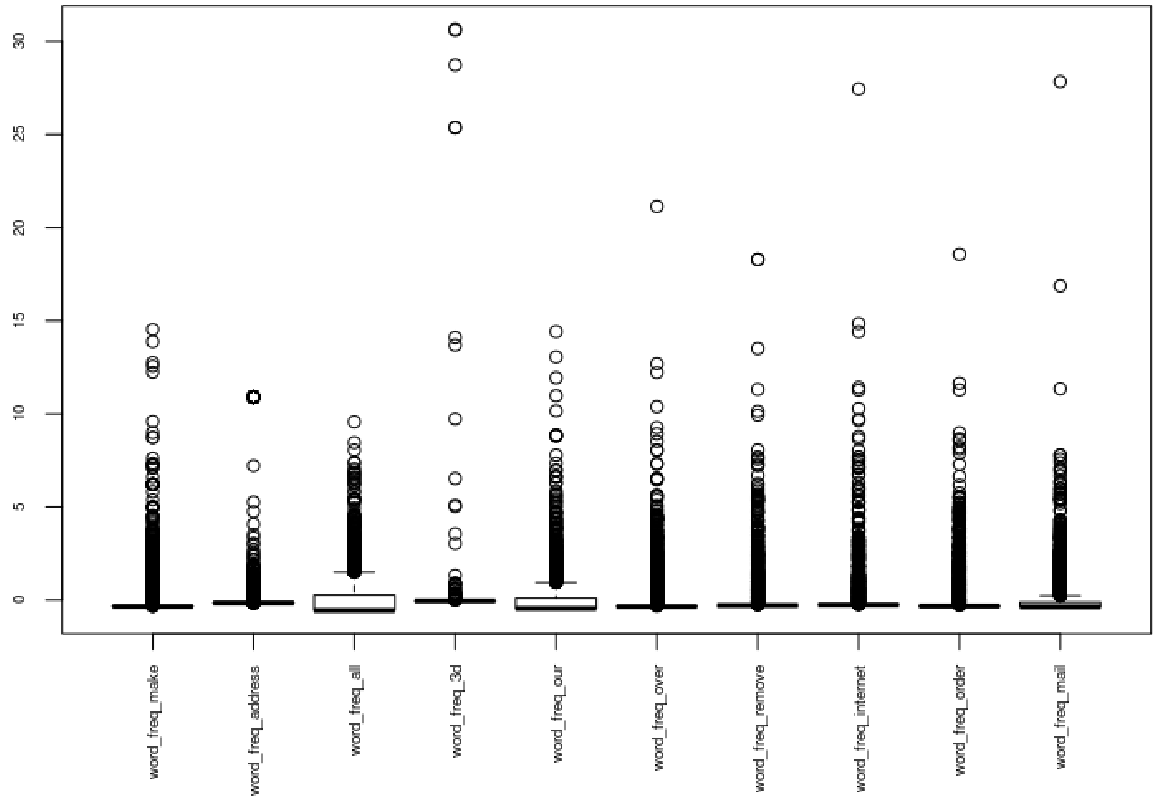


Figure 1.5: Box plot of a sample of variables

Part 2: Multivariate Exploratory Data Analysis

The next section is a natural continuation of the data quality check, however it extends the data analysis in that is begins to apply exploratory techniques to the relationships between the predictors themselves, and the relationships between the predictors and the target, rather than considering these variables in isolation.

2.1 Correlations

Highly correlated data within a multi-variate data set can impact the performance and interpretability of some models, so it is useful to understand the correlations within the predictor data set. The diagram below shows the strength of correlation between the continuous predictors.

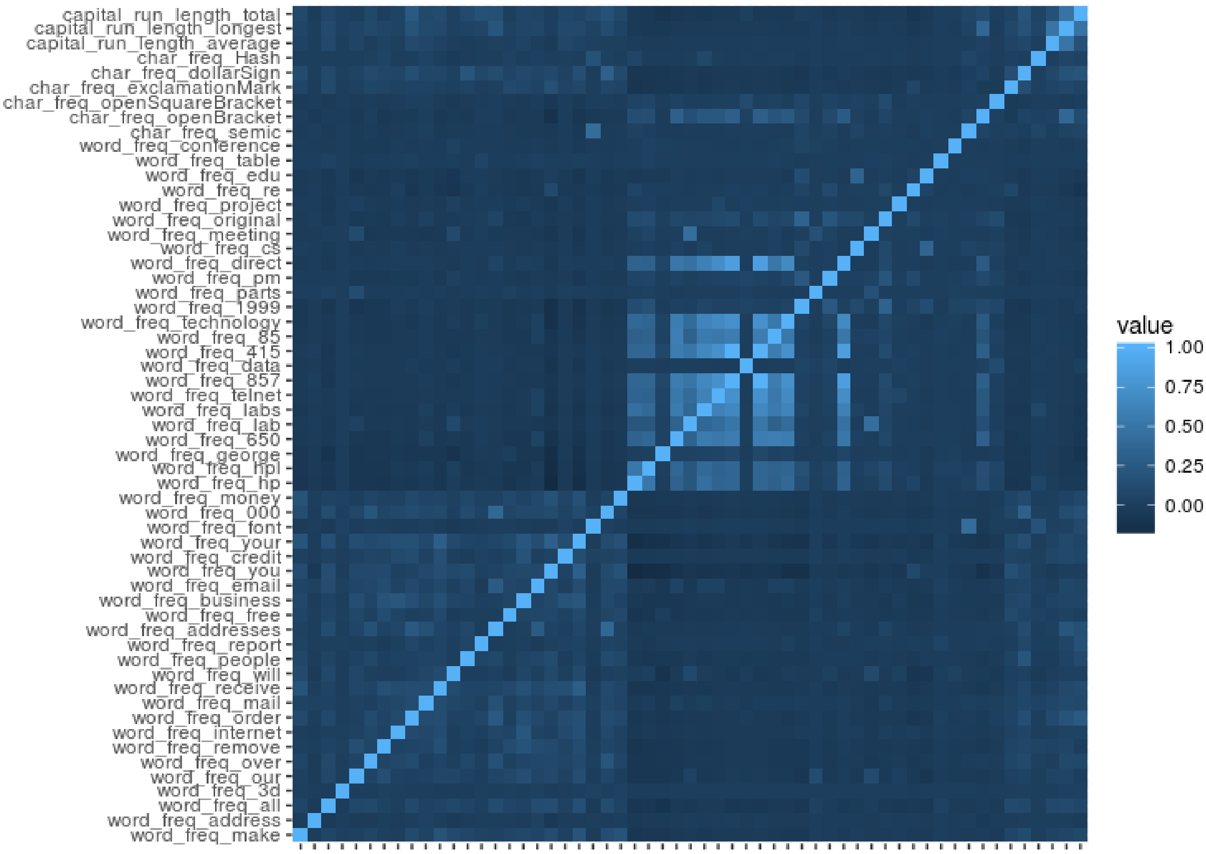


Figure 2.1: Correlations between continuous variables

Variable 1	Variable 2	Correlation
word_freq_415	word_freq_857	0.9960660509
word_freq_direct	word_freq_857	0.8480207086
word_freq_direct	word_freq_415	0.8453591164
word_freq_857	word_freq_telnet	0.7375548552
word_freq_415	word_freq_telnet	0.7351868494
word_freq_technology	word_freq_857	0.7297496053
word_freq_technology	word_freq_415	0.7271185659
word_freq_direct	word_freq_telnet	0.6999181733
word_freq_technology	word_freq_telnet	0.6777902592
word_freq_direct	word_freq_technology	0.6742493941
word_freq_857	word_freq_labs	0.6602842578

Figure 2.2: Top Correlations between continuous variables

Observations from this chart are as follows:

1. Generally, there is very low correlations amongst the predictor variables
2. The most correlated variable pair (word_freq_415 and word_freq_857) are almost perfectly correlated, this suggests that there is little value being added by both variables and one can be dropped
3. The words “857”, “415” are presumably unique to a particular pattern/sender of Spam email. These values do not feel like a generalized model that would hold their value over the long term.

2.2 Conditional Distributions against the target variable

There is apparent separability offered by the continuous variables to the “spam” variable. Some examples are below:

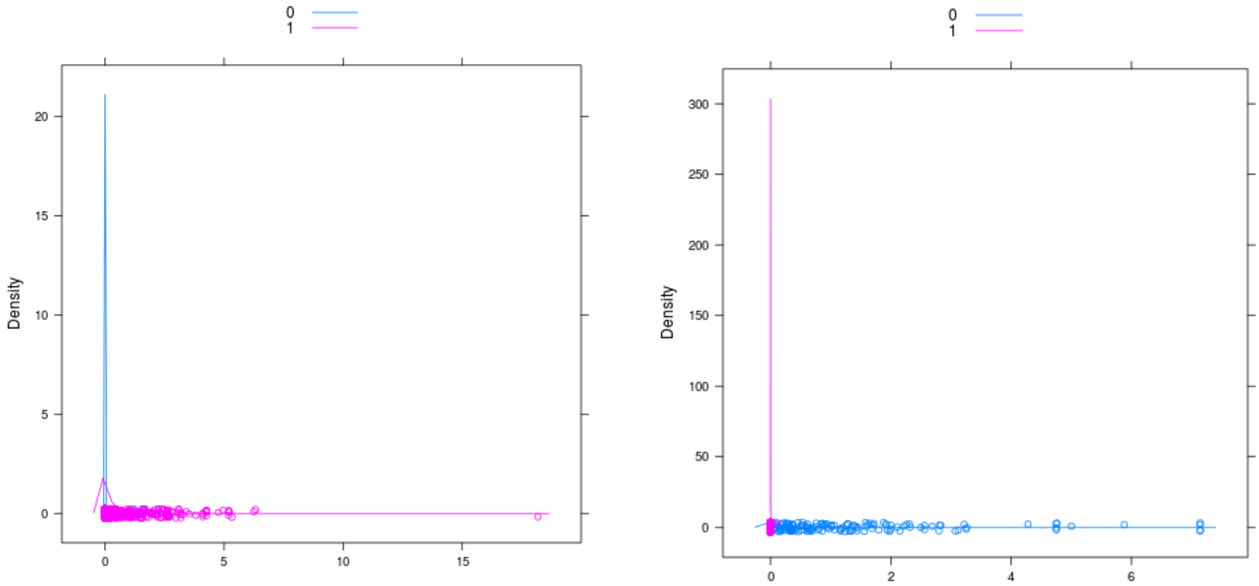


Figure 2.2: Distributions of Spam by the word “credit” (left) and “cs” (right)

These charts indicate that there are some words which are binary indicators of Spam as well as indicators of non-Spam. Emails featuring the word “credit” are almost always Spam, whereas emails featuring the word “cs” are almost always non-Spam.

Part 3: Model Based Exploratory Data Analysis

3.1 Fitting a tree for EDA insight

The diagram below shows the resulting of fitting a tree using a minimum split of 20:

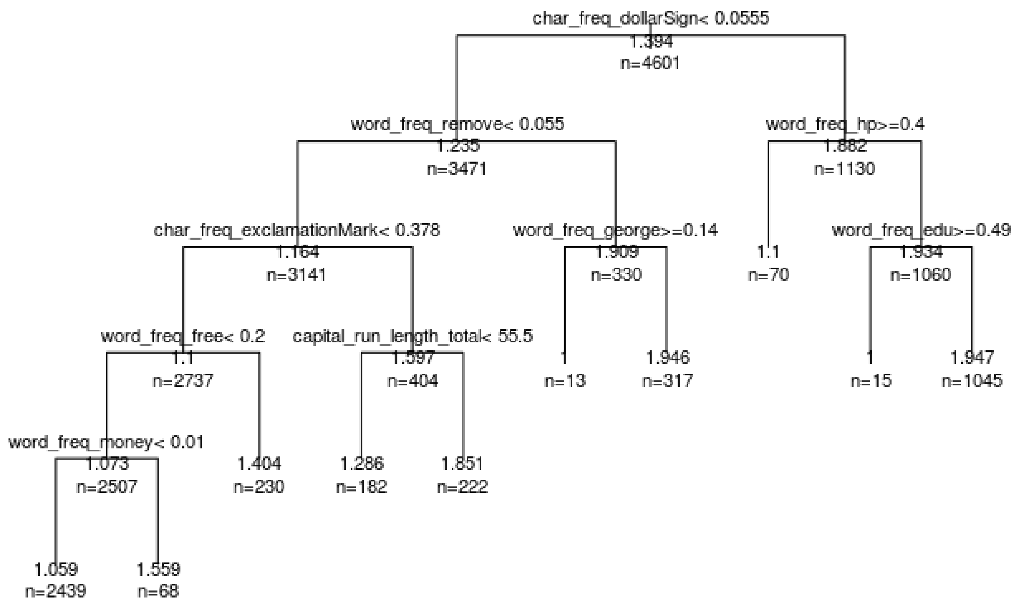


Figure 3.1: Visualization of a simple tree fit to the Data

Variable Name	Importance Measure
char_freq_dollarSign	357.08
word_freq_remove	165.66
word_freq_money	131.88
word_freq_000	126.62
char_freq_exclamationMark	86.63
capital_run_length_longest	85.58
word_freq_credit	61.85
word_freq_order	55.52
word_freq_hp	49.97
capital_run_length_total	32.48
word_freq_free	25.55
word_freq_hpl	23.87

Figure 3.2: Top variable importance measures from the tree

The tree structure suggests some interactions, “Free” and “Money”, the a high capital_run_length with and an exclamation mark. However further analysis will be needed to determine if these interactions are valuable.

This strongly indicates that the “$” character is the most important predictor of the “spam” variable, followed by the words “Remove”, “Money”, “000”.

3.2 Fitting a Random Forest to understand variable importance, transforms and interaction

Fitting a Random Forest in order to understand the importance of variables. The Random Forest confirms the importance of the Dollar Sign (“$”) variable, although not as pronounced, and while the top variables are the similar to the tree, the order is changed:

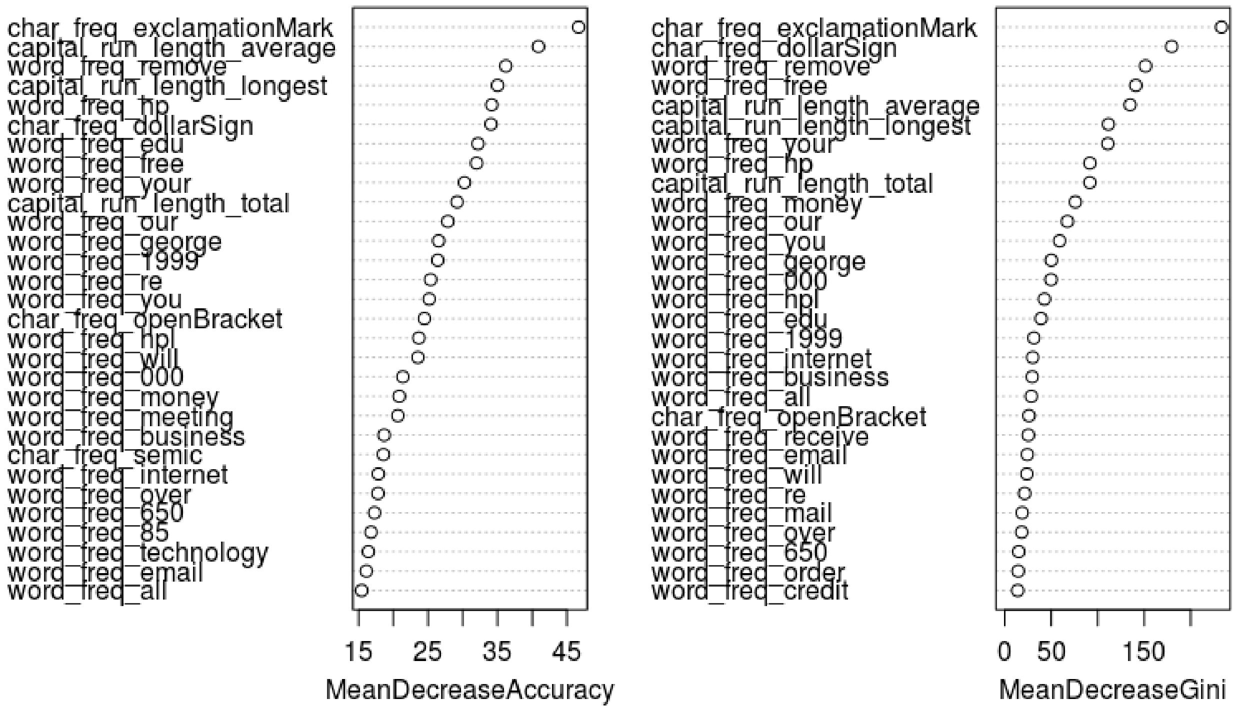


Figure 3.3: Variable importance according to the Random Forest

Part 4: Variable Transforms

There are no missing variables in the data set, and generally the outliers cannot be discounted as being unreasonable, so therefore there is no data imputation required.

Unlike a linear regression, there are no assumptions about the linearity of the relationship between the predictors and the target variable, so transforms for linearity will not be required.

Part 5: Choosing the performance metric

There are a number of metrics which could be used to determine the best model. In a binary classification problem, some of those available are:

1. Accuracy, or how accurate the model across positive and negative predictions.
2. Precision, the proportion of true positives over all positive predictions
3. Sensitivity or Recall, which measures the positive outcomes correctly identified as such
4. Specificity, the proportion of true negatives over all negative predictions

For this business problem, I think the best metric to use here is to target high precision as the primary measure of success. Incorrectly identifying email as Spam, and then taking action on that email, such as moving it out of view, could be very expensive.

The secondary measure will be the Sensitivity, as the goal is to declutter a user's email inbox, so low Sensitivity would be a suboptimal outcome.

Therefore the final measure to determine the best model will be **Max(2*Precision+Sensitivity)**

Part 6: Modeling

For this section, I have split the data 70% training and 30% test. Any model requiring validation will need to split the training set.

6.1 Baseline model: logistic model with all variables

The findings from fitting a logistic model with no variable selection is a high performing model out of sample with a Precision of 0.919 and a Sensitivity of 0.884.

The model parameters are as follows:

Variable	Beta
(Intercept)	-1.6417320759
word_freq_make	-0.143859057
word_freq_address	-0.1555025547
word_freq_all	-0.0034436726
word_freq_3d	1.8430427733
word_freq_our	0.4095468022
word_freq_over	0.9561882712
word_freq_remove	3.1578086352
word_freq_internet	0.4810065068
word_freq_order	0.5562562526
word_freq_mail	0.1767582158
word_freq_receive	-0.8971688778
word_freq_will	-0.1642757032
word_freq_people	-0.4075238965
word_freq_report	0.129228776
word_freq_addresses	3.1556027367
word_freq_free	1.2368994903
word_freq_business	1.1893922564
word_freq_email	-0.0153388143
word_freq_you	0.1036458558
word_freq_credit	1.3209552906
word_freq_your	0.2144687899
word_freq_font	0.0776369829
word_freq_000	3.317014341
word_freq_money	1.468637172
word_freq_hp	-2.3292752596
word_freq_hpl	-0.6663971668
word_freq_george	-17.8729438824
word_freq_650	0.4237705945

Variable	Beta
word_freq_lab	-2.4605961253
word_freq_labs	-0.1171773516
word_freq_telnet	-3.9839732806
word_freq_857	1.9494055506
word_freq_data	-0.81462856
word_freq_415	-7.5284397585
word_freq_85	-1.8197146971
word_freq_technology	1.2116768463
word_freq_1999	-0.0832539243
word_freq_parts	-0.7187278475
word_freq_pm	-0.9450893268
word_freq_direct	-0.1355742699
word_freq_cs	-514.4400581668
word_freq_meeting	-3.4163925644
word_freq_original	-1.3200652906
word_freq_project	-1.5866616735
word_freq_re	-0.9159744504
word_freq_edu	-1.5928347727
word_freq_table	-4.7361847779
word_freq_conference	-6.2299865454
char_freq_semic	-1.2441452207
char_freq_openBracket	-0.275350643
char_freq_openSquareBracket	-0.7513203067
char_freq_exclamationMark	0.2581075397
char_freq_dollarSign	5.9461664614
char_freq_Hash	3.5205593885
capital_run_length_average	0.0410296329
capital_run_length_longest	0.0091734429
capital_run_length_total	0.0011167992

Figure 6.1: Variable coefficients for a simple logistic regression (all predictors)

Observations from this model are as follows:

1. 29 of the variables are deemed to be significant at the 95% level
2. A number of variables seem to have a strong negative (non-spam) impact, for example the word “cs”, “415”, “telnet”, “lab”, “hp” and “george”
3. The “$”, “credit”, “remove”, “addresses”, “3d” are confirmed as an important contributor towards spam in this model

6.2 Regression model using Stepwise variable selection

Running a Stepwise variable selection logistic regression serves to reduce the number of variables from 57 to 44, however with a slight reduction in out-of-sample precision (0.917), the model performs worse than the full set of variables.

This resulted in the following model coefficients:

Variable	Beta
(Intercept)	-1.7201506041
word_freq_address	-0.1446472521
word_freq_3d	1.774328955
word_freq_our	0.4175689372
word_freq_over	0.8954102012
word_freq_remove	3.1650226244
word_freq_internet	0.5015337409
word_freq_order	0.537121646
word_freq_mail	0.1796680269
word_freq_receive	-0.953760851
word_freq_will	-0.1652654901
word_freq_addresses	2.6696181827
word_freq_free	1.2545522203
word_freq_business	1.1588033342
word_freq_you	0.1042275106
word_freq_credit	1.4354858551
word_freq_your	0.2167152089
word_freq_000	3.2386493485
word_freq_money	1.4093564535
word_freq_hp	-2.3919752878
word_freq_hpl	-0.6510226201
word_freq_george	-17.665813725
word_freq_650	0.3896638981

Variable	Beta
word_freq_lab	-2.4807805314
word_freq_telnet	-3.8551984519
word_freq_data	-0.7902726181
word_freq_85	-1.7890413147
word_freq_technology	1.3040385962
word_freq_parts	-0.718507406
word_freq_pm	-1.0051368274
word_freq_cs	-511.7504014164
word_freq_meeting	-3.4897388165
word_freq_original	-1.2576845523
word_freq_project	-1.6173767876
word_freq_re	-0.9236725356
word_freq_edu	-1.6105743166
word_freq_table	-5.2249083379
word_freq_conference	-6.402744348
char_freq_semic	-1.0997635758
char_freq_exclamationMark	0.2614462392
char_freq_dollarSign	5.9551651141
char_freq_Hash	3.7237791666
capital_run_length_average	0.040399673
capital_run_length_longest	0.0091208616
capital_run_length_total	0.0010852516

Figure 6.2: Variable coefficients for a simple logistic regression (all predictors)

Observations from this Stepwise model are as follows:

1. The low-importance variables have been removed, words such as “make”, “all”, “people” and “report”
2. Similar patterns exist as for the initial model in terms of the significant words and characters.

6.3: Support Vector Machine

Fitting a variety of Support Vector Machine Models, varying initially the scaling of the variables, and the type of kernel, the following results were observed:

Type of SVM	Precision	Sensitivity	Performance Metric
Unscaled, linear Kernel, cost=10	0.89	0.772	2.552
Scaled, linear Kernel, cost=10	0.92	0.893	2.733
Scaled, Polynomial Kernel, cost=10	0.924	0.705	2.553
Scaled, Radial Kernel, cost=10	0.928	0.886	2.742

This indicated that a SVM with a Radial Kernel with scaled variables was worth tuning further. A range of Costs and Gammas was studied, resulting in the following cross-validation error. The optimal SVM model was when the gamma was 0.01 and the cost was 10.

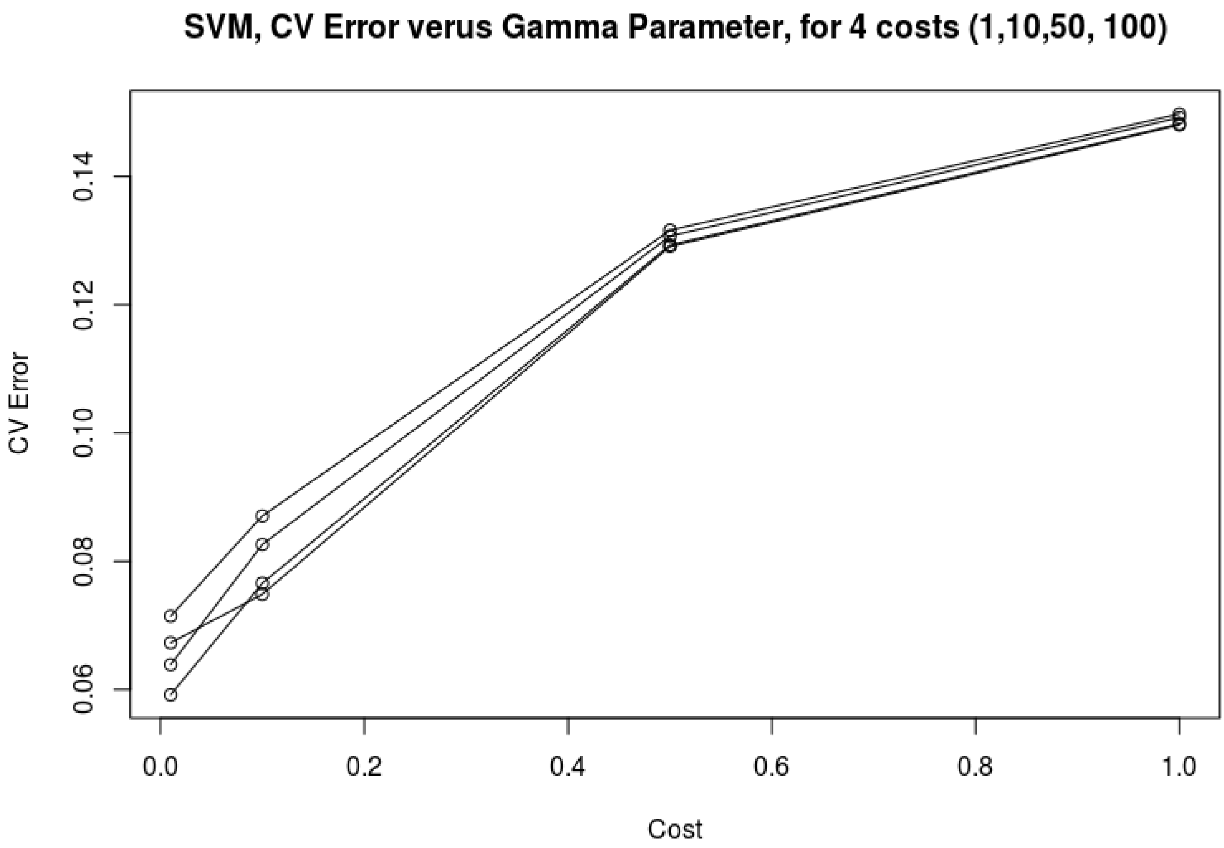
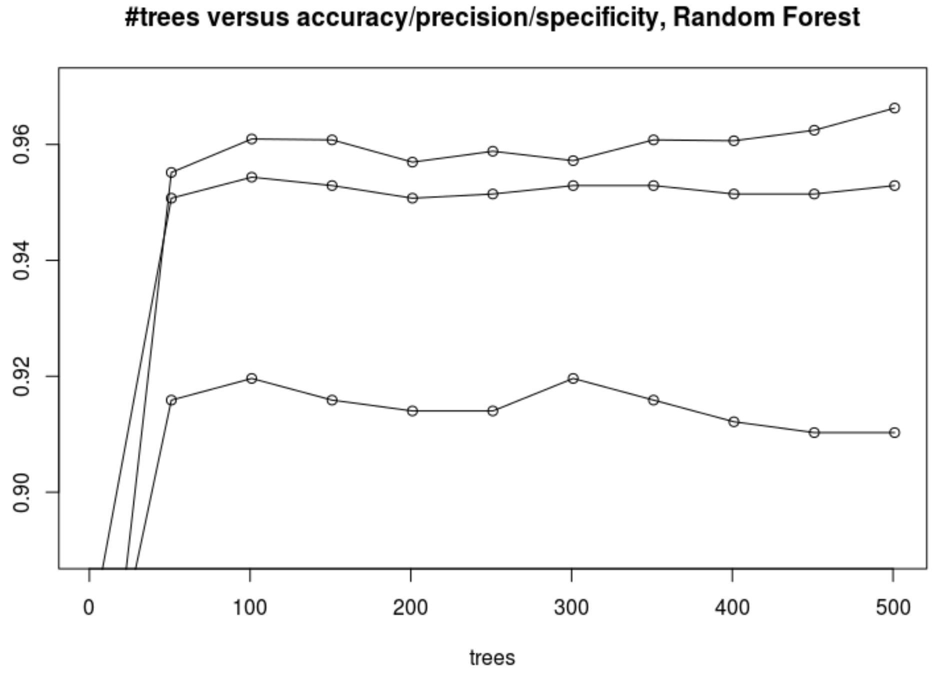


Figure 6.x: CV performance of SVM Radial models varying gamma and cost

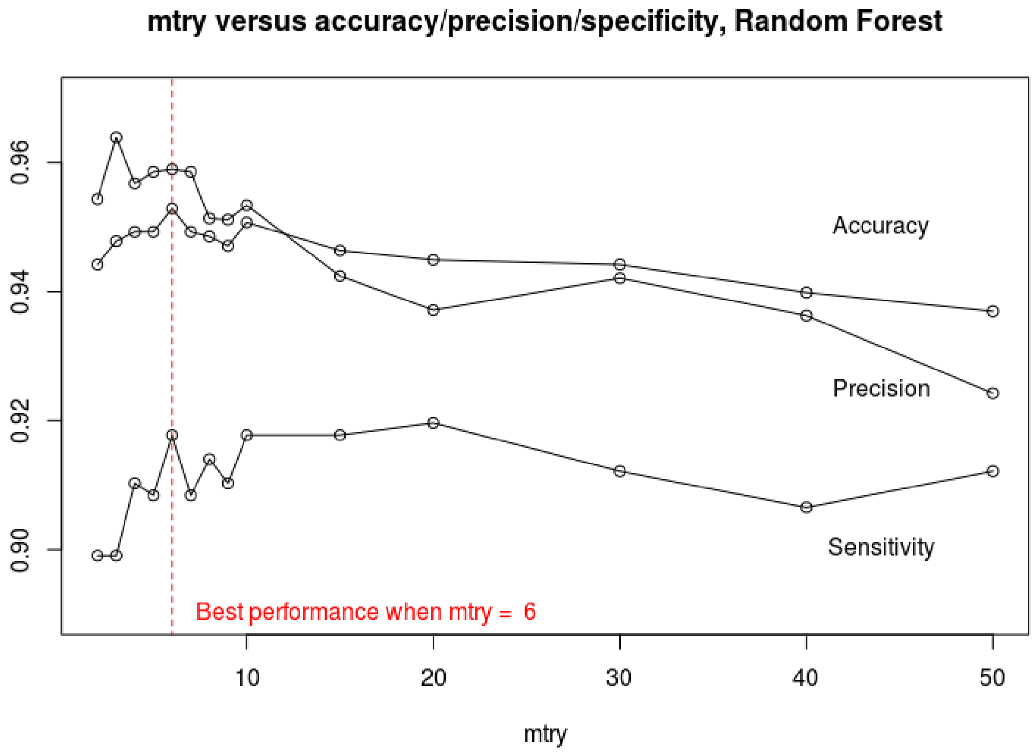
The final, tuned, SVM model gave a precision of 0.935 and a Sensitivity of 0.89 on out of sample testing. When fitting to a reduced data set (44 variables), the performance was slightly degraded

6.4: Random Forest

Fitting a Random Forest ensemble model to the full data set yielded the following results, firstly when varying the number of trees:



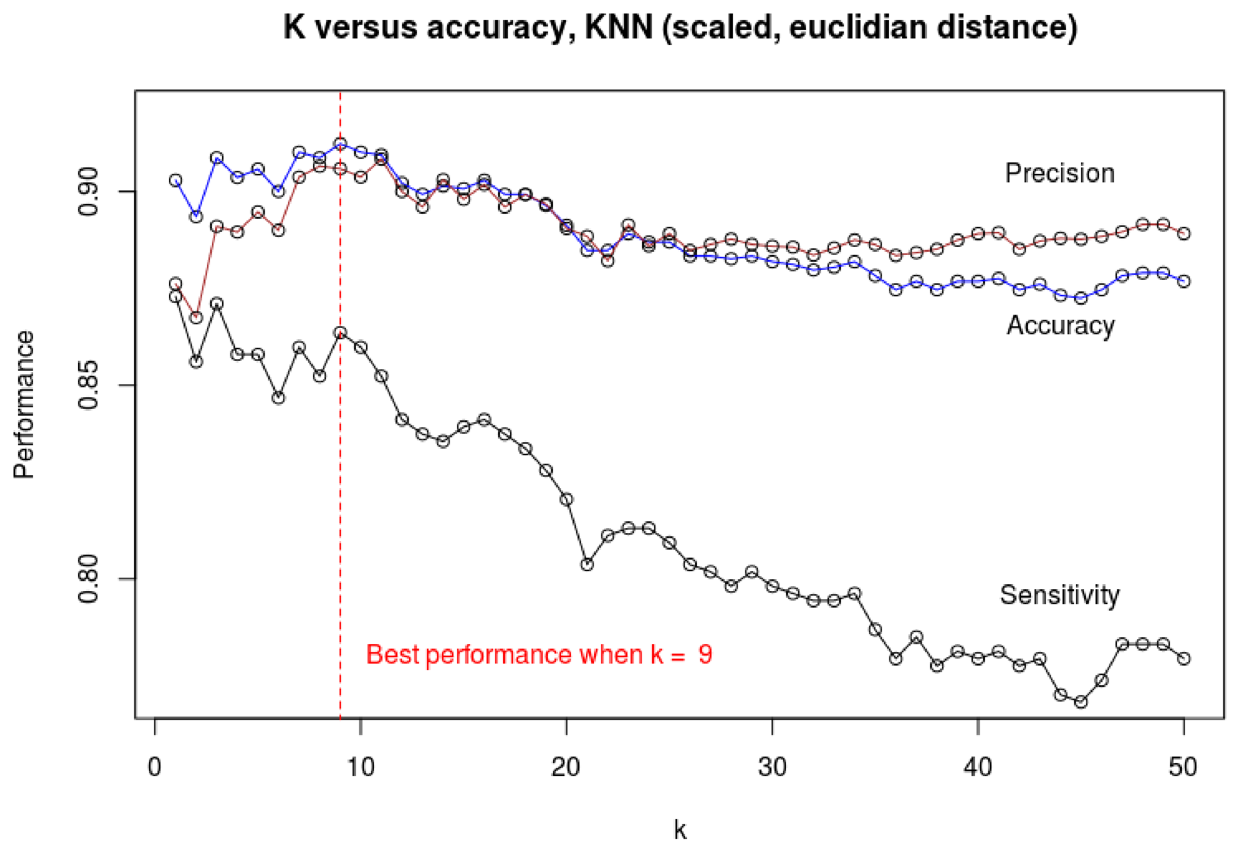
We see that the performance flattens when the number of trees is approximately 100. By examining the performance of the Random Forest as we adjust mtry, or the number of variables in the trees making up the Forest:



We see that the optimal mtry number is 6. For this Random Forest, the precision is 0.959 and the Specificity is 0.908. When fitting to a reduced data set (44 variables), the performance was slightly degraded

6.5 KNN

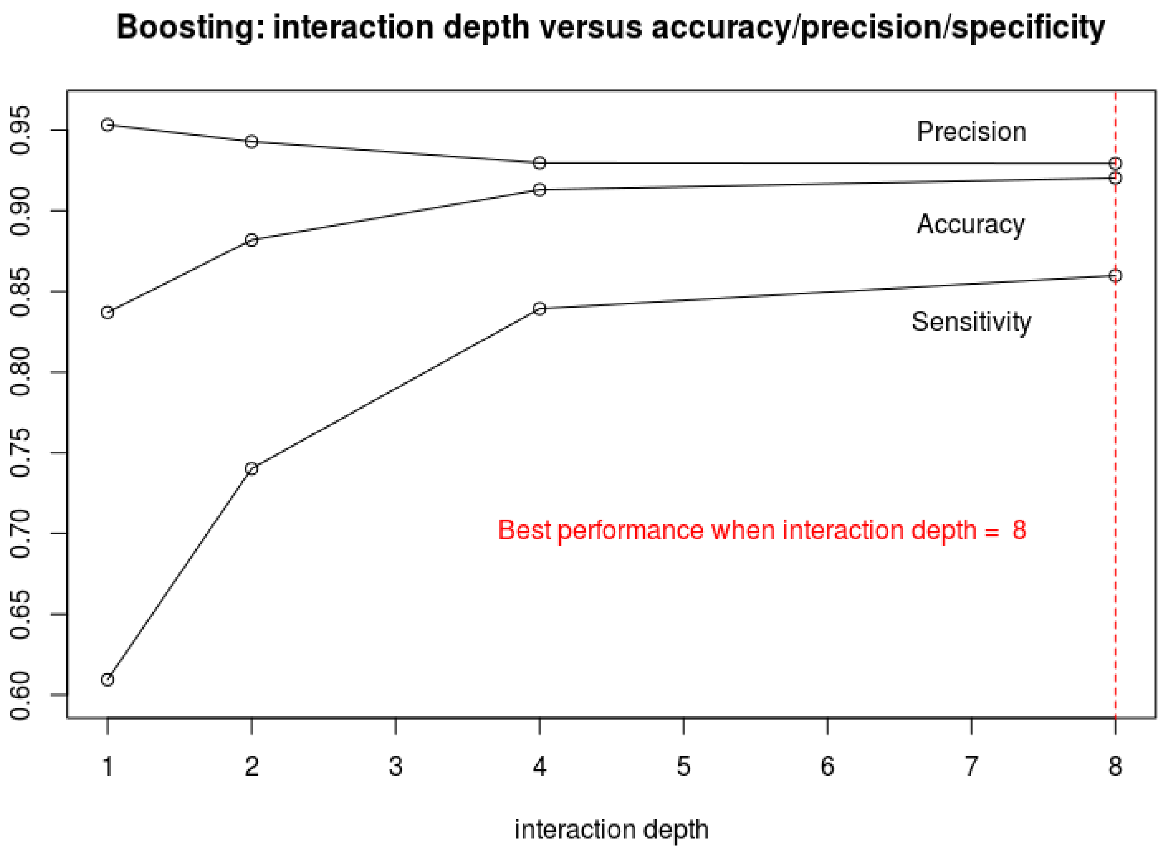
With all the predictors being continuous, this presents an opportunity to use KNN. By adjusting k, using a scaled data set, and a Euclidean distance measure, the performance of the models was as follows:



The best performing model, when k=9, had a Precision of 0.906 and Sensitivity of 0.864.

6.6 Gradient Boosted Trees

The final model explored for this data was Gradient Boosted Trees. I used a tree size of 1000 and by varying the interaction depth, according to my measure, the best performing model had an interaction depth of 8, as can be seen below:



The precision for the best performing boosted model was 0.923 and the Sensitivity was 0.86.

Part 7: Model Comparison and Conclusion

7.1 Model Comparison

The final model comparison for all models developed is shown below, ordered in descending order of performance.

Model Type	Number of Variables	Out of sample Accuracy	Out of Sample Precision	Out of Sample Sensitivity	Performance Measure (2*Precision	Position
Best Random Forest, mtry=8, ntree=100	57	0.949	0.959	0.908	2.826	1
Random Forest, reduced variable set, mtry=8, ntree=100	44	0.947	0.957	0.905	2.819	2
SVM, Scaled, Radial Kernel, cost=10, gamma = 0.01	57	0.933	0.935	0.890	2.76	3
SVM, Scaled, Radial Kernel, cost=10, gamma = 0.01, reduced variable set	44	0.936	0.924	0.908	2.756	4
SVM, Scaled, Radial Kernel, cost=10	57	0.929	0.928	0.886	2.742	5
SVM, Scaled, linear Kernel, cost=10	57	0.928	0.92	0.893	2.733	6
Baseline logistic regression model	57	0.925	0.919	0.884	2.722	7
Logistic regression model with StepWise selection	44	0.925	0.917	0.886	2.72	8
Best Gradient Boosted Trees	57	0.920	0.930	0.860	2.72	9
Best KNN, k=9, Euclidian distance	57	0.912	0.906	0.864	2.676	10
SVM, Scaled, Polynomial Kernel, cost=10	57	0.863	0.924	0.705	2.553	11
SVM, Unscaled, linear Kernel, cost=10	57	0.875	0.89	0.772	2.552	12
SVM, Scaled, Sigmoid Kernel, cost=10	57	0.85	0.8	0.82	2.42	13

7.2 Conclusion

The particularly interesting aspects of this project are the sparseness and volume of the predictor variables, and the choice of success measure for the modeling exercise.

There was little opportunity or value in variable transformation, outside of scaling the variables where appropriate, as the models did not depend on a linear relationship between the independent and the target variable, and there was no missing variables or variable requiring imputation.

Similarly there was little additional value, interpretability aside, gained through reducing the variable set for all models including the logistic regression models, with the Stepwise selection model performing worse than a full logistic regression for all variables, and the same being true for the non-linear models.

However there is clearly an opportunity to find the optimally performing model through tuning the models, and the non-linear models (SVM, Random Forests) outperformed the Logistic, KNN and Boosted Tree models.

Appendix: R-Code

R-code developed in the course of this project

```
#setting up the data set
setwd("~/Downloads/PA454/6_week")

df <- read.csv("spambase.data", header = FALSE)
header <- read.csv("spambase.names", header = FALSE)
t <- c("spam")
r <- as.character(unlist(header))
headervec <-  c(r,t)
colnames(df) <- headervec

#fixing the spam as a a factor
df$spam <- as.factor(df$spam)

# Basic Stats)
stats <- my.Summary(df_cont)
write.csv(stats, "stats.csv")

# Histograms
PlotHistograms(df_cont)
PlotNonZeroHistograms(df_cont)

#boxplots for outliers
boxplot(scale(df_cont[,1:10]), cex.axis=0.60, las=3)

#4 correlations
df_cont <- df[,-58]
cor <- cor(df_cont, method="pearson")
library(ggplot2)
library(reshape2)
qplot(x=Var1, y=Var2, data=melt(cor), fill=value, geom="tile")

library(corrplot)
col3 <- colorRampPalette(c("white", "blue"))
corrplot(cor, order="AOE",  method="color",  col=col3(10),addCoef.col="black", tl.cex=1,
tl.col="black",type = c("lower"),diag = FALSE)
corrplot(cor)

#get the top correlations
library(dplyr)
library(reshape2)
d_cor <- as.matrix(cor)
d_cor_melt <- arrange(melt(d_cor), -abs(value))
write.csv(d_cor_melt, "correlations.csv")

#Advanced EDA: conditional density plots
PlotConditionalDesityPlotsGrouped(df)

#Model based EDA (tree)
EDA_tree <- rpart(spam~., data=df, method="anova",control = rpart.control(minsplit = 20))
plot(EDA_tree, margin = 0.05)
text(EDA_tree, use.n=TRUE, all=TRUE, cex=0.7)
summary(EDA_tree)
write.csv(EDA_tree$variable.importance,"varimp.csv")

#Model based EDA (RF)
library("randomForest")
rf = randomForest(spam ~ ., data = df, mtry = 5, ntree = 500, importance = T)
importance(rf,type=1)
varImpPlot(rf)
partialPlot(rf, df, "word_freq_project" )

#trying to find interaction
library(plotmo)
plotmo(rf)

#actual modelling
#set up training/test
temp_index <- sample(nrow(df), round(nrow(df)*0.3))
df_training <- df[-temp_index,]
df_test <- df[temp_index,]

rownames(df_test) <- seq(length=nrow(df_test))
```

```
rownames(df_training) <- seq(length=nrow(df_training))
validation_index <- sample(nrow(df_training), round(nrow(df_training)*0.2))

#for the performance meausre functions:
df_test$class <- df_test$spam

#model1: baseline linear regression
model1 <- glm(spam ~ ., data = df_training, family = "binomial")
summary(model1)
pred1 <- predict(model1, df_test, type="response")
df_test$scored.class <- round(pred1,0)
PrintPerformanceMetrics(df_test)
write.csv(model1$coefficients, "coef.csv")

#model2: stepwise glm selection
model2 <- step(model1, direction="both")
summary(model2)
pred2 <- predict(model2, df_test, type="response")
df_test$scored.class <- round(pred2,0)
PrintPerformanceMetrics(df_test)

#model3: forward glm selection
model3 <- step(model1, direction="forward")
summary(model3)
pred3 <- predict(model3, df_test, type="response")
df_test$scored.class <- round(pred3,0)
PrintPerformanceMetrics(df_test)

#model4: backward glm selection
model4 <- step(model1, direction="backward")
summary(model4)
pred4 <- predict(model4, df_test, type="response")
df_test$scored.class <- round(pred4,0)
PrintPerformanceMetrics(df_test)

#trees (methood="class" or "anova" )
library(rpart)
tree <- rpart(spam~., data=df_training, control = rpart.control(minsplit = 5))
pred_TREE <- predict(tree, df_test, type = c("class"))
df_test$scored.class <- pred_TREE
PrintPerformanceMetrics(df_test)

#tree, plotting
plot(tree, uniform = TRUE,margin = 0.2)
text(tree, use.n=TRUE, all=TRUE, cex=0.8)

#model: support vector machine
library(e1071)
c <- 10
svmmodel <- svm(spam ~ ., data = df_training, kernel="linear", cost = c, scale=FALSE)
svmpred <- predict(svmmodel, df_test)
df_test$scored.class <- svmpred
PrintPerformanceMetrics(df_test)

svmmodel2 <- svm(spam ~ ., data = df_training, kernel="linear", cost = c, scale=TRUE)
svmpred2 <- predict(svmmodel2, df_test)
df_test$scored.class <- svmpred2
PrintPerformanceMetrics(df_test)

svmmodel3 <- svm(spam ~ ., data = df_training, kernel="polynomial", cost = c, scale=TRUE)
svmpred3 <- predict(svmmodel3, df_test)
df_test$scored.class <- svmpred3
PrintPerformanceMetrics(df_test)

svmmodel4 <- svm(spam ~ ., data = df_training, kernel="radial", cost = c, scale=TRUE)
svmpred4 <- predict(svmmodel4, df_test)
df_test$scored.class <- svmpred4
PrintPerformanceMetrics(df_test)

#using the internal cross validation of SVM
svmtune <- tune(svm,as.numeric(spam) ~ ., data = df_training, kernel="radial", ranges =
list(cost=c(1,10,50, 100),gamma=c(0.01, 0.1, 0.5, 1)))

graphSVM <- summary(svmtune)$performances
plot(graphSVM[,2],graphSVM[,3] , xlab="Cost", ylab="CV Error", main="SVM, CV Error verus Gamma
Parameter, for 4 costs (1,10,50, 100)")
lines(graphSVM[graphSVM[,1]==1e+02,2],graphSVM[graphSVM[,1]==1e+02,3] )
lines(graphSVM[graphSVM[,1]==5e+01,2],graphSVM[graphSVM[,1]==5e+01,3] )
lines(graphSVM[graphSVM[,1]==1e+01,2],graphSVM[graphSVM[,1]==1e+01,3] )
```

```
lines(graphSVM[graphSVM[,1]==1e+00,2],graphSVM[graphSVM[,1]==1e+00,3] )

svmmodel5 <- svm(spam ~ ., data = df_training, kernel="sigmoid", cost = c, scale=TRUE)
svmpred5 <- predict(svmmodel5, df_test)
df_test$scored.class <- svmpred5
PrintPerformanceMetrics(df_test)

#best model
df_training$spam <- as.factor(df_training$spam)
svmmodel6 <- svm(spam ~ ., data = df_training, kernel="radial", cost = 10, gamma=0.01, scale=TRUE)
svmpred6 <- predict(svmmodel6, df_test)
df_test$scored.class <- svmpred6
PrintPerformanceMetrics(df_test)

#best model, reduced variable set
svmmodel7 <- svm(formula1, data = df_training, kernel="radial", cost = 10, gamma=0.01, scale=TRUE)
svmpred7 <- predict(svmmodel7, df_test)
df_test$scored.class <- svmpred7
PrintPerformanceMetrics(df_test)

#model: knn, note for report: scaling improves perofrmance significantly, presumably because it
distorts distance less
library(class)
results = matrix(nrow=50,ncol=4)
for(kn in 1:50)
{
predknn <- knn(scale(df_training[,1:57]), scale(df_test[,1:57]), df_training[,58], k = kn)
df_test$scored.class <- predknn
results[kn,1] <- kn
results[kn,2] <- FAccuracy(df_test)
results[kn,3] <- FPrecision(df_test)
results[kn,4] <- FSensitivity(df_test)
}
plot(results, xlab="k", ylab="Performance", main="K versus accuracy, KNN (scaled, euclidian
distance)", ylim=c(0.77,0.92))
lines(results, col="blue")
lines(results[,1],results[,3], col="brown")
lines(results[,1],results[,4])
points(results[,1],results[,3])
points(results[,1],results[,4])
#lines for the best k
maxAccuracy <- results[which.max(2*results[,3]+results[,4]),1]
lines(c(maxAccuracy,maxAccuracy),c(0,1), col="red",lty=2)
text(x=maxAccuracy+0.5, y=0.78, paste("Best performance when k = ",maxAccuracy),pos=4, col="red")

text(44, 0.795,"Sensitivity")
text(44, 0.905,"Precision")
text(44, 0.865,"Accuracy")

#model: RF (tuning tree number)
results = matrix(nrow=11,ncol=4)
n<-1
for(tr in seq(from=1, to=501, by=50))
{
rf = randomForest(spam ~ ., data = df_training, mtry = 5, ntree = tr, importance = T)
pred7 <- predict(rf,df_test)
df_test$scored.class <- pred7
df_test$class <- df_test$spam
results[n,1] <- tr
results[n,2] <- FAccuracy(df_test)
results[n,3] <- FPrecision(df_test)
results[n,4] <- FSensitivity(df_test)
n<-n+1
}

plot(results, xlab="trees", ylab="", main="#trees versus accuracy/precision/specificity, Random
Forest", ylim=c(0.89,0.97))
lines(results)
lines(results[,1],results[,3])
lines(results[,1],results[,4])
points(results[,1],results[,3])
points(results[,1],results[,4])

#model: RF (tuning mtry number)
results = matrix(nrow=14,ncol=4)
n<-1
for(mt in c(2,3,4,5,6,7,8,9,10,15,20,30,40,50))
{
  rf = randomForest(spam ~ ., data = df_training, mtry = mt, ntree = 100, importance = T)
```

```
  pred7 <- predict(rf,df_test)
  df_test$scored.class <- pred7
  df_test$class <- df_test$spam
  results[n,1] <- mt
  results[n,2] <- FAccuracy(df_test)
  results[n,3] <- FPrecision(df_test)
  results[n,4] <- FSensitivity(df_test)
  n<-n+1
}

plot(results, xlab="mtry", ylab="", main="mtry versus accuracy/precision/specificity, Random
Forest", ylim=c(0.89,0.97))
lines(results)
lines(results[,1],results[,3])
lines(results[,1],results[,4])
points(results[,1],results[,3])
points(results[,1],results[,4])

#plot a vertical line with the max accuracy:
maxAccuracy <- results[which.max(2*results[,3]+results[,4]),1]
lines(c(maxAccuracy,maxAccuracy),c(0,1), col="red",lty=2)
text(x=maxAccuracy+0.5, y=0.89, paste("Best performance when mtry = ",maxAccuracy),pos=4,
col="red")

text(44, 0.9,"Sensitivity")
text(44, 0.925,"Precision")
text(44, 0.95,"Accuracy")

#RF, reduced variables
df_training$spam <- as.factor(df_training$spam)
formula1 <- as.formula("spam ~ word_freq_address + word_freq_3d + word_freq_our + word_freq_over +
word_freq_remove + word_freq_internet + word_freq_order + word_freq_mail + word_freq_receive +
word_freq_will + word_freq_addresses + word_freq_free + word_freq_business + word_freq_you +
word_freq_credit + word_freq_your + word_freq_000 + word_freq_money + word_freq_hp + word_freq_hpl
+ word_freq_george + word_freq_650 + word_freq_lab + word_freq_telnet + word_freq_data +
word_freq_85 + word_freq_technology + word_freq_parts + word_freq_pm + word_freq_cs +
word_freq_meeting + word_freq_original + word_freq_project + word_freq_re + word_freq_edu +
word_freq_table + word_freq_conference + char_freq_semic + char_freq_exclamationMark +
char_freq_dollarSign + char_freq_Hash + capital_run_length_average + capital_run_length_longest +
capital_run_length_total")
rf2 = randomForest(formula1, data = df_training, mtry = 5, ntree = 100, importance = T)
pred8 <- predict(rf2,df_test)
df_test$scored.class <- pred8
df_test$class <- df_test$spam
PrintPerformanceMetrics(df_test)

#boosting
library(gbm)
df_training$spam <- as.character(df_training$spam)

results = matrix(nrow=4,ncol=4)
n<-1
for(idde in c(1,2,4,8))
{
modelboost <- gbm(spam ~ ., data = df_training, distribution="bernoulli", n.trees=1000,
interaction.depth=idde, verbose = TRUE)
boostpred <- predict(modelboost, df_test, n.trees=1000, type="response")
boostpred <- round(boostpred)
df_test$scored.class <- boostpred
results[n,1] <- idde
results[n,2] <- FAccuracy(df_test)
results[n,3] <- FPrecision(df_test)
results[n,4] <- FSensitivity(df_test)
n<-n+1
}

plot(results, xlab="interaction depth", ylab="", main="Boosting: interaction depth versus
accuracy/precision/specificity", ylim=c(0.6,0.96))
lines(results)
lines(results[,1],results[,3])
lines(results[,1],results[,4])
points(results[,1],results[,3])
points(results[,1],results[,4])

text(7, 0.83,"Sensitivity")
text(7, 0.95,"Precision")
text(7, 0.89,"Accuracy")

#plot a vertical line with the max accuracy:
```

```
maxAccuracy <- results[which.max(2*results[,3]+results[,4]),1]
lines(c(maxAccuracy,maxAccuracy),c(0,1), col="red",lty=2)
text(x=maxAccuracy-0.5, y=0.7, paste("Best performance when interaction depth =
",maxAccuracy),pos=2, col="red")
```