Assignment 3

Kai Lukowiak

2018-03-25

Abstract

This papers using logistic regression to analyze the Boston Crime Dataset to predict if an area has crime above or bellow the median. This work is completed for the CUNY course DATA 621

Contents

1 Data Exploration

1.1 The Data Frames

Table 1: Sample of Values for the Training Set

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	18.10	0	0.713	5.976	87.9	2.5806	24	666	20.2	19.01	12.7	1
0	10.59	0	0.489	5.783	72.7	4.3549	4	277	18.6	18.06	22.5	0

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
28	15.04	0	0.464	6.249	77.3	3.6150	4	270	18.2	10.59	20.6	0
0	8.56	0	0.520	6.167	90.0	2.4210	5	384	20.9	12.33	20.1	0
0	18.10	1	0.631	6.683	96.8	1.3567	24	666	20.2	3.73	50.0	1
0	27.74	0	0.609	5.093	98.0	1.8226	4	711	20.1	29.68	8.1	0

The training or labeled data-set is comprised of 12 categorical and continuous variables and one `target` variable that indicates if an area is higher crime.

Table 2: Sample of Values for the Test Set

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
0	18.10	0	0.740	6.219	100.0	2.0048	24	666	20.2	16.59	18.4
0	18.10	0	0.584	6.348	86.1	2.0527	24	666	20.2	17.64	14.5
0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	10.26	18.2
0	18.10	0	0.655	6.209	65.4	2.9634	24	666	20.2	13.22	21.4
0	9.69	0	0.585	5.794	70.6	2.8927	6	391	19.2	14.10	18.3
0	4.49	0	0.449	6.121	56.8	3.7476	3	247	18.5	8.44	22.2

The evaluation set is a similar data frame but excludes the target variable. As such it cannot be used for cross validation.

- `zn`: proportion of residential land zoned for large lots (over 25000 square feet) (predictor variable)
- `indus`: proportion of non-retail business acres per suburb (predictor variable)
- `chas`: a dummy var. for whether the suburb borders the Charles River (1) or not (0) (predictor variable)
- `nox`: nitrogen oxides concentration (parts per 10 million) (predictor variable)
- `rm`: average number of rooms per dwelling (predictor variable)
- `age`: proportion of owner-occupied units built prior to 1940 (predictor variable)
- `dis`: weighted mean of distances to five Boston employment centers (predictor variable)
- `rad`: index of accessibility to radial highways (predictor variable)
- `tax`: full-value property-tax rate per $10,000 (predictor variable)
- `ptratio`: pupil-teacher ratio by town (predictor variable)
- `lstat`: lower status of the population (percent) (predictor variable)
- `medv`: median value of owner-occupied homes in $1000s (predictor variable)
- `target`: whether the crime rate is above the median crime rate (1) or not (0) (response variable)

1.2 Descriptive Statistics

Table 3: Summary Statistics

zn	indus	chas	nox	rm	age
Min. : 0.00	Min. : 0.460	Min. :0.00000	Min. :0.3890	Min. :3.863	Min. : 2.90
1st Qu.: 0.00	1st Qu.: 5.145	1st Qu.:0.00000	1st Qu.:0.4480	1st Qu.:5.887	1st Qu.: 43.88
Median : 0.00	Median : 9.690	Median :0.00000	Median :0.5380	Median :6.210	Median : 77.15
Mean : 11.58	Mean :11.105	Mean :0.07082	Mean :0.5543	Mean :6.291	Mean : 68.37
3rd Qu.: 16.25	3rd Qu.:18.100	3rd Qu.:0.00000	3rd Qu.:0.6240	3rd Qu.:6.630	3rd Qu.: 94.10
Max. :100.00	Max. :27.740	Max. :1.00000	Max. :0.8710	Max. :8.780	Max. :100.00

Table 4: Summary Statistics

dis	rad	tax	ptratio	lstat	medv	target
Min. : 1.130	Min. : 1.00	Min. :187.0	Min. :12.6	Min. : 1.730	Min. : 5.00	Min. :0.0000
1st Qu.: 2.101	1st Qu.: 4.00	1st Qu.:281.0	1st Qu.:16.9	1st Qu.: 7.043	1st Qu.:17.02	1st Qu.:0.0000
Median : 3.191	Median : 5.00	Median :334.5	Median :18.9	Median :11.350	Median :21.20	Median :0.0000
Mean : 3.796	Mean : 9.53	Mean :409.5	Mean :18.4	Mean :12.631	Mean :22.59	Mean :0.4914
3rd Qu.: 5.215	3rd Qu.:24.00	3rd Qu.:666.0	3rd Qu.:20.2	3rd Qu.:16.930	3rd Qu.:25.00	3rd Qu.:1.0000
Max. :12.127	Max. :24.00	Max. :711.0	Max. :22.0	Max. :37.970	Max. :50.00	Max. :1.0000

Table 5: Descriptive Statistics

	vars	n	mean	sd	median
zn	1	466	11.5772532	23.3646511	0.00000
indus	2	466	11.1050215	6.8458549	9.69000
chas	3	466	0.0708155	0.2567920	0.00000
nox	4	466	0.5543105	0.1166667	0.53800
rm	5	466	6.2906738	0.7048513	6.21000
age	6	466	68.3675966	28.3213784	77.15000
dis	7	466	3.7956929	2.1069496	3.19095
rad	8	466	9.5300429	8.6859272	5.00000
tax	9	466	409.5021459	167.9000887	334.50000
ptratio	10	466	18.3984979	2.1968447	18.90000
lstat	11	466	12.6314592	7.1018907	11.35000
medv	12	466	22.5892704	9.2396814	21.20000
target	13	466	0.4914163	0.5004636	0.00000

Table 6: Descriptive Statistics

	trimmed	mad	min	max
zn	5.3542781	0.0000000	0.0000	100.0000
indus	10.9082353	9.3403800	0.4600	27.7400
chas	0.0000000	0.0000000	0.0000	1.0000
nox	0.5442684	0.1334340	0.3890	0.8710
rm	6.2570615	0.5166861	3.8630	8.7800
age	70.9553476	30.0226500	2.9000	100.0000
dis	3.5443647	1.9144814	1.1296	12.1265
rad	8.6978610	1.4826000	1.0000	24.0000
tax	401.5080214	104.5233000	187.0000	711.0000
ptratio	18.5970588	1.9273800	12.6000	22.0000
lstat	11.8809626	7.0720020	1.7300	37.9700
medv	21.6304813	6.0045300	5.0000	50.0000
target	0.4893048	0.0000000	0.0000	1.0000

Table 7: Descriptive Statistics

	range	skew	kurtosis	se
zn	100.0000	2.1768152	3.8135765	1.0823466
indus	27.2800	0.2885450	-1.2432132	0.3171281

	range	skew	kurtosis	se
chas	1.0000	3.3354899	9.1451313	0.0118957
nox	0.4820	0.7463281	-0.0357736	0.0054045
rm	4.9170	0.4793202	1.5424378	0.0326516
age	97.1000	-0.5777075	-1.0098814	1.3119625
dis	10.9969	0.9988926	0.4719679	0.0976026
rad	23.0000	1.0102788	-0.8619110	0.4023678
tax	524.0000	0.6593136	-1.1480456	7.7778214
ptratio	9.4000	-0.7542681	-0.4003627	0.1017669
lstat	36.2400	0.9055864	0.5033688	0.3289887
medv	45.0000	1.0766920	1.3737825	0.4280200
target	1.0000	0.0342293	-2.0031131	0.0231835

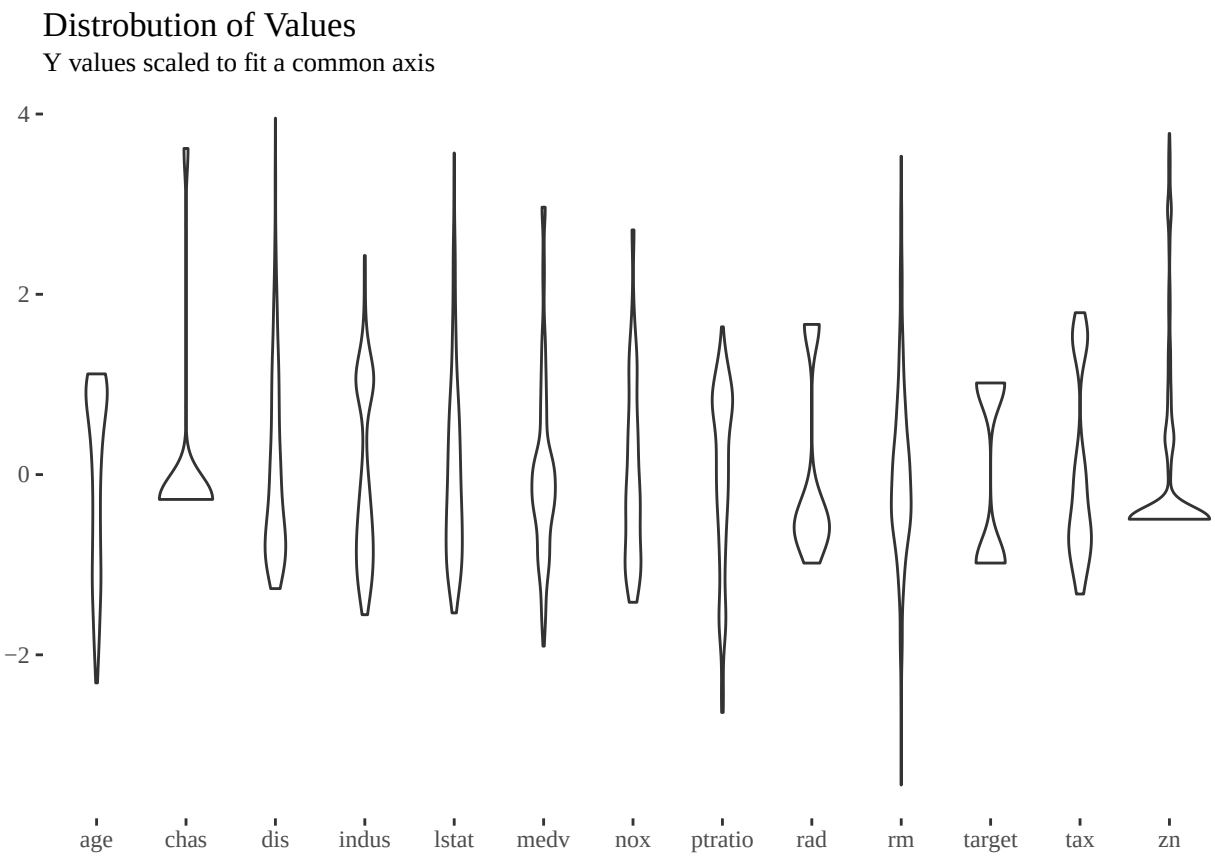
The count of NA values for each variable is given below.

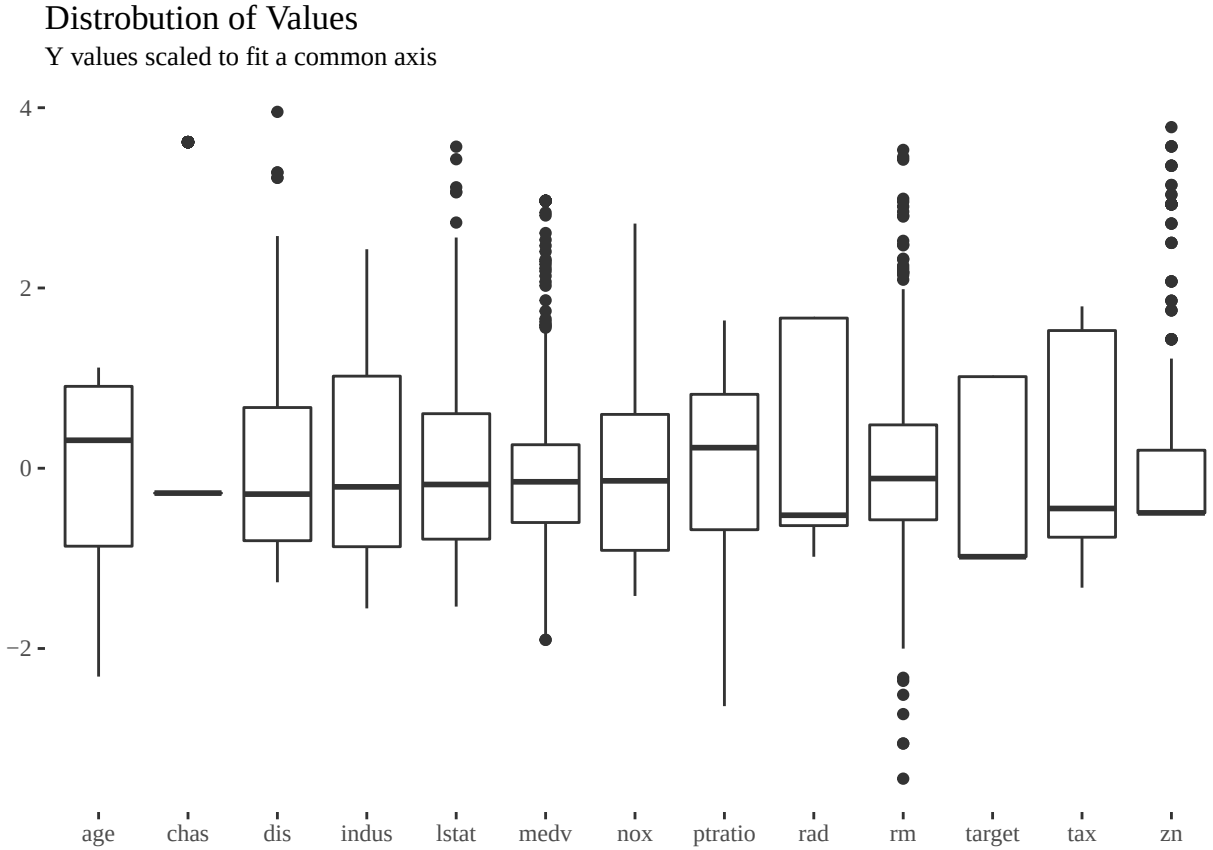
Table 8: Count of NA Values

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	0	0	0	0	0	0	0	0	0	0	0	0

There are no missing values.

1.3 Graphical EDA

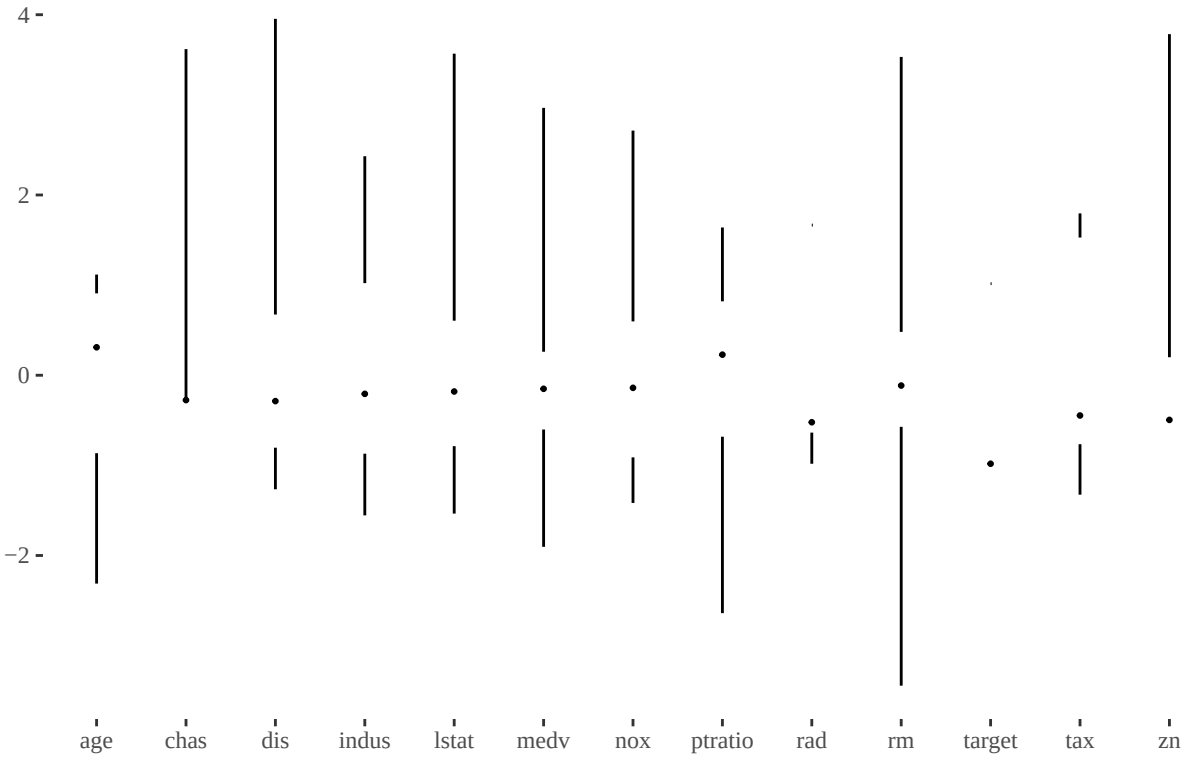


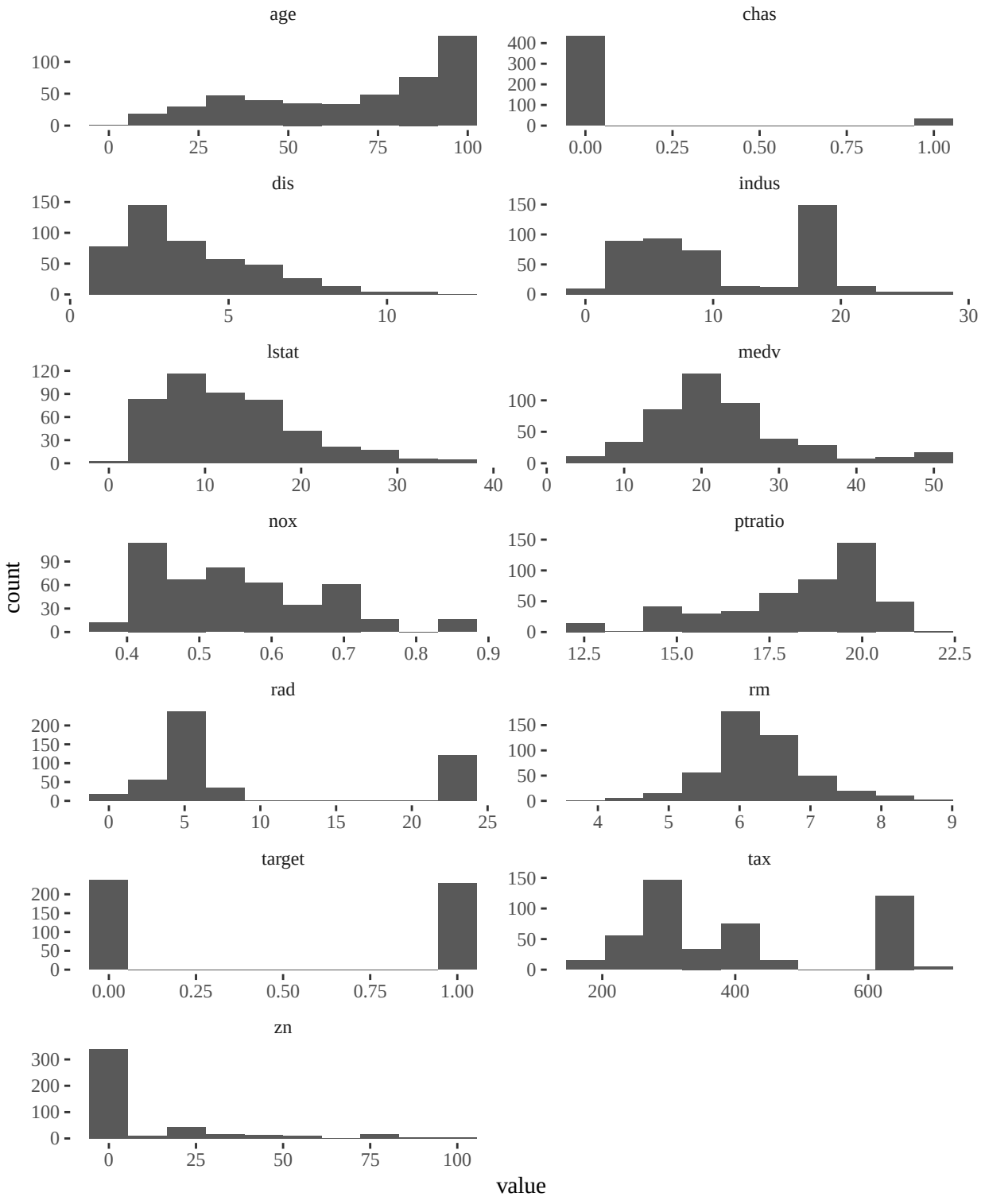


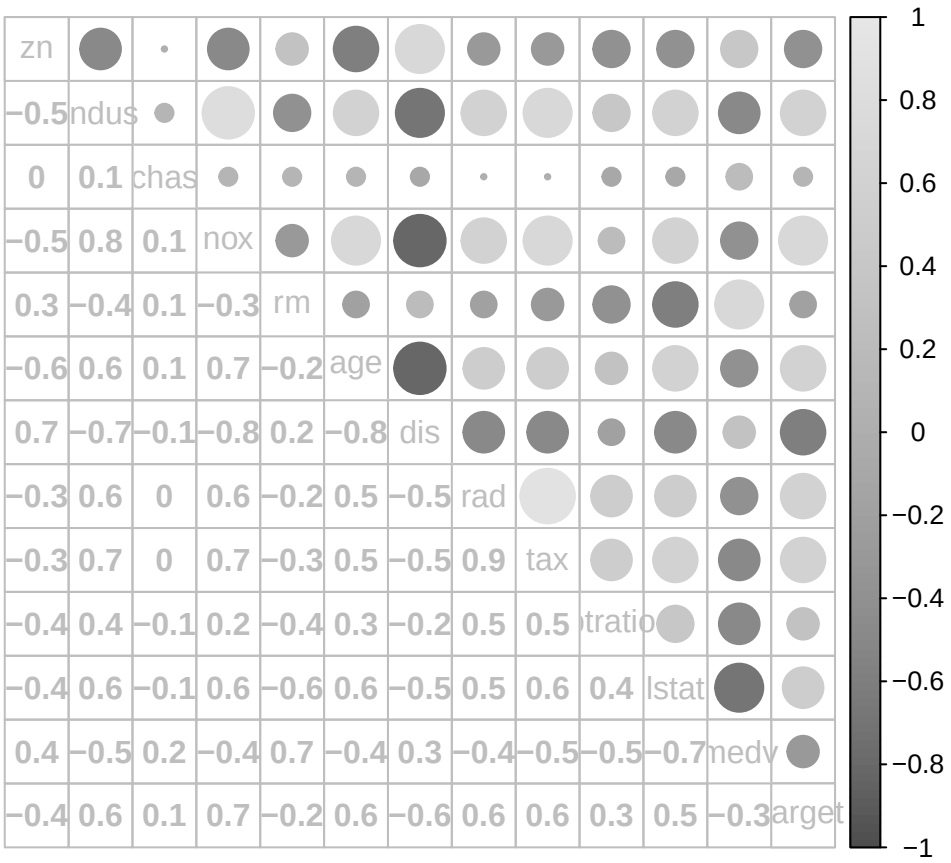
From these two graphs we can see that many of distributions are skewed in one direction or another. It is also interesting to see that the target variable is below zero. This means that the median and mean values are different.

Distrobution of Values

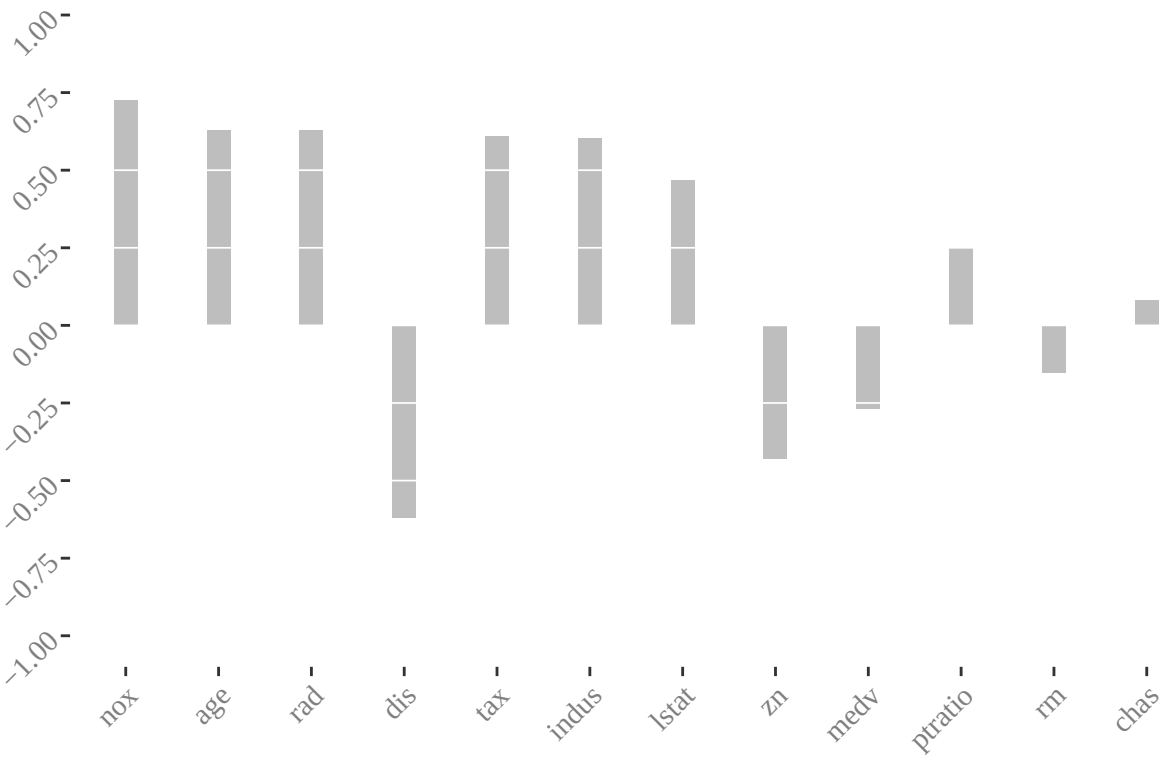
Y values scaled to fit a common axis




age
chas
dis
indus
lstat
medv
nox
ptratio
rad
rm
target
tax
zn
count
value



This correlation plot shows that while there are some highly correlated variables, the most correlated variable is only 0.9, which doesn't raise alarm bells WRT multicoliniarity.



There are some interesting correlations here. Namely indus, nox, age, rad, tax all have a correlation over 0.5. The lowest correlation with the target variable is chas.

We will see which variables play more of a role during our logistic classification, but this gives a good preview.

2 Data Preperation

Transformation of variables is less needed for logistic regresion because normallacy is not a requriment. However we will transform some variables, which have a large skewness, to see if they aid in prediction.

The other reason to transform variables is to account for interactions between variables. Logistic regression is 'linear' and an exhaustive search of all possible combinations/polynomials would be difficult even if we limited them to three degrees each. Instead, I suggest a test where we fit an extremely non-linear model `KNN` on the data and compare the ROC to the ROC from a simple logistic regression. If there is little difference, it can be safe to assume that the underlying relationship is linear.

I used this example to create a KNN model.

```
##
## AboveMed BelowMed
## 49.14286 50.85714
```

```
## Created from 350 samples and 12 variables
##
## Pre-processing:
##   - centered (12)
##   - ignored (0)
##   - scaled (12)
```

The AUC for the KNN Model is 0.9565288 and for the logistic regression model it was 0.9737623. This leads me to believe that there is little need to transformation.

2.1 Transformed Skewed Variables

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	19.58	0	0.605	7.929	96.2	2.0459	1.609438	5.998937	14.7	3.70	50.0	1
0	19.58	1	0.871	5.403	100.0	1.3216	1.609438	5.998937	14.7	26.82	13.4	1
0	18.10	0	0.740	6.485	100.0	1.9784	3.178054	6.501290	20.2	18.85	15.4	1
30	4.93	0	0.428	6.393	7.8	7.0355	1.791759	5.703782	16.6	5.19	23.7	0
0	2.46	0	0.488	7.155	92.2	2.7006	1.098612	5.262690	17.8	4.82	37.9	0
0	8.56	0	0.520	6.781	71.3	2.8561	1.609438	5.950643	20.9	7.67	26.5	0

3 Build Models

This project will focus on automated variable selection. New techniques will be compared to the basic logistic regression.

3.1 Baisic Logistic Regression

The basic logistic regression gives a summary of:

% latex table generated in R 3.4.0 by xtable 1.8-2 package %

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-1.6014	0.3595	-4.45	0.0000
zn	-0.0010	0.0009	-1.02	0.3064
indus	0.0031	0.0043	0.73	0.4664
chas	0.0060	0.0588	0.10	0.9190
nox	1.9722	0.2633	7.49	0.0000
rm	0.0250	0.0315	0.79	0.4282
age	0.0032	0.0009	3.51	0.0005
dis	0.0125	0.0141	0.89	0.3758
rad	0.0207	0.0043	4.77	0.0000
tax	-0.0003	0.0003	-1.07	0.2874
ptratio	0.0115	0.0093	1.23	0.2180
lstat	0.0045	0.0039	1.16	0.2469
medv	0.0089	0.0030	2.98	0.0031

Here we have the output of all provided variables without any transformation.

% latex table generated in R 3.4.0 by xtable 1.8-2 package %

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-34.9015	8.3985	-4.16	0.0000
zn	-0.0645	0.0337	-1.91	0.0558
indus	-0.0633	0.0496	-1.28	0.2021
chas	1.0045	0.7440	1.35	0.1769
nox	47.3768	7.7812	6.09	0.0000
rm	-0.4729	0.7148	-0.66	0.5082
age	0.0342	0.0136	2.51	0.0122
dis	0.7279	0.2267	3.21	0.0013
rad	3.1881	0.7623	4.18	0.0000
tax	-1.6370	1.1581	-1.41	0.1575
ptratio	0.4167	0.1256	3.32	0.0009
lstat	0.0353	0.0541	0.65	0.5136
medv	0.1787	0.0684	2.61	0.0090

3.2 Stepwise Selection on Baisic Model

If we do a step wise selection to find the variables that limit the scope but still provide excellent performance we get:

% latex table generated in R 3.4.0 by xtable 1.8-2 package %

	Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1				453.00	44.11	251.85
2	- chas	1.00	0.00	454.00	44.11	249.86
3	- indus	1.00	0.05	455.00	44.16	248.43
4	- rm	1.00	0.05	456.00	44.22	246.97
5	- dis	1.00	0.06	457.00	44.27	245.57
6	- zn	1.00	0.06	458.00	44.34	244.23
7	- lstat	1.00	0.08	459.00	44.42	243.07
8	- tax	1.00	0.13	460.00	44.54	242.39

(Intercept) nox age rad ptratio -1.412836094 1.956694224 0.003531713 0.017106647 0.012716341 medv 0.008021190 target ~ nox + age + rad + ptratio + medv % latex table generated in R 3.4.0 by xtable 1.8-2 package %

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-24.9365	3.6834	-6.77	0.0000
nox	25.3348	4.0841	6.20	0.0000
age	0.0194	0.0093	2.08	0.0371
rad	0.5126	0.1148	4.46	0.0000
ptratio	0.2742	0.0987	2.78	0.0055
medv	0.0854	0.0280	3.05	0.0023

This presentation offers an interesting critique of step wise selection and some of the issue that make it less ideal.

3.3 BestGLM Model Selection

This model selection using the steps algorithm selects significantly fewer variables.

```
## Morgan-Tatar search since family is non-gaussian.
```

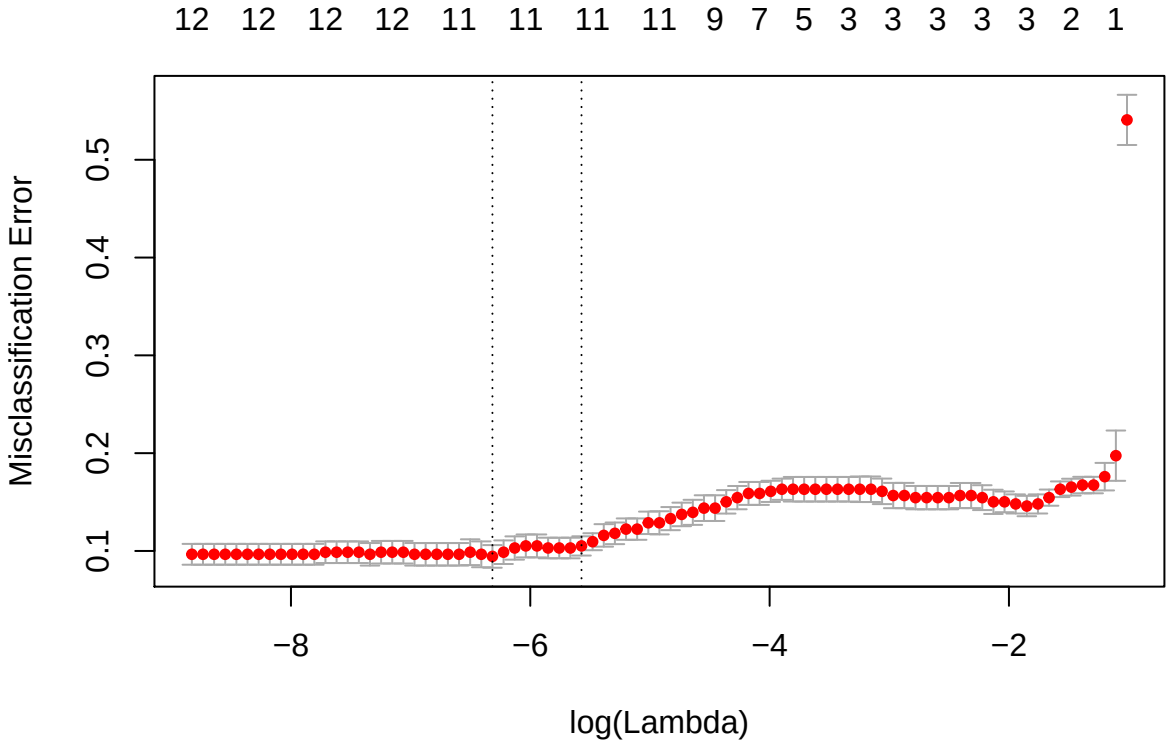
```
## BIC
## BICq equivalent for q in (0.0115719591613259, 0.54658790604255)
## Best Model:
```

```
##                  Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept) -19.867421883 2.368324820 -8.388808 4.910965e-17
## nox          35.633515210 4.523677298  7.877113 3.350331e-15
## rad           0.637642983 0.119444473  5.338405 9.376777e-08
## tax          -0.008145957 0.002332059 -3.493032 4.775700e-04
```

3.4 LASSO Regression

Looking at lasso logistic regression might give us a better model selection and coefficient values. Below is the results.

```
## Warning in aucDF$lasso.prob <- predict(lasso.model, type = "response", newx
## = X, : Coercing LHS to a list
```




```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -25.016113852
## zn           -0.025749251
## indus        -0.041630599
## chas          0.755989557
## nox          30.620136853
## rm            .
## age           0.019060550
## dis           0.278520115
## rad           0.330360865
## tax          -0.002594363
## ptratio       0.198760128
## lstat         0.013804668
## medv          0.073090589
```

3.5 Regular logostic without LASSO dropped Variable

This models coefficients deviate significantly from a normal `glm` model that excludes the one variable dropped. This is because LASSO penalizes large coefficients. For example, `glm` model excluding `rm` is:

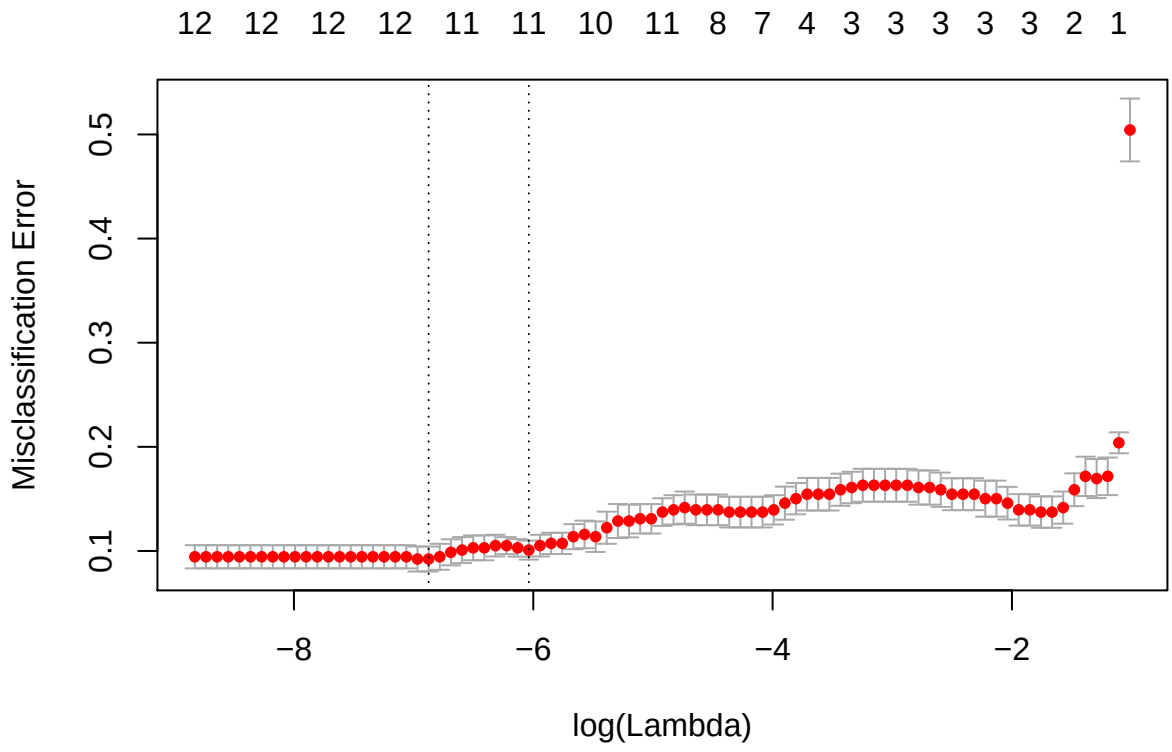
% latex table generated in R 3.4.0 by xtable 1.8-2 package %

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-41.6494	6.5038	-6.40	0.0000
zn	-0.0684	0.0344	-1.99	0.0467
indus	-0.0634	0.0475	-1.34	0.1819
chas	0.9311	0.7575	1.23	0.2190
nox	48.0922	7.7282	6.22	0.0000
age	0.0284	0.0116	2.44	0.0146
dis	0.6915	0.2183	3.17	0.0015
rad	0.6439	0.1592	4.04	0.0001
tax	-0.0063	0.0029	-2.15	0.0314
ptratio	0.3618	0.1139	3.18	0.0015
lstat	0.0625	0.0496	1.26	0.2074
medv	0.1379	0.0414	3.33	0.0009

This is interesting because we can see how different the coefficents are even though it has the same variables.

3.6 Lasso with scaled variable

```
## Warning in aucDF$lasso.prob <- predict(lasso.model, type = "response", newx
## = logX, : Coercing LHS to a list
```




```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -27.36650471
```

```
## zn          -0.03601799
## indus       -0.04161848
## chas         0.82316880
## nox         35.01440948
## rm           .
## age          0.02250038
## dis          0.39362984
## rad          2.47159307
## tax         -1.00392389
## ptratio      0.28590641
## lstat        0.01567105
## medv         0.09466393
```

4 Chose a Model

Model selection in the previous section used a variety of different methods ranging from none to BIC to AIC. As such it is unfair to select a criteria that we have used already.

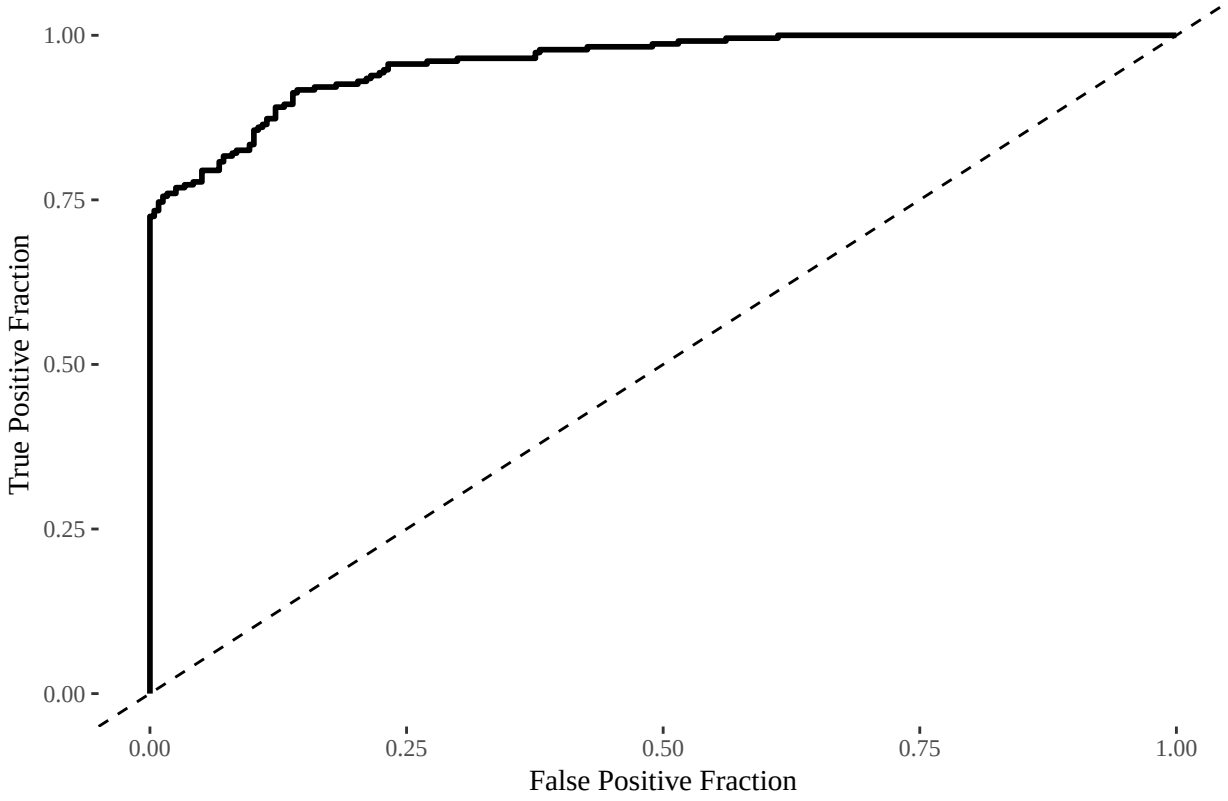
I've chosen to use AUC because it is easily understood and provides a nice visual way to differentiate model performance.

4.1 Baisic GLM Model

We see this model has good performance, especially for a dataset that has roughly equal numbers of positive and negative examples.

4.2 GLM Model Evaluation

```
## [[1]]
```

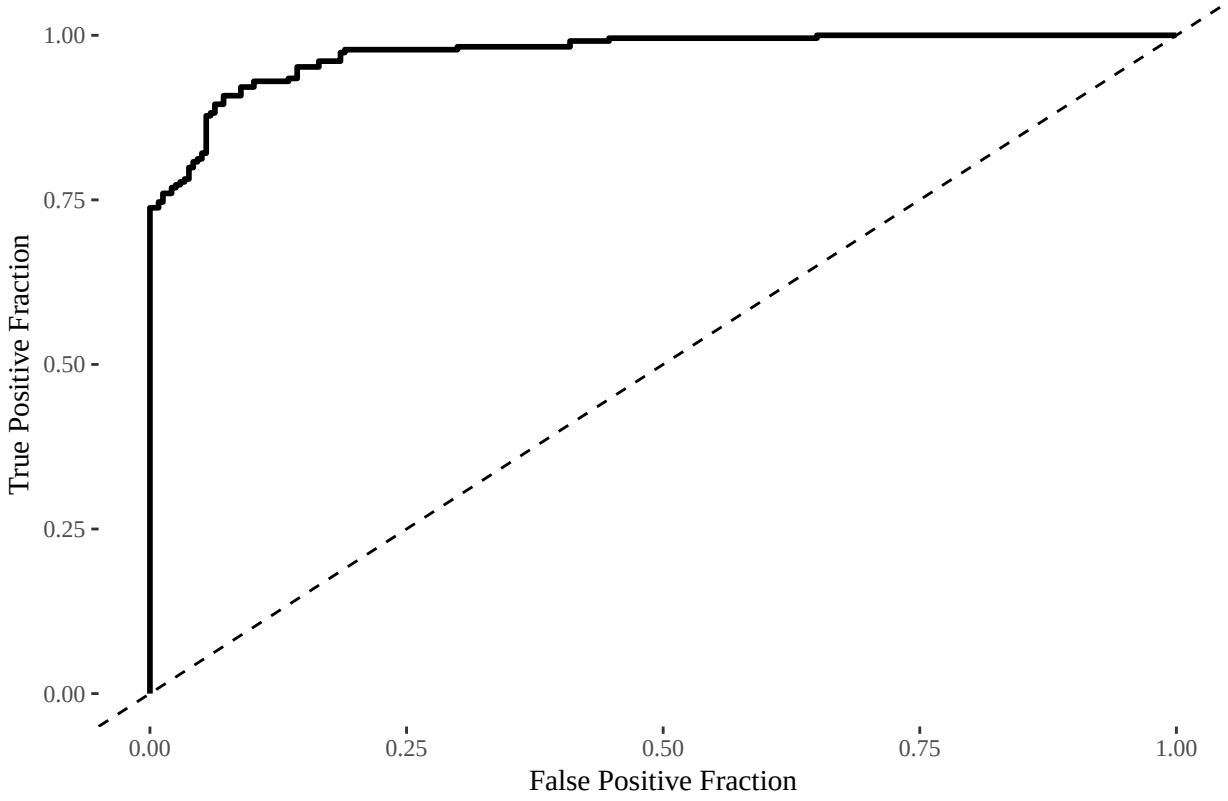



```
##
## [[2]]
## Area under the curve: 0.9582
```

4.3 Scaled GLM Model

```
## [[1]]
```

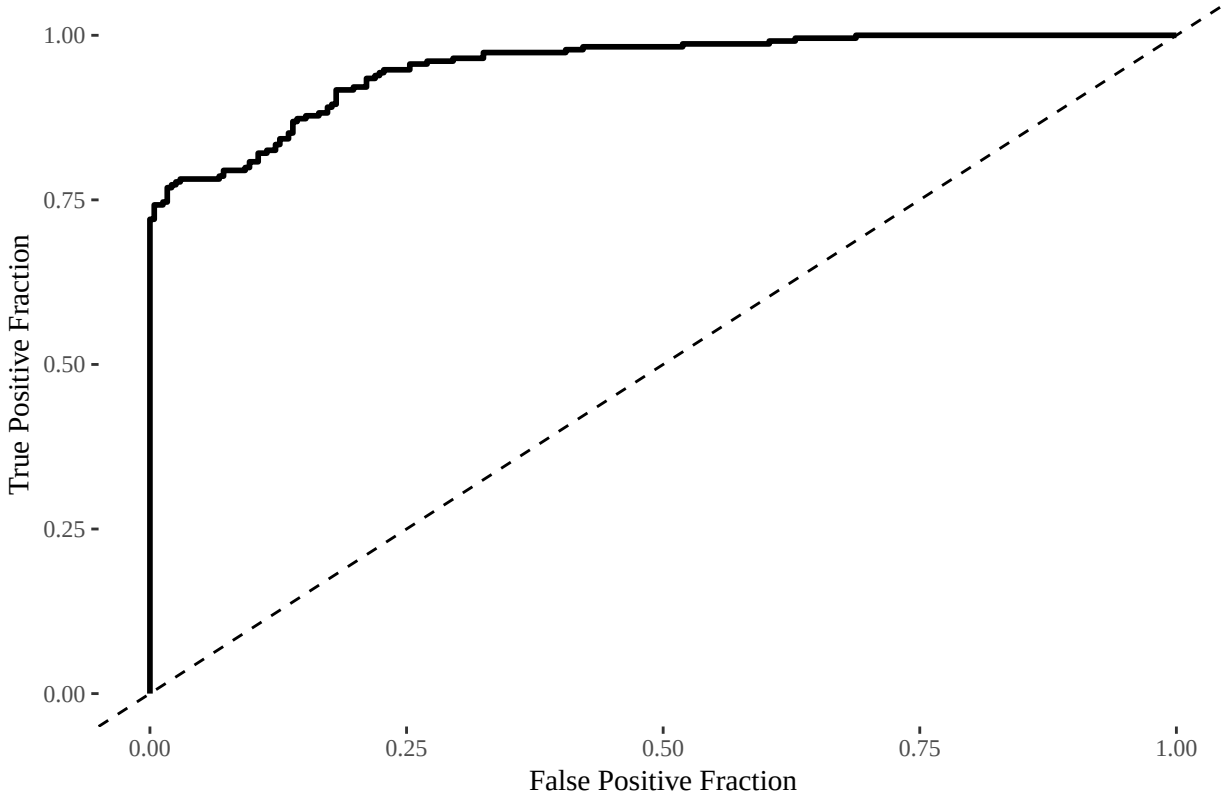
AUC Graph for Sclaed GLM




```
##
## [[2]]
## Area under the curve: 0.9729
```

4.4 STEPWISE AIC MODEL

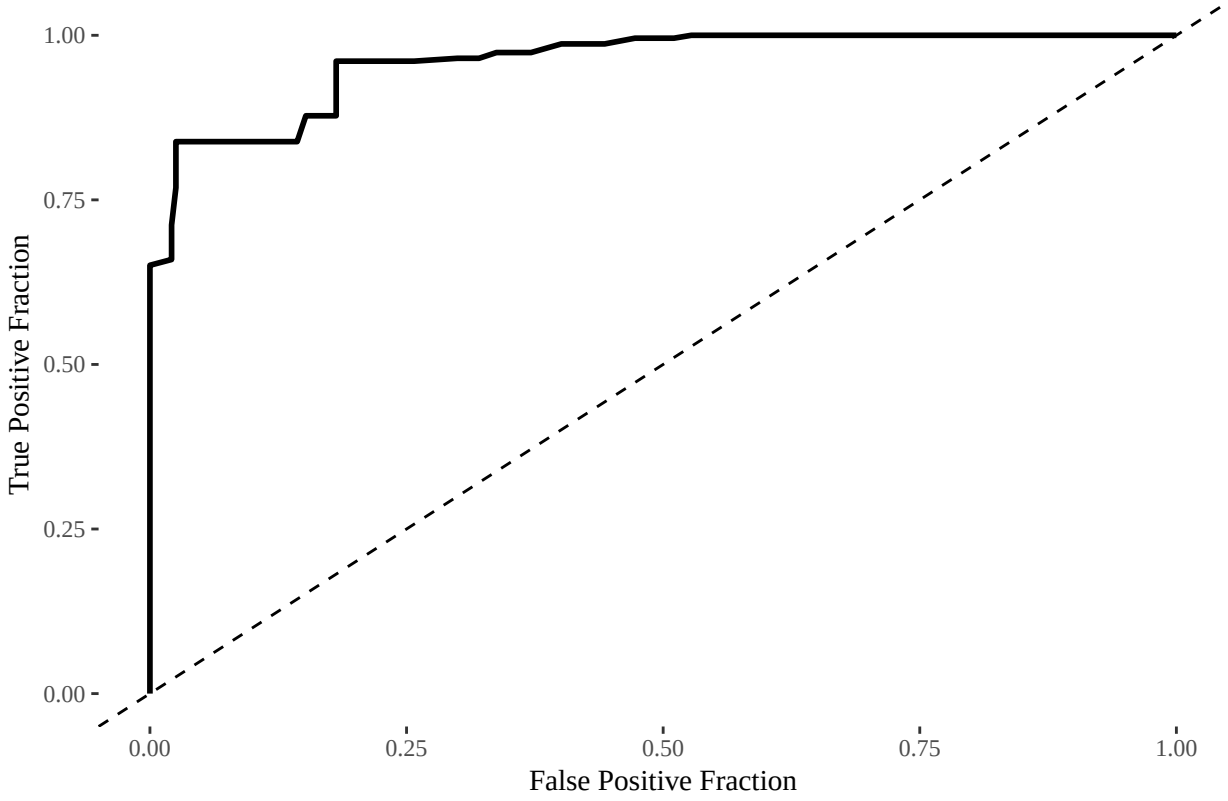
```
## [[1]]
```




```
##
## [[2]]
## Area under the curve: 0.9527
```

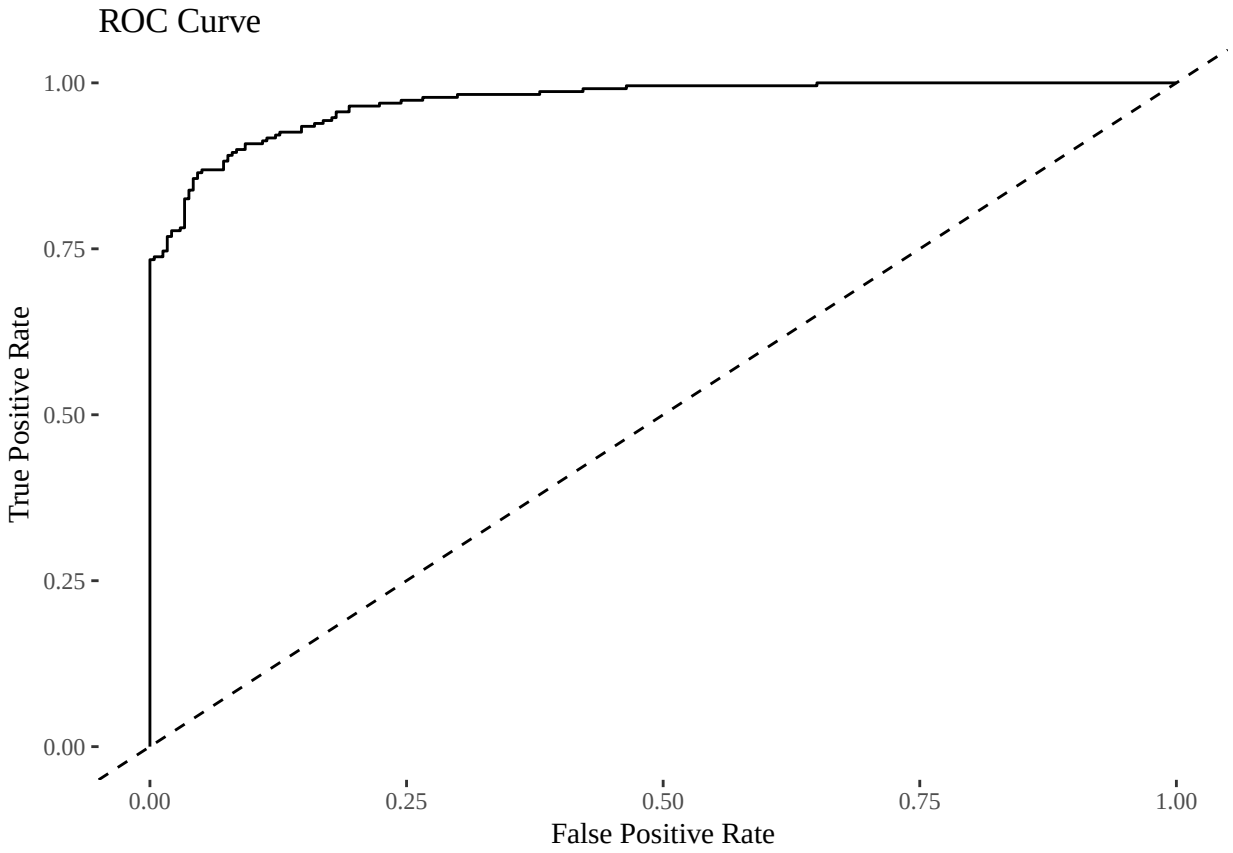
4.5 BestGLM Model

```
## [[1]]
```



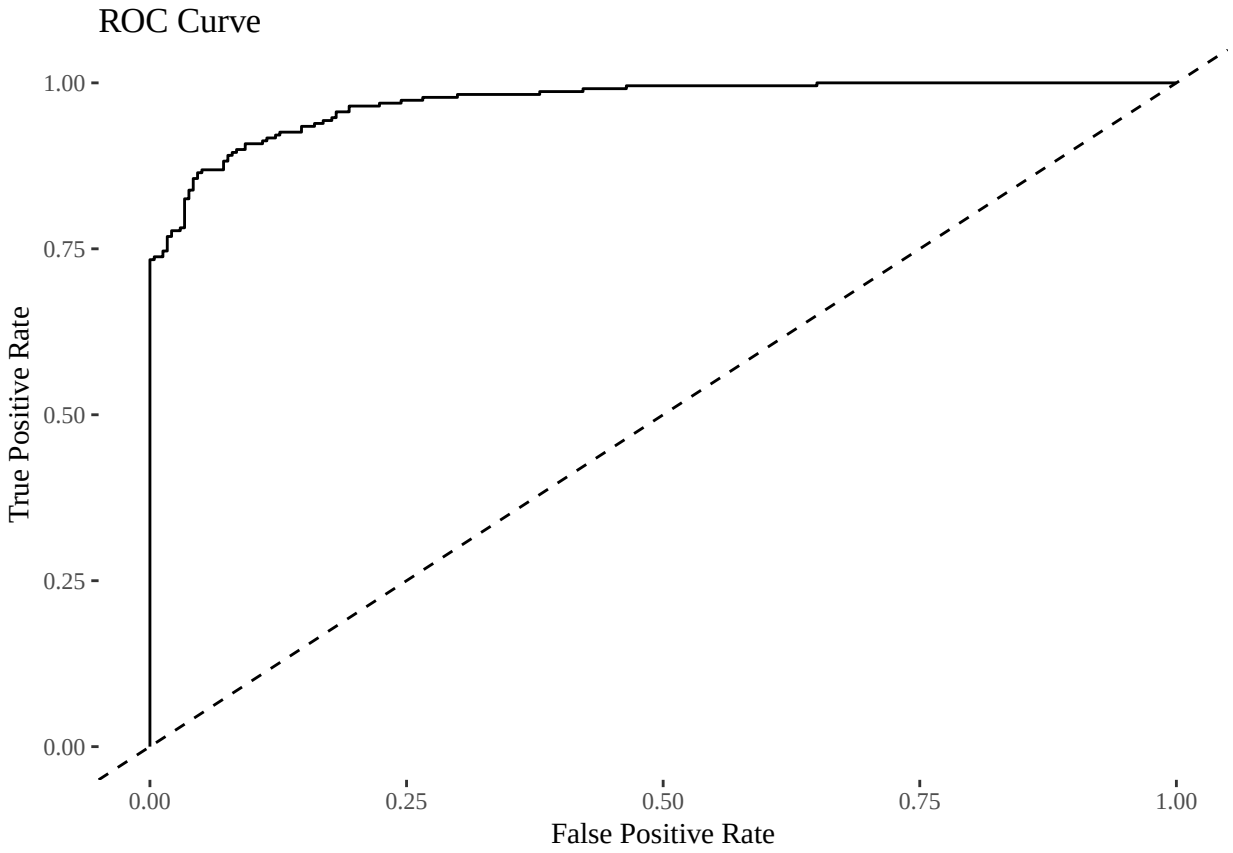

```
##
## [[2]]
## Area under the curve: 0.9594
```

4.6 Lasso Model




```
## [[1]]
## [1] 0.9709985
```

4.7 Scaled LASSO




```
## [[1]]
## [1] 0.9709985
```

4.8 Which model?

All models performed well with no model below 0.95. The best model was the LASSO model. This has advantages over the step wise selection as the PDF linked above goes over. This means we can confidently select this mode.

4.9 Prediction

The predictions for the test set are given below. The first column corresponds to the probabilities and the second column corresponds to the actual prediction (the rounded probabilities).

Predicted_Probabilities	Predicted_Outcome
0.0963140	0
0.5356428	1
0.5936776	1
0.4598782	0
0.1547428	0
0.2045212	0
0.2436646	0
0.0436209	0

Predicted_Probabilities	Predicted_Outcome
0.0276064	0
0.0185725	0
0.4210005	0
0.3801533	0
0.7748370	1
0.6441183	1
0.5754163	1
0.3168680	0
0.3026643	0
0.8173990	1
0.0721549	0
0.0004601	0
0.0007037	0
0.1341159	0
0.2420760	0
0.1925227	0
0.1680612	0
0.4552867	0
0.0060445	0
0.9999766	1
0.9999699	1
0.9971125	1
0.9999687	1
0.9999764	1
0.9999713	1
0.9999855	1
0.9999749	1
0.9999491	1
0.9999619	1
0.9997567	1
0.7232785	1
0.4978899	0

5 Apendix

```
knitr::opts_chunk$set(echo = FALSE)
# Libraries
##################
library(MASS)
library(car)
library(leaps)
library(tidyverse)
library(knitr)
library(kableExtra)
library(psych)
library(ggthemes)
library(corrplot)
library(glmnet)
library(bestglm)
library(xtable)
```

```r
options(xtable.floating = FALSE)
options(xtable.timestamp = "")
####################
# Loading the data
df <- read_csv('/Users/kailukowiak/DATA621/Assignments/Assignment3/crime-training-data_modified.csv')
df %>% sample_n(6) %>% kable(caption = 'Sample of Values for the Training Set')
testDF <- read_csv('/Users/kailukowiak/DATA621/Assignments/Assignment3/crime-evaluation-data_modified.cs
testDF %>% sample_n(6) %>% kable(caption = 'Sample of Values for the Test Set')
##############################
# Summary Tables
SumTab <- summary(df)
SumTab1 <- SumTab[, 1:6]
SumTab2 <- SumTab[, 7:13]
kable(SumTab1, caption = 'Summary Statistics')
kable(SumTab2, caption = 'Summary Statistics')
####################
dis <- describe(df)
dis[, 1:5] %>% kable(caption = 'Descriptive Statistics')
dis[, 6:9] %>% kable(caption = 'Descriptive Statistics')
dis[, 10:13] %>% kable(caption = 'Descriptive Statistics')
map(df, ~sum(is.na(.))) %>% t() %>% kable(caption = 'Count of NA Values')
df %>%
  scale() %>%
  as_tibble() %>%
  gather() %>%
  ggplot(aes(x = key, y = value)) +
  theme_tufte() +
  geom_violin()+
  #geom_tufteboxplot(outlier.colour="black")+
  theme(axis.title=element_blank()) +
  ylab('Scaled Values')+
  xlab('Variable')+
  ggtitle('Distrobution of Values', subtitle = 'Y values scaled to fit a common axis')
df %>%
  scale() %>%
  as_tibble() %>%
  gather() %>%
  ggplot(aes(x = key, y = value)) +
  # geom_violin()+
  # geom_tufteboxplot(outlier.colour="black", outlier.shape = 22)+
  geom_boxplot()+
  theme_tufte() +
  theme(axis.title=element_blank()) +
  ylab('Scaled Values')+
  xlab('Variable')+
  ggtitle('Distrobution of Values', subtitle = 'Y values scaled to fit a common axis')
df %>%
  scale() %>%
  as_tibble() %>%
  gather() %>%
  ggplot(aes(x = key, y = value)) +
  theme_tufte() +
  # geom_violin()+
```

```
  geom_tufteboxplot(outlier.colour="black")+
  theme(axis.title=element_blank()) +
  ylab('Scaled Values')+
  xlab('Variable')+
  ggtitle('Distrobution of Values', subtitle = 'Y values scaled to fit a common axis')
ggplot(data = gather(df), mapping = aes(x = value)) +
  geom_histogram(bins = 10) + facet_wrap(~key, ncol = 2, scales = 'free') +
  theme_tufte()
corr <- round(cor(df), 1)
corrplot.mixed(corr, lower.col = 'grey', upper.col = gray.colors(100), tl.col='grey')
corr2 <- corr
diag(corr2) <- 0
ind <- which(corr2 == max(corr2), arr.ind = TRUE)

maxCorr <- corr[ind][1]
corrDF <- cor(x = df[, 1:12], y = df$target) %>%
  as_tibble() %>%
  rename(Correlation = V1) %>%
  mutate(VarNames = names(df[, 1:12]))

ggplot(corrDF, aes(x= reorder(VarNames, -abs(Correlation)), y=Correlation)) +
  ggtitle('Correlation of Variables with Target') +
  theme_tufte(base_size=14, ticks=T) +
  geom_bar(width=0.25, fill="gray", stat = "identity") +
  theme(axis.title=element_blank()) +
  scale_y_continuous(breaks=seq(-1, 1, 0.25)) +
  geom_hline(yintercept=seq(-1, 1, 0.25), col="white", lwd=.3) +
  theme(axis.text = element_text(angle = 45, hjust = 1, colour = 'grey50'))

interestingCorr <- corrDF %>% filter(Correlation >= 0.5)
library(ISLR)
library(caret)
library(ROCR)
library(plotROC)
library(pROC)
set.seed(300)
#Spliting data as training and test set. Using createDataPartition() function from caret
df1 = df
df1$target <- ifelse(df1$target == 1, 'AboveMed', 'BelowMed')
indxTrain <- createDataPartition(y = df1$target,p = 0.75,list = FALSE)
training <- df1[indxTrain,]
testing <- df1[-indxTrain,]

#Checking distibution in origanl data and partitioned data
prop.table(table(training$target)) * 100



trainX <- training[,names(training) != "target"] # Make this target
preProcValues <- preProcess(x = trainX,method = c("center", "scale"))
preProcValues

training$target <- as.factor(training$target)
```

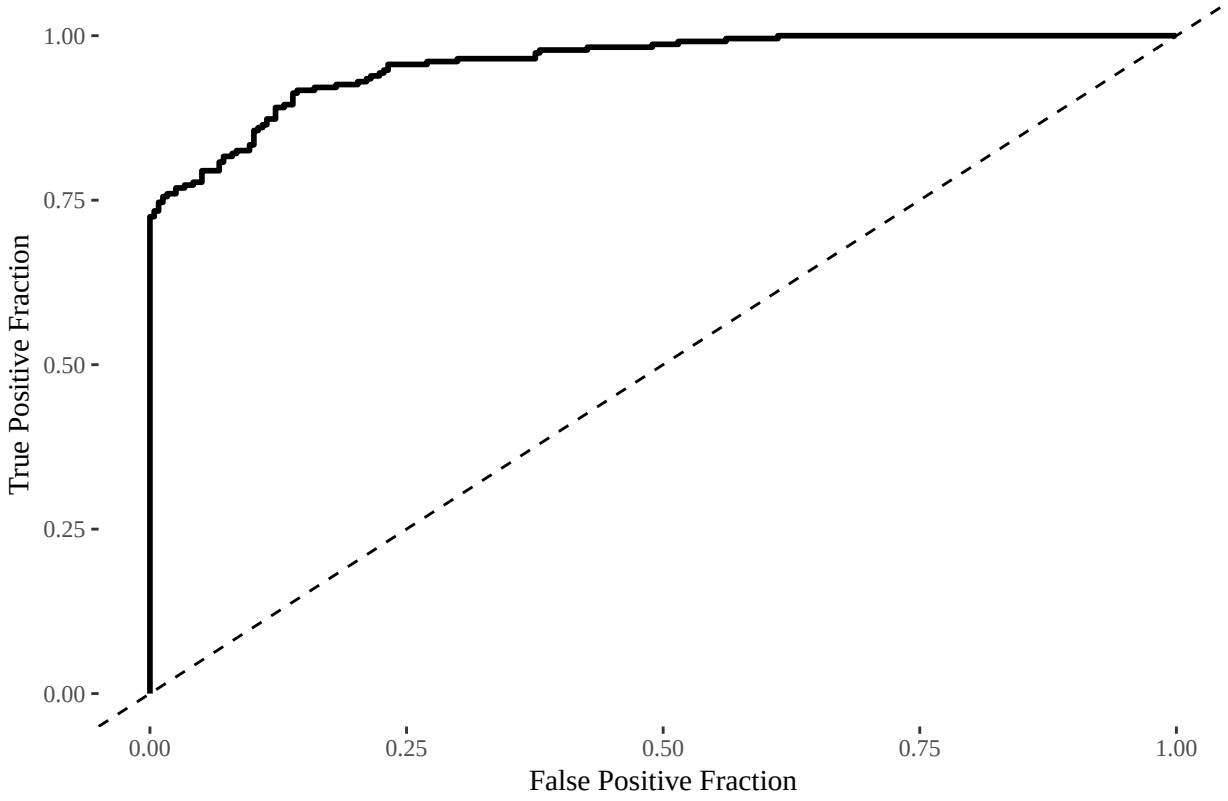
```r
set.seed(400)
ctrl <- trainControl(method="repeatedcv",repeats = 3,classProbs=TRUE,summaryFunction = twoClassSummary)
#ctrl <- trainControl(method="repeatedcv",repeats = 3) #,classProbs=TRUE,summaryFunction = twoClassSumm
knnFit <- train(target ~ ., data = training, method = "knn", trControl = ctrl, preProcess = c("center",

#Output of kNN fit
knnAUC <- knnFit$results[1,2]

mod1 <- glm(target~., data = df, family = 'binomial')
prob = predict(mod1,type = c("response"))
g <- roc(target ~ prob, data = df)
logAUC <- g$auc
logDF <- df
logDF$rad <- log(logDF$rad)
logDF$tax <- log(logDF$tax)

logDF %>% head() %>% kable()
mod1 <- glm(target ~ ., data = df)
#summary(mod1)
library(xtable)
xtable(mod1)
logData <- glm(target~., family = binomial(), data = logDF)
xtable(logData)
step <- stepAIC(mod1, direction="both", trace = FALSE)
xtable(step$anova)
step$coefficients
formulaLength <- length(step$coefficients)
formulaNames <- names(step$coefficients)[2:formulaLength]
stepFormula <- as.formula(paste("target~", paste(formulaNames, collapse="+")))
stepFormula
stepModel <- glm(formula = stepFormula, family = binomial, data = df)
xtable(stepModel)
df1 <- df
df1 <- dplyr::rename(df1, y = target)
df1$y <- as.factor(df1$y)
df1 <- data.frame(df1)
BestGLMModel <- bestglm(df1, family = binomial)
#xtable(BestGLMModel)
BestGLMModel
X <- df %>% dplyr::select(-target)
X <- data.matrix(X)
#X <- as.matrix(X, ncol=12)
y <- as.factor(df$target)
fit = glmnet(X, y, family = "binomial")
library(ROCR)
aucDF <- X
lasso.model = cv.glmnet(X, y, family = "binomial", type.measure = 'class')

aucDF$lasso.prob <- predict(lasso.model, type="response", newx = X, s = 'lambda.1se')
pred <- prediction(aucDF$lasso.prob, y)


cvfit = cv.glmnet(X, y, family = "binomial", type.measure = "class")
```

```r
plot(cvfit)
coef(cvfit, s = "lambda.1se")
mod3 <- glm(target ~ . -rm, family = 'binomial', data = df)
xtable(mod3)
logX <- logDF %>% dplyr::select(-target)
logX <- data.matrix(logX)
y <- as.factor(logDF$target)
fit = glmnet(logX, y, family = "binomial")
library(ROCR)
aucDF <- logX
lasso.model = cv.glmnet(logX, y, family = "binomial", type.measure = 'class')

aucDF$lasso.prob <- predict(lasso.model, type="response", newx = logX, s = 'lambda.1se')
pred <- prediction(aucDF$lasso.prob, y)


cvfitLOG = cv.glmnet(logX, y, family = "binomial", type.measure = "class")
plot(cvfitLOG)
coef(cvfitLOG, s = "lambda.1se")
AUC <- function(df, mod, modelName){
  library(plotROC)
  library(pROC)
  prob = predict(mod,type = c("response"))
  df$prob=prob
  p = ggplot(df, aes(d = target, m = prob)) +
    geom_roc(n.cuts = 0) +
    ggtitle(paste('AUC Graph for', modelName)) +
    xlab("False Positive Fraction") +
    ylab('True Positive Fraction') +
    geom_abline(linetype = 'dashed') +
    theme_tufte()

  g <- roc(target ~ prob, data = df)
  return(list(p, g$auc))
}
AUC(df, mod1, 'Baisic GLM Model')
AUC(df, logData, 'Sclaed GLM')
AUC(df, step, 'Step AIC Model')

AUC(df, BestGLMModel$BestModel, 'Model Best')
fittedGLMcv <- predict(cvfit, X, s = "lambda.1se", type = "class")

perf <- performance(pred,"tpr","fpr")
auc <-  performance(pred,"auc") # shows calculated AUC for model
auc <- auc@y.values

roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    #geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
```

```
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate') + theme_tufte()
auc
fittedGLMcv <- predict(cvfitLOG, X, s = "lambda.1se", type = "class")

perf <- performance(pred,"tpr","fpr")
auc <-  performance(pred,"auc") # shows calculated AUC for model
auc <- auc@y.values

roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values))

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    #geom_ribbon(alpha=0.2) +
    geom_line(aes(y=tpr)) +
    geom_abline(slope=1, intercept=0, linetype='dashed') +
    ggtitle("ROC Curve") +
    ylab('True Positive Rate') +
    xlab('False Positive Rate') + theme_tufte()
auc
testMat <- data.matrix(testDF)
PredictedProbabilities <- predict(cvfit,type = "response", newx =  testMat)
PredictedValues <- round(PredictedProbabilities)
predDF <- data.frame(PredictedProbabilities, PredictedValues)
colnames(predDF) <- c('Predicted_Probabilities', 'Predicted_Outcome')
predDF %>% kable()
# Test to see if logged values are better

nonLog <- mod1

logDF <- df
#logDF$zn <- log(logDF$zn)
#logDF$chas <- log(logDF$chas)
logDF$rad <- log(logDF$rad)
logDF$tax <- log(logDF$tax)
logData <- glm(target~., family = binomial(), data = logDF)

AUC(df, nonLog, 'Non log values')
AUC(logDF, logData, 'logged values')
```

```
## [[1]]
```

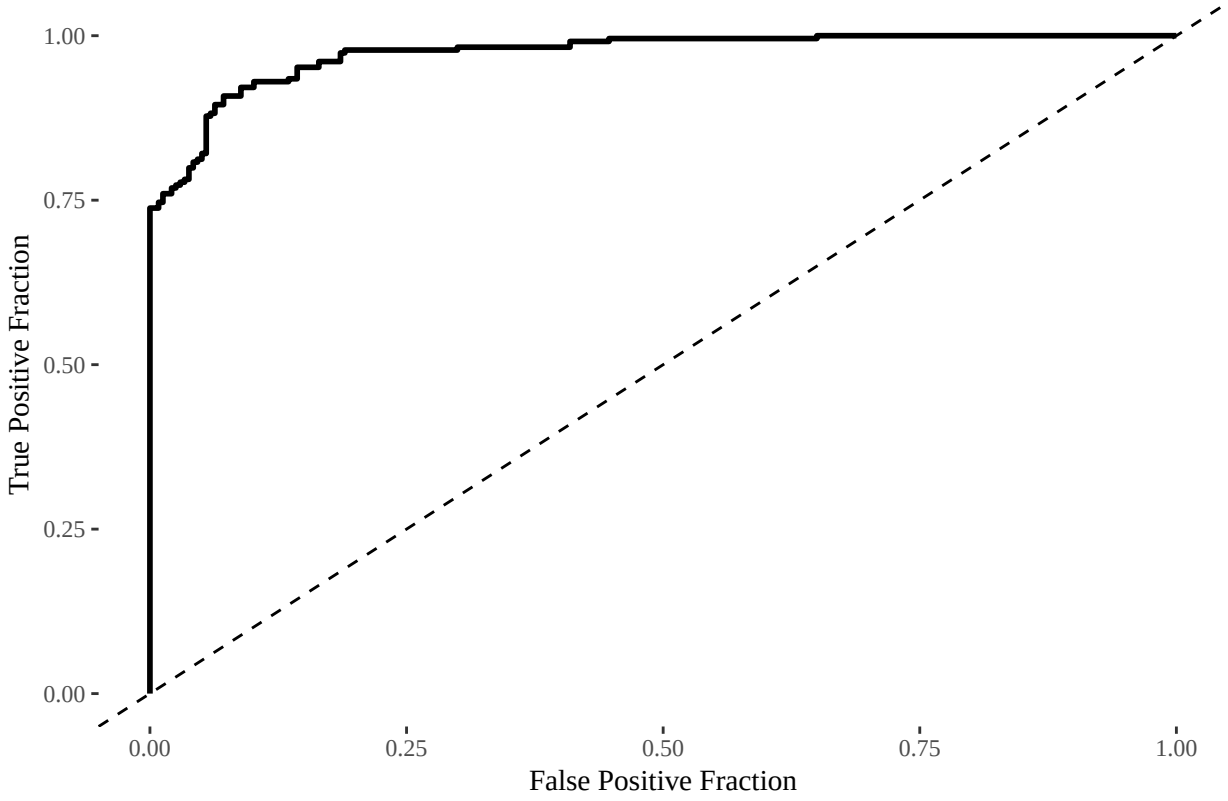
AUC Graph for Non log values




```
##
## [[2]]
## Area under the curve: 0.9582
```

```
## [[1]]
```

AUC Graph for logged values




```
##
## [[2]]
## Area under the curve: 0.9729
```