BDA Coursework 1

MSc Data Science

Mark Rotchell, 13181875

1. Statistical Learning Methods

1.a) Large p, small n

Flexible method worse. Increasing flexibility increases the likelihood of overfitting. This problem is exacerbated when there are few observations. The likelihood of one or more predictors with no relationship to the dependent variable passing tests of significance by coincidence increases as the number of predictors increases and the number of observations decreases. Flexible methods also tend to produce less interpretable results.

1.b) Small p, large n

Flexible method better. Increasing the number of observations increases confidence that a model reflects the underlying relationship, rather than random noise. In particular, the average contribution of irreducible error to the dependent variable tends to zero with increasing sample size. This confidence allows for searching a wider gamut of potential parametrisations without fear of overfitting. Having fewer predictors reduces the likelihood of finding, and subsequently including, predictors which correlate with the random errors by sheer chance. That said, if interpretability were the goal, rather than predictive accuracy, then a less flexible method may be more appropriate.

1.c) Highly non-linear relationship

Flexible method better. Non-linear relationships require more parameters to describe. Such models would generally require a more flexible learning method in order to be found, though a big sample size would be required to provide high predictive accuracy. Again, a more flexible method may prove less interpretable than a simpler method.

1.d) High error-term standard deviation

Flexible method worse. The more of each observation that can be attributed to random error, the more likely it is that over-fitting will occur. Flexible methods are more prone to overfitting, so are best avoided in this instance unless the sample size is also extremely large. A simpler method has the added benefit of being more interpretable.

2. Bayes' Rule

```
Temperature =  c('hot', 'hot', 'hot', 'cool', 'hot', 'hot', 'cool', 'cool', 'cool', 'hot',
                 'cool', 'hot', 'hot', 'hot', 'cool', 'cool', 'hot', 'cool', 'hot', 'hot')
Play_Golf = c('no', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no', 'yes',
              'yes', 'yes', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'no', 'yes')
knitr::kable(addmargins(table(paste('Temperature = ',Temperature),
                              paste('Play Golf = ',Play_Golf))))
```

	Play Golf = no	Play Golf = yes	Sum
Temperature = cool	3	5	8
Temperature = hot	7	5	12
Sum	10	10	20

P(Play Golf = yes) = 10 / 20 = 0.5

P(Play Golf = no) = 10 / 20 = 0.5

P(Temperature = hot) = 12 / 20 = 0.6

P(Temperature = cool) = 8 / 20 = 0.4

P(Temperature = hot | Play Golf = yes) = 5 / 10 = 0.5

P(Temperature = cool | Play Golf = yes) = 5 / 10 = 0.5

P(Temperature = hot | Play Golf = no) = 7 / 10 = 0.7

P(Temperature = cool | Play Golf = no) = 3 / 10 = 0.3

So, via Bayes rule:

$$\text{P(Play Golf = yes | Temperature = hot)} = (0.5 \times 0.5) \,/\, 0.6 = \frac{5}{12} = 0.41666\ldots$$

$$\text{P(Play Golf = no | Temperature = hot)} = (0.7 \times 0.5) \,/\, 0.6 = \frac{7}{12} = 0.58333\ldots$$

$$\text{P(Play Golf = yes | Temperature = cool)} = (0.5 \times 0.5) \,/\, 0.4 = \frac{5}{8} = 0.625$$

$$\text{P(Play Golf = no | Temperature = cool)} = (0.3 \times 0.5) \,/\, 0.4 = \frac{3}{8} = 0.375$$

3. Descriptive Analysis

```
library(ISLR)
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
summary(Auto[,-which(names(Auto)=='name')])
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
```

3.a) Which predictors are quantitative and which qualitative

- `mpg`, `cylinders`, `displacement`, `horsepower`, `weight` and `acceleration` are all quantitative
- `year` is numerical but ordinal (addition does not make sense) so is qualitative
- `origin` is numerical but the numbers are presumably arbitrary category labels, so is qualitative
- `name` is clearly qualitative

3.b) Range of quantitative predictors

```
range_table <-function(data_set){
  r = data.frame(max = sapply(data_set,max), min = sapply(data_set,min))
  r$range = r$max - r$min
  return(r)
}
quants = c('mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration')
```

```
knitr::kable(range_table(Auto[,quants]))
```

	max	min	range
mpg	46.6	9	37.6
cylinders	8.0	3	5.0
displacement	455.0	68	387.0
horsepower	230.0	46	184.0
weight	5140.0	1613	3527.0
acceleration	24.8	8	16.8

3.c) Median and variance of quantitative preditors

```
var_med_table <-function(data_set){
  r = data.frame(variance = sapply(data_set,var), median=sapply(data_set,median))
  return(r)
}
knitr::kable(var_med_table(Auto[,quants]))
```

	variance	median
mpg	6.091814e+01	22.75
cylinders	2.909696e+00	4.00
displacement	1.095037e+04	151.00
horsepower	1.481569e+03	93.50
weight	7.214847e+05	2803.50
acceleration	7.611331e+00	15.50

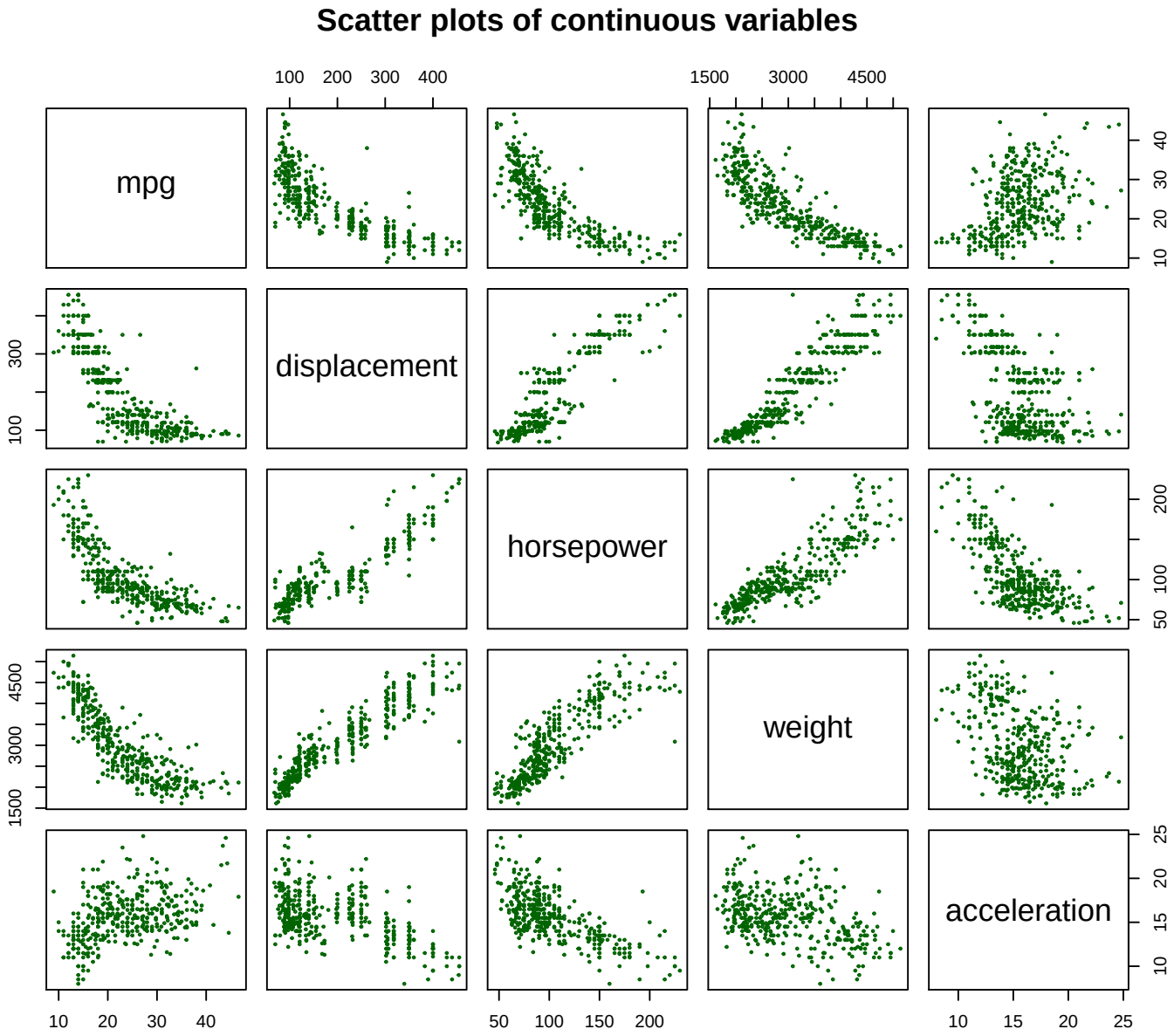
3.d) With observations removed

```
AutoSubset = Auto[-(11:79),]
knitr::kable(data.frame(range_table(AutoSubset[,quants]),
                        var_med_table(AutoSubset[,quants])))
```

	max	min	range	variance	median
mpg	46.6	11.0	35.6	6.135039e+01	23.9
cylinders	8.0	3.0	5.0	2.748938e+00	4.0
displacement	455.0	68.0	387.0	1.003229e+04	144.0
horsepower	230.0	46.0	184.0	1.291977e+03	90.0
weight	4997.0	1649.0	3348.0	6.574970e+05	2789.0
acceleration	24.8	8.5	16.3	7.296234e+00	15.5

3.e) Graphical Exploration

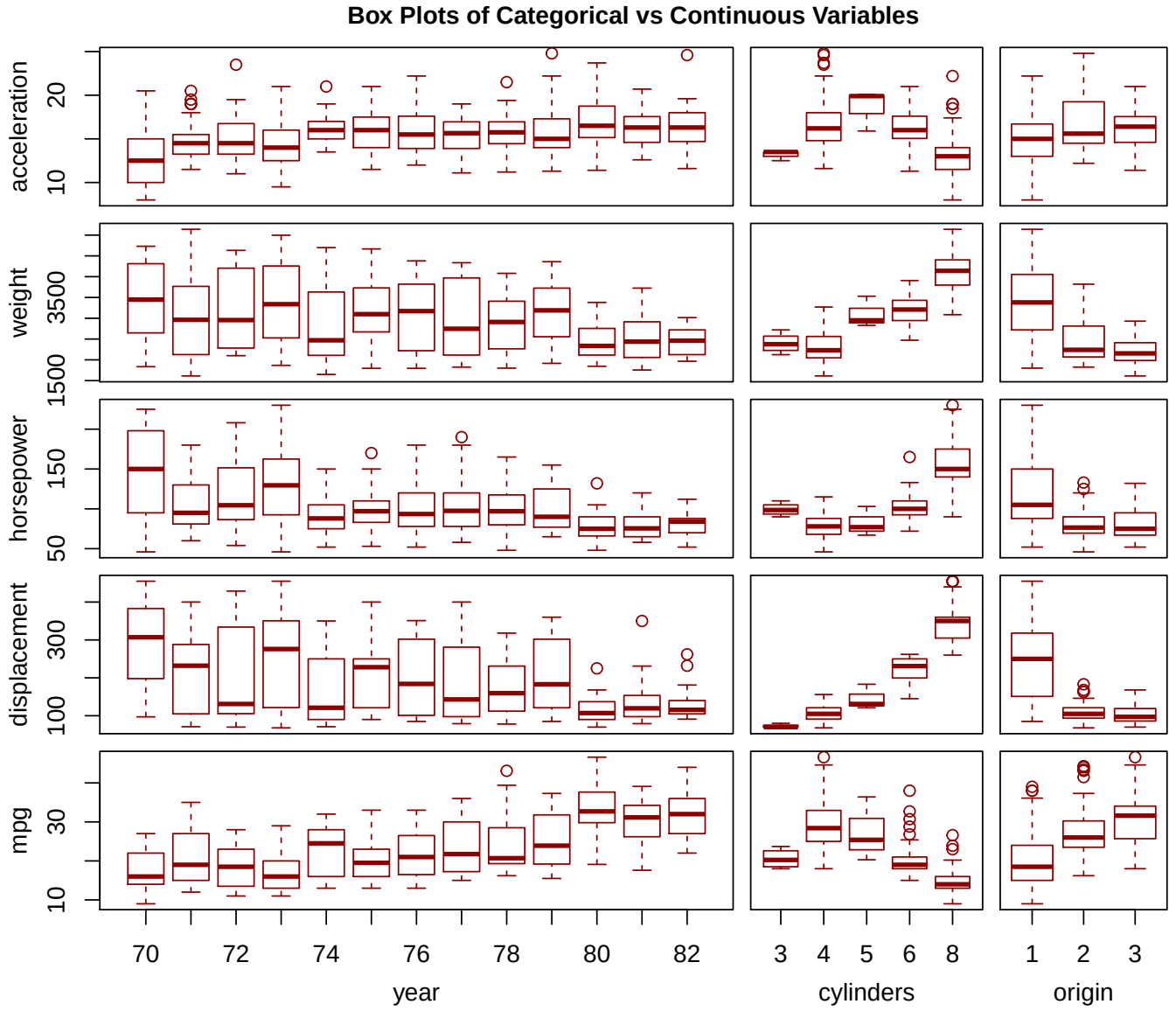
```
conts = c('mpg','displacement','horsepower','weight','acceleration')
pairs(Auto[,conts], col="darkgreen", cex=0.3,
      main = 'Scatter plots of continuous variables')
```



There are clearly strong relationships among the continuous predictors. `Displacement`, `horsepower` and `weight` look to have strong, positive nearly-linear relationships with each other and a negative, non-linear relationship with `mpg`. `Acceleration` is the least related to the other variables, but does have a slight positive relationship with `mpg` and a negative relationship with `displacement`, `horsepower` and `weight`.

```
discs = c('year','cylinders','origin')
y = c(0,0.2,0.4,0.6,0.8,1)
x = c(0,0.6,0.83,1)

for (j in 1:3){
  for (i in 1:5){
    par(fig=c(x[j],x[j+1],y[i],y[i+1]), new=!(i==1 & j==1), mar=rep(0.3,4),oma=c(3,3,2,0))
    boxplot(Auto[,conts[i]]~Auto[,discs[j]], ylab='', xlab='', border='dark red',
```

```
            xaxt=if(i==1) "s" else "n", yaxt=if(j==1) "s" else "n")
  }
}
mtext('Box Plots of Categorical vs Continuous Variables', side=3, outer = TRUE, line=0.3, font=2)
mtext(discs, side=1, outer = TRUE, line=2, at=c(0.3,0.715,0.915))
mtext(conts, side = 2, outer = TRUE, line=2, at = seq(0.1, 0.9, by = 0.2))
```



`Acceleration` and `mpg` tend to increase with `year`, whilst `weight`, `horsepower` and `displacement` tend to reduce with `year`. `Mpg` and `acceleration` tend to be best for cars with 4-5 `cylinders` and from `origin` 3. `Displacement`, `horsepower`, and `weight` tend to increase with more `cylinders`, and in `origin` 1.

3.f) Useful variables in predicting mpg

The scatter plots suggest `mpg` has a strong, negatively-correlated relationship with `displacement`, `horsepower` and `weight`, however these relationships are not perfectly linear. `Acceleration` does not appear to have a strong relationship with `mpg`, but there is a small positive correlation. There does seems to be a strong relationship with the categorical variables - `year`, `cylinders` and `origin` - in particular we see very different values depending on `origin`. All of these predictor variables would seem to be useful in a regression or other prediction of `mpg`.

4. Linear Regression

4.a) Regress mpg on horsepower

```
model = lm(mpg~horsepower,Auto)
summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

4.a.i) Is there a relationship?

The low p-value for horsepower suggests we can reject the null hypothesis that there is no relationship - i.e. we can have reasonable confidence in the existence of a relationship.

4.a.ii) How strong is the relationship?

The R-squared value is 60.6%, suggesting that 60.6% of the variability in the mpg can be explained by a linear relationship with horsepower.

4.a.iii) Is the relationship positive or negative

The estimate of the slope of the relationship is -0.16, so as horsepower increases, mpg decreases - i.e. a negative relationship.

4.a.iv) Prediction for horsepower of 89

```
predict(model,data.frame(horsepower=89),interval='confidence', level = 0.99)
```

```
##        fit      lwr      upr
## 1 25.88768 25.19633 26.57903
```

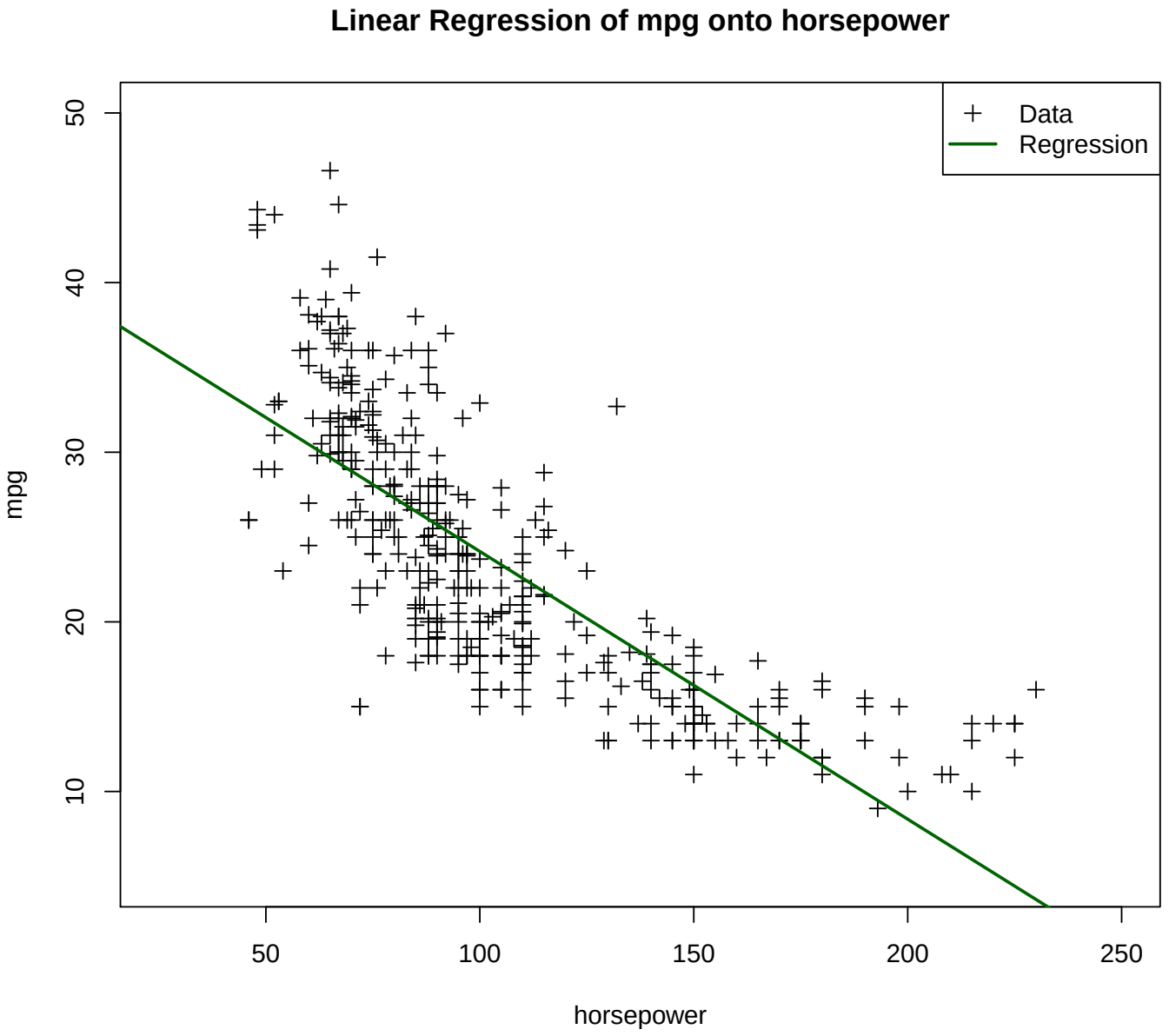
```
predict(model,data.frame(horsepower=89),interval='prediction', level = 0.99)
```

```
##        fit      lwr      upr
## 1 25.88768 13.17035 38.60501
```

The predicted mpg is 25.88768 with a 99% confidence interval between 25.19633 and 26.57903, and a 99% prediction interval between 13.17035 and 38.60501.

4.b) Plot of response, predictor and regression line

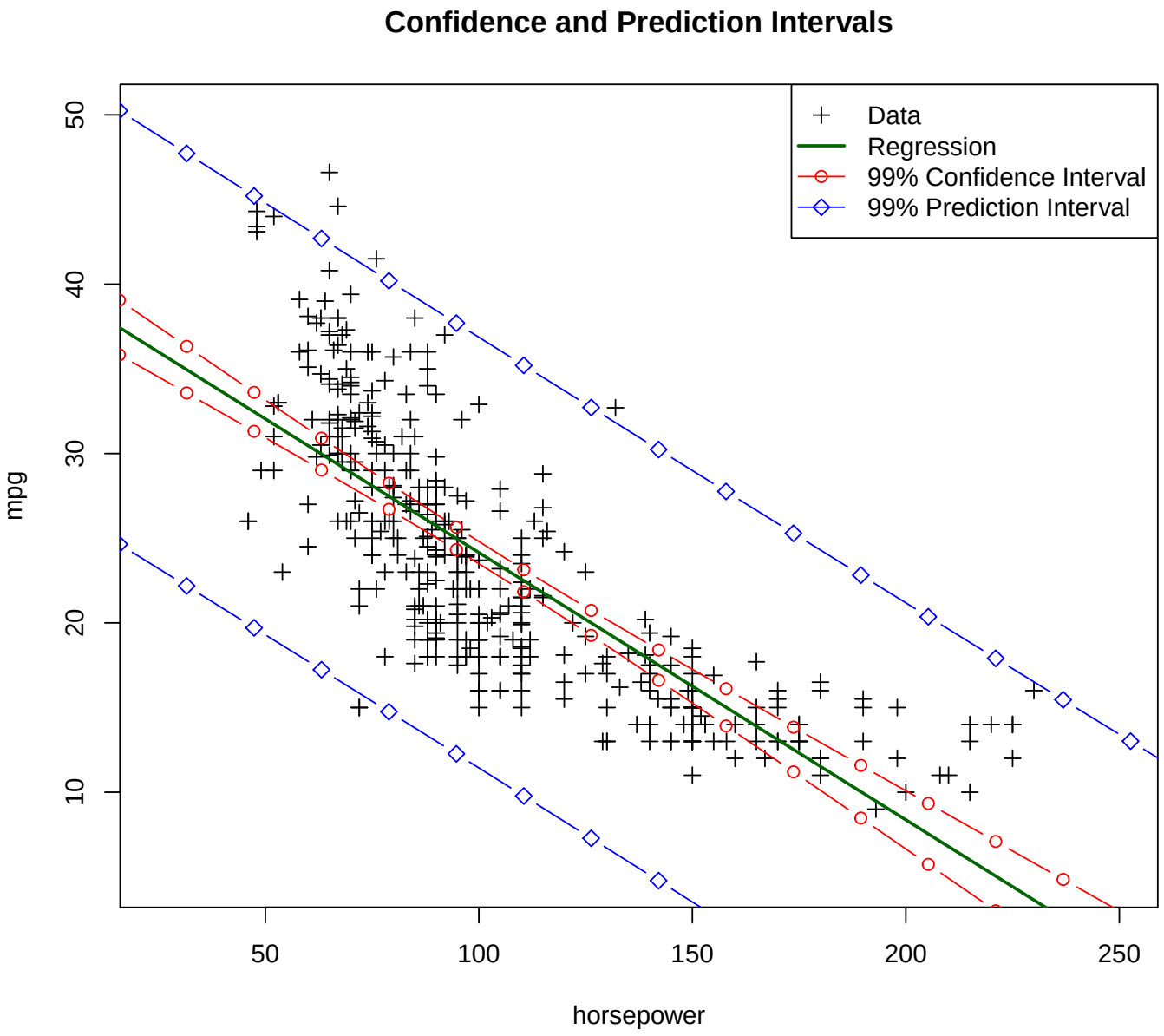
```
plot(mpg~horsepower,data=Auto, xlab='horsepower',ylab='mpg', xlim=c(25,250), ylim=c(5,50),
     pch=3, main = 'Linear Regression of mpg onto horsepower')
abline(model, col='dark green', lwd=2)
legend(x ="topright", col=c("black","dark green"), legend = c('Data','Regression'),
       lty=c(0,1), lwd=c(1,2), pch=c(3,NA))
```



4.c) Plot of confidence intervals and prediction intervals

```
plot(Auto$horsepower, Auto$mpg, xlab='horsepower', ylab='mpg', pch=3,
     xlim=c(25,250), ylim=c(5,50), new=TRUE, main='Confidence and Prediction Intervals')

abline(model, col='dark green', lwd=2)

new_horsepower = data.frame(horsepower=seq(0,300,length=20))

ivals_conf = predict(model,new_horsepower,interval='confidence', level=0.99)
```

```
ivals_pred = predict(model,new_horsepower,interval='prediction', level=0.99)

lines(new_horsepower$horsepower,ivals_conf[,"lwr"], col="red", type="b", pch=1)
lines(new_horsepower$horsepower,ivals_conf[,"upr"], col="red", type="b", pch=1)
lines(new_horsepower$horsepower,ivals_pred[,"lwr"], col="blue", type="b", pch=5)
lines(new_horsepower$horsepower,ivals_pred[,"upr"], col="blue", type="b", pch=5)

legend(x ="topright", col=c("black","dark green","red", "blue"),
       legend = c('Data','Regression','99% Confidence Interval','99% Prediction Interval'),
       lty=c(0,1,1,1), lwd=c(1,2,1,1), pch=c(3,NA,1,5))
```



5. Logistic Regression

5.a) Load data and display stats about training set

```
train = read.csv("Training_set for Q5.txt")
test =  read.csv("Testing_set for Q5.txt")

#Number of rows in the training set
nrow(train)
```

```
## [1] 2000
```

```
#Quick Summary of fields
kable(summary(train),align=rep('r',6))
```

Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
Min. :20.10	Min. :18.96	Min. : 0.0	Min. : 426.0	Min. :0.002824	Min. :0.0000
1st Qu.:20.89	1st Qu.:21.82	1st Qu.: 0.0	1st Qu.: 448.0	1st Qu.:0.003375	1st Qu.:0.0000
Median :21.20	Median :25.00	Median : 0.0	Median : 485.5	Median :0.003905	Median :0.0000
Mean :21.42	Mean :24.22	Mean :144.7	Mean : 634.6	Mean :0.003836	Mean :0.2775
3rd Qu.:22.10	3rd Qu.:26.29	3rd Qu.:433.0	3rd Qu.: 845.8	3rd Qu.:0.004343	3rd Qu.:1.0000
Max. :23.18	Max. :28.50	Max. :744.0	Max. :1139.0	Max. :0.004817	Max. :1.0000

```
#Standard Deviations
kable(sapply(train,sd), col.names = 'Standard Deviation')
```

	Standard Deviation
Temperature	0.7285295
Humidity	2.5959727
Light	216.9285743
CO2	243.7779598
HumidityRatio	0.0005573
Occupancy	0.4478773

```
#Correlations
kable(cor(train),digits=3)
```

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
Temperature	1.000	0.795	0.687	0.827	0.900	0.600
Humidity	0.795	1.000	0.499	0.588	0.979	0.452
Light	0.687	0.499	1.000	0.889	0.587	0.905
CO2	0.827	0.588	0.889	1.000	0.698	0.823
HumidityRatio	0.900	0.979	0.587	0.698	1.000	0.524
Occupancy	0.600	0.452	0.905	0.823	0.524	1.000

Set up some reusable functions to reduce code duplication

```
#Print out confusion matris
print_confusion_matrix <- function(model, test_data, test_actual){
  probs = predict(model,newdata=test_data,type='response')
  preds = rep(0,nrow(test_data))
  preds[probs>=0.5] = 1
  addmargins(table(prediction=preds,actual=test_actual))
}
#Print out various accuracy measures
print_accuracy_rates <- function(model, test_data, test_actual){
  probs = predict(model,newdata=test_data,type='response')
  preds = rep(0,nrow(test_data))
  preds[probs>=0.5] = 1
  accuracy = data.frame(
    Accuracy_Rate = mean(preds == test_actual),
    Error_Rate = mean(preds != test_actual),
    True_Positive_Rate = sum(preds==1 & test_actual==1) / sum(test_actual==1),
    False_Positive_Rate = sum(preds==1 & test_actual==0) / sum(test_actual==0),
    True_Negative_Rate = sum(preds==0 & test_actual==0) / sum(test_actual==0),
    False_Negative_Rate = sum(preds==0 & test_actual==1) / sum(test_actual==1),
    row.names = 'Rate')
  kable(t(accuracy))
}
```

5.b) - Regress Occupancy on Temperature

```
# Build Model
model_1 = glm(Occupancy~Temperature, data=train, family=binomial)

# Model Summary
summary(model_1)
```

```
##
## Call:
## glm(formula = Occupancy ~ Temperature, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2180  -0.4862  -0.3368   0.6739   2.5871
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -54.4413     2.4166  -22.53   <2e-16 ***
## Temperature   2.4701     0.1107   22.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2362.3  on 1999  degrees of freedom
## Residual deviance: 1548.5  on 1998  degrees of freedom
## AIC: 1552.5
##
## Number of Fisher Scoring iterations: 5
```

```
#Confusion Matrix
print_confusion_matrix(model_1, test, test$Occupancy)
```

```
##           actual
## prediction   0   1 Sum
##        0   182  39 221
##        1    58  21  79
##        Sum 240  60 300
```

```
#Predictive Accuracy
print_accuracy_rates(model_1, test, test$Occupancy)
```

	Rate
Accuracy_Rate	0.6766667
Error_Rate	0.3233333
True_Positive_Rate	0.3500000
False_Positive_Rate	0.2416667
True_Negative_Rate	0.7583333
False_Negative_Rate	0.6500000

5.c) - Regress Occupancy on Humidity

```
# Build Model
model_2 = glm(Occupancy~Humidity, data=train, family=binomial)

# Model Summary
summary(model_2)
```

```
##
## Call:
## glm(formula = Occupancy ~ Humidity, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4511  -0.8361  -0.2682   0.8937   2.3203
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -16.56899    0.94141  -17.60   <2e-16 ***
## Humidity      0.62293    0.03659   17.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2362.3  on 1999  degrees of freedom
## Residual deviance: 1845.8  on 1998  degrees of freedom
## AIC: 1849.8
##
## Number of Fisher Scoring iterations: 5
```

```
# Confusion Matrix
print_confusion_matrix(model_2, test, test$Occupancy)
```

```
##            actual
## prediction   0   1 Sum
##        0   133  22 155
##        1   107  38 145
##        Sum 240  60 300
```

```
# Accuracy Rates
print_accuracy_rates(model_2, test, test$Occupancy)
```

	Rate
Accuracy_Rate	0.5700000
Error_Rate	0.4300000
True_Positive_Rate	0.6333333
False_Positive_Rate	0.4458333
True_Negative_Rate	0.5541667
False_Negative_Rate	0.3666667

5.d) - Regress Occupancy on All Features

```
# Build Model
model_3 = glm(Occupancy~., data=train, family=binomial)

# Model Summary
summary(model_3)
```

```
##
## Call:
## glm(formula = Occupancy ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9467  -0.1274  -0.1082   0.2486   2.0429
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    9.089e+00  4.658e+01   0.195    0.845
## Temperature   -1.102e+00  2.185e+00  -0.504    0.614
## Humidity       8.278e-01  1.351e+00   0.613    0.540
## Light          1.415e-02  1.100e-03  12.864  < 2e-16 ***
## CO2            4.800e-03  1.034e-03   4.642 3.45e-06 ***
## HumidityRatio -3.495e+03  8.434e+03  -0.414    0.679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2362.3  on 1999  degrees of freedom
## Residual deviance:  487.4  on 1994  degrees of freedom
## AIC: 499.4
##
## Number of Fisher Scoring iterations: 7
```

```
# Confusion Matrix
print_confusion_matrix(model_3, test, test$Occupancy)
```

```
##           actual
## prediction   0   1 Sum
##        0   184  13 197
##        1    56  47 103
##        Sum 240  60 300
```

```
# Accuracy Rates
print_accuracy_rates(model_3, test, test$Occupancy)
```

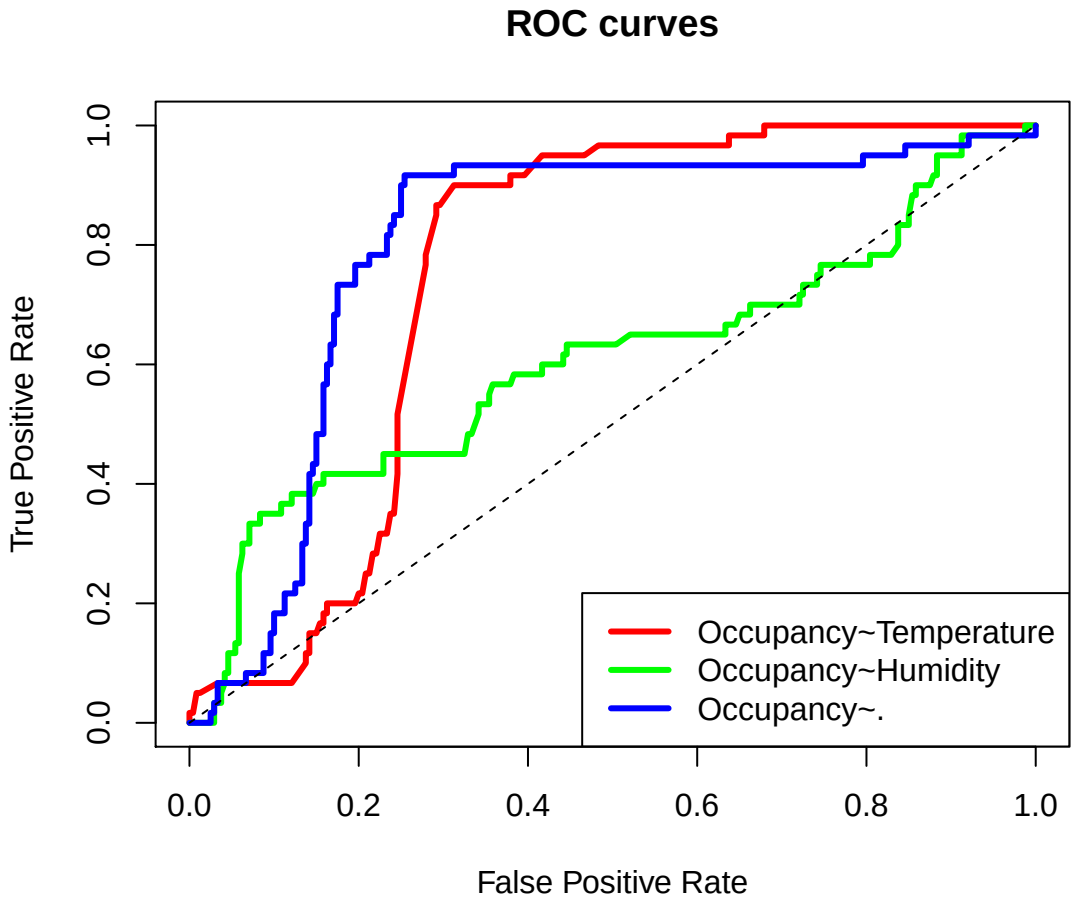
	Rate
Accuracy_Rate	0.7700000
Error_Rate	0.2300000
True_Positive_Rate	0.7833333
False_Positive_Rate	0.2333333
True_Negative_Rate	0.7666667
False_Negative_Rate	0.2166667

5.e Compare models by drawing ROC Curves and calculating AUROC values

ROC curves

```
library(ROCR)
```

```
model_formulas = c('Occupancy~Temperature','Occupancy~Humidity','Occupancy~.')

get_ROC_Curve <- function(model, test_data, test_actual){
  probs = predict(model,newdata=test_data,type='response')
  preds = prediction(probs,test_actual)
  perf = performance(preds, measure="tpr", x.measure="fpr")
  return(perf)
}

plot(0,xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC curves",  ylim = 0:1, xlim = 0:1, type = "l")

plot(get_ROC_Curve(model_1, test, test$Occupancy),lwd=3,col="red",add=TRUE)
plot(get_ROC_Curve(model_2, test, test$Occupancy),lwd=3,col="green",add=TRUE)
plot(get_ROC_Curve(model_3, test, test$Occupancy),lwd=3,col="blue",add=TRUE)

lines(0:1,0:1,lty=2)

legend("bottomright", model_formulas, col=c('red','green','blue'), lty=rep(1,3), lwd=rep(3,3))
```



Calculate Areas under the ROC curves

```
get_AUROC_Value <- function(model, test_data, test_actual){
  probs = predict(model,newdata=test_data,type='response')
  preds = prediction(probs,test_actual)
  perf = performance(preds, measure="auc")@y.values[[1]]
  return(perf)
}
kable(data.frame(AUROC = c(get_AUROC_Value(model_1, test, test$Occupancy),
                           get_AUROC_Value(model_2, test, test$Occupancy),
                           get_AUROC_Value(model_3, test, test$Occupancy)),
                 row.names = model_formulas))
```

	AUROC
Occupancy~Temperature	0.7527431
Occupancy~Humidity	0.6038194
Occupancy~.	0.7977778

The model which regresses Occupancy upon all of the predictors has the highest AUROC and seems to be closer to the top left corner of the ROC plot. This is the better model for most thresholds.

6. Resampling methods

6.a) prediction of better model

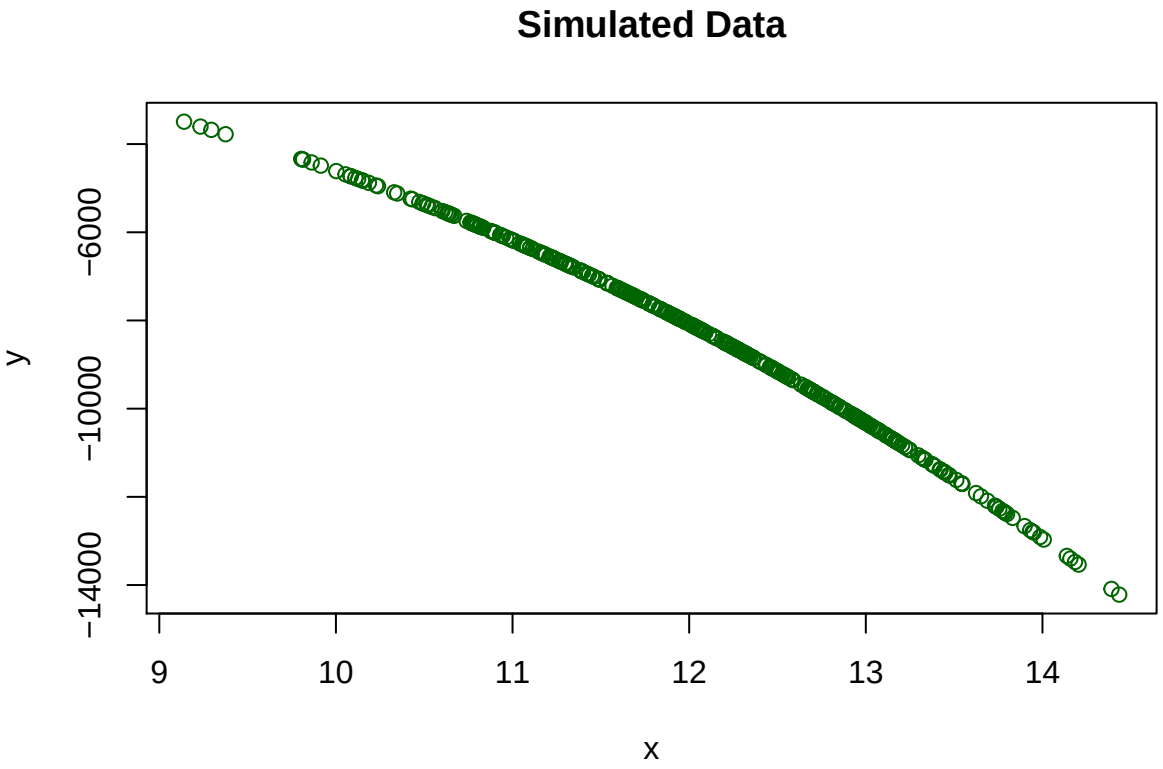
The quartic (degree 4) model is likely to fit the test data better.

The quartic model is more flexible than is necessary to describe the underlying relationship, so it will suffer from some variance and have a tendency to overfit; however, this is mostly mitigated by the high sample size and the inclusion of all the actual factors upon which y depends, i.e. all the terms up to and including cubic.

The quadratic model is less flexible than necessary to describe the actual relationship so is susceptible to bias. Given the large sample size I would expect the MSE to be lower for the quartic model than the quadratic model in the majority of cases. That said, where the cubic coefficient makes the cubic term very small compared to the irreducible error then the quartic model may be marginally worse than the quadratic model.

6.b) Generate the simulated data

```
set.seed(235)
x = 12 + rnorm(400)
y = 1 - x + 4*x^2 - 5*x^3 + rnorm(400)
plot(x,y,col='dark green',main='Simulated Data',pch=1)
```



The relationship between X and Y is close to linear, but not exactly. The irreducible error term generated by `rnom(400)` will have a standard deviation of 1, which is very small compared to the underlying value of y, so is not particularly visible on the chart.

6.c) Compute LOOCV and 10-fol CV errors

```
library(boot)
set.seed(34)
d = data.frame(X=x,Y=y)
```

6.c.i) Build Quadratic Model

```
lm.m.p2 = glm(Y~poly(X,2),data=d)
summary(lm.m.p2)
```

```
##
## Call:
## glm(formula = Y ~ poly(X, 2), data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -48.414   -6.713    1.202    6.156   63.558
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -8.406e+03  4.829e-01 -17407.5   <2e-16 ***
## poly(X, 2)1 -4.207e+04  9.658e+00  -4355.6   <2e-16 ***
## poly(X, 2)2 -4.659e+03  9.658e+00   -482.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 93.27952)
##
##     Null deviance: 1791391934  on 399  degrees of freedom
## Residual deviance:      37032  on 397  degrees of freedom
## AIC: 2954.4
##
## Number of Fisher Scoring iterations: 2
```

```
#The LOOCV for quadratic model is
cv.glm(d,lm.m.p2)$delta[1]
```

```
## [1] 100.2671
```

```
#The 10-fold CV for quadratic model is
cv.glm(d,lm.m.p2,K=10)$delta[1]
```

```
## [1] 105.8857
```

6.c.i) Build Quartic Model

```
lm.m.p4 = glm(Y~poly(X,4),data=d)
summary(lm.m.p4)
```

```
##
## Call:
## glm(formula = Y ~ poly(X, 4), data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9184  -0.5951   0.0207   0.6436   3.3437
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -8.406e+03  5.008e-02 -1.678e+05   <2e-16 ***
## poly(X, 4)1 -4.207e+04  1.002e+00 -4.200e+04   <2e-16 ***
## poly(X, 4)2 -4.659e+03  1.002e+00 -4.651e+03   <2e-16 ***
## poly(X, 4)3 -1.914e+02  1.002e+00 -1.911e+02   <2e-16 ***
## poly(X, 4)4  2.331e-03  1.002e+00  2.000e-03    0.998
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.003333)
##
##     Null deviance: 1.7914e+09  on 399  degrees of freedom
## Residual deviance: 3.9632e+02  on 395  degrees of freedom
## AIC: 1143.5
##
## Number of Fisher Scoring iterations: 2
```

```
#The LOOCV for quartic model is
cv.glm(d,lm.m.p4)$delta[1]
```

```
## [1] 1.017966
```

```
#The 10-fold CV for quartic model is
cv.glm(d,lm.m.p4,K=10)$delta[1]
```

```
## [1] 1.023266
```

6.d) Compute LOOCV and 10-fold CV errors with different seed

```
set.seed(68)
#The LOOCV for quadratic model is
cv.glm(d,lm.m.p2)$delta[1]
```

```
## [1] 100.2671
```

```
#The LOOCV for quartic model is
cv.glm(d,lm.m.p4)$delta[1]
```

```
## [1] 1.017966
```

```
#The 10-fold CV for quadratic model is
cv.glm(d,lm.m.p2,K=10)$delta[1]
```

```
## [1] 103.7111
```

```
#The 10-fold CV for quartic model is
cv.glm(d,lm.m.p4,K=10)$delta[1]
```

```
## [1] 1.020981
```

The LOOCV results are the same for 6.c as they are for 6.d, whereas for 10-fold CV the results differ slightly between 6.c and 6.d. LOOCV does not include any random sampling - the result is a deterministic function of the input - so it makes sense that they would not differ. K-fold CV requires random sampling so the result is dependent upon the random samples selected, which is dependent upon the seed. As the seeds differ, so will the samples chosen, and so will the cross-validation errors calculated therefrom.

6.e) Which model had smallest error?

The quartic model had significantly smaller CV error than the quadratic model. This agrees with the expectation in 6.a) - the quartic model covers all factors relevant to the actual population.

6.f) Comment on the coefficient estimates

The t values are quite large and the p values very small for the terms up to and including cubic, suggesting it is very likely there is a cubic relationship. The coefficient for the quartic term is close to zero with a very small t value and a large p value suggesting very little significance can be derived from including the quartic term.

6.g) Fit a cubic model and compare

```
set.seed(34)

#cubic model
lm.m.p3 = glm(Y~poly(X,3),data=d)
summary(lm.m.p3)
```

```
##
## Call:
## glm(formula = Y ~ poly(X, 3), data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9184  -0.5950   0.0207   0.6436   3.3436
##
## Coefficients:
##               Estimate Std. Error   t value Pr(>|t|)
## (Intercept) -8.406e+03  5.002e-02 -168056.7   <2e-16 ***
## poly(X, 3)1 -4.207e+04  1.000e+00  -42050.4   <2e-16 ***
## poly(X, 3)2 -4.659e+03  1.000e+00   -4657.2   <2e-16 ***
## poly(X, 3)3 -1.914e+02  1.000e+00    -191.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.0008)
##
##     Null deviance: 1.7914e+09  on 399  degrees of freedom
## Residual deviance: 3.9632e+02  on 396  degrees of freedom
## AIC: 1141.5
##
## Number of Fisher Scoring iterations: 2
```

```
#The LOOCV for the cubic model is
cv.glm(d,lm.m.p3)$delta[1]
```

```
## [1] 1.010868
```

```
#The 10-fold CV for cubic model is
cv.glm(d,lm.m.p3,K=10)$delta[1]
```

```
## [1] 1.011995
```

The LOOCV and 10-fold CV error are very close to those for the quartic model discussed above, but slightly smaller. Given that the true underlying relationship is actually cubic it makes sense that they would be slightly smaller - the quartic term in the best model above provided opportunity for overfitting. The actual noise in the generated data is generated from a standard gaussian distribution, so has a variance of 1, which is very close to the MSE we see here, suggesting we have a good fit.