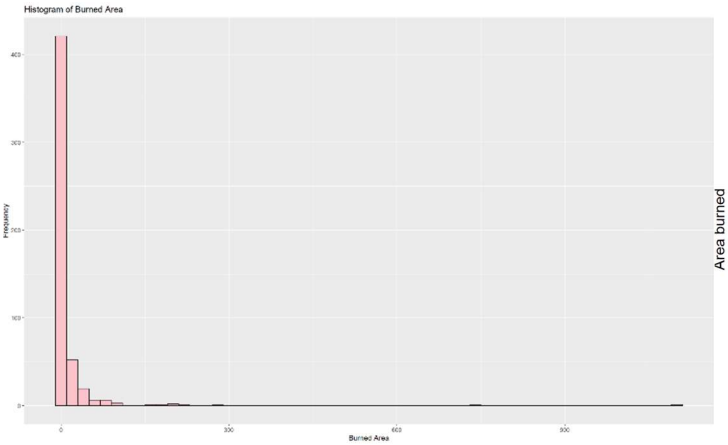
STAT 6340 Mini Project 2

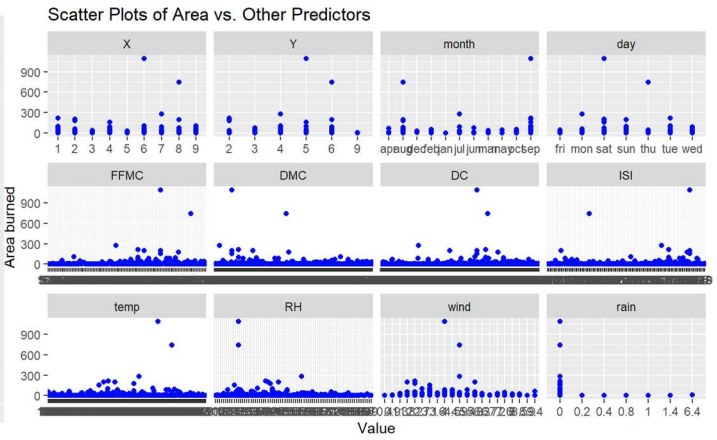
Lakshmipriya Narayanan

SECTION 1: Observations and Answers

1

(a): Exploratory analysis of data:

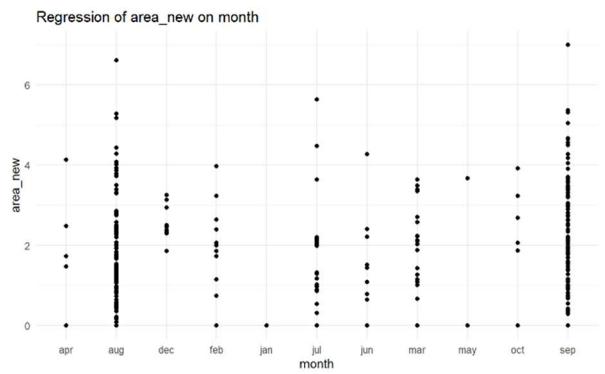


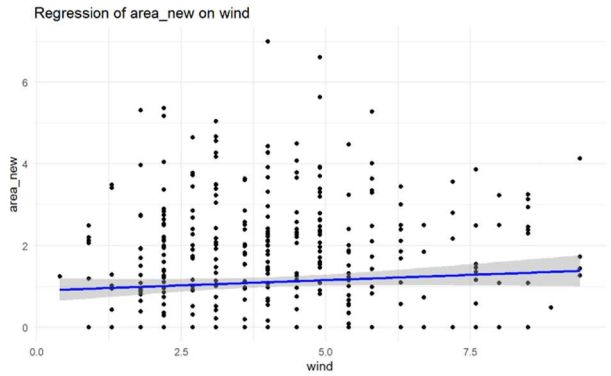


These are some plots from the analysis(see section 2 for more). From general knowledge, I would have anticipated variables like temperature, wind, rain and FFMC to contribute to the forest fire and from the scatter plots, these variables seem to affect the area burned. Also, the histogram for the area is heavily left skewed indicating a transformation requirement but there are some significant fires. From the correlation plot, we see that most variables have positive correlation with area burned.

(b): From the exploratory analysis done above, we see from the histogram and boxplot of area burned that the data is very left skewed. This may result in inaccurate analysis and predictions and hence, transformation is necessary. We perform log transformation as mentioned in the question. Since area = 0 for many data, we add 1 and then take log.

(c): From the p-values obtained in the summary of our linear models, we conclude that predictors with p-value <0.05 are statistically significant. And from the summaries, predictor variable 'month' is statistically significant with a p-value= 0.0377. The rest of the variables' p-value is >0.05. (See code and output for p-values of other predictors in 'Section 2'). Therefore, we can conclude that the month of the year affects the amount of area burned in a forest fire. From the plots of transformed area with month and wind below, we can see this conclusion in picture.





(d): We fit the multiple regression model with all predictors (excluding untransformed area).
H_0: $\beta_j = 0$ VS H_A: At least one inequality.
The test statistic, F_{obs} = 1.445 and p-value= 0.073 > 0.05 and hence we fail to reject H_0. But we reject H_0 for predictor variables 'DMC' and 'monthdec' because their p-values are 0.035 < 0.05 and 0.006 < 0.05 respectively.

(e) : Our reasonably good multiple regression model is the one with predictor variables 'temp', 'month', 'DMC', 'DC', 'RH', 'wind', 'rain', and 'ISI'. This was done by eliminating each predictor one at a time to see which model has a p-value < 0.05. Our model's p-value = 0.02008. This can also be justified by taking a close look at the correlation plot while we conducted our exploratory data analysis. These predictor variables were strongly correlated with the transformed burned area.

(f):
area_new = $\beta_0 + \beta_1 * temp + \beta_2 * wind + \beta_3 * RH + \beta_4 * rain + \beta_5 * DMC + \beta_6 * DC + \beta_7 * ISI + \sum_{i=1}^{11} \beta_8 * month_i$ where,
$\sum_{i=1}^{11} \beta_8 * month_i$ = 0.25 monthaug + 2.12 monthdec + 0.14 monthfeb - 0.64 monthjan + 0.11 monthjul - 0.29 monthjun - 0.36 monthmar + 0.67 monthmay + 0.84 monthoct + 0.89 monthsep

Therefore, the final equation is:

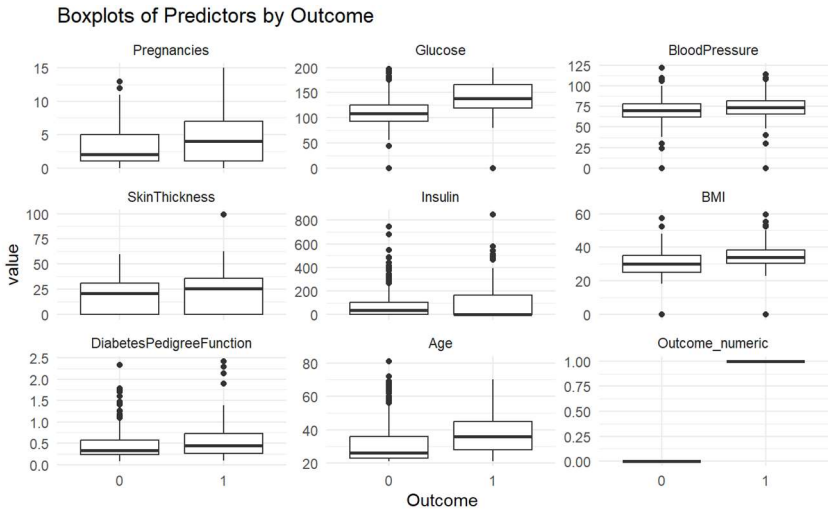
area_new =0.42 + 0.036 * temp + 0.054 * wind + 0.001 * RH + 0.081 * rain + 0.003 * DMC – 0.0019 * DC – 0.008 * ISI+0.25 monthaug + 2.12 monthdec + 0.14 monthfeb - 0.64 monthjan + 0.11 monthjul - 0.29 monthjun - 0.36 monthmar + 0.67 monthmay + 0.84 monthoct + 0.89 monthsep

(g): Predicted burned area = 1.564 hectares. Refer to code in Section 2.

(h): Yes, there are outliers in our reasonably good model. This can be seen by making a diagnostic plot of our model and figuring out the cook's distance to identify the present outliers. The model with outliers has adjusted R^2 = 0.022 and p value = 0.07. Whereas the model without the outliers has R^2 = 0.04 (which is closer to 1 than 0.02 implying better fit) and p value = 0.0009 which is very small compared to the model with the outliers present. So, it has a better fit. Hence, removing the outliers makes our model a better fit. And also, apart from monthdec being a significant predictor, we have temp also to be significant (which is common knowledge) and on the other hand DMC is not significant because it has a large p-value = 0.098. Finally, I would suggest the model without the outliers since it is more robust.

2

(a): Exploratory analysis:



From the above boxplots of all predictors across Outcome, we can see that variables like Glucose, Blood pressure, BMI, Pregnancies and Age have a significant effect on determining whether a person is diabetic. This also matched the common knowledge of factors that cause diabetes like Glucose, Blood pressure and BMI. And the correlation matrix confirms our assertions.

(b): For LDA, from the confusion matrix, we see that sensitivity and specificity for training data is 0.88 and 0.56 respectively. Whereas for test data it is 0.82 and 0.61. Our training data here has high sensitivity which indicates that this model is a good fit for our data where we have included all predictors to make a prediction. The misclassification rate for training data is 0.22 and test data is 0.25 which are low and ensures that the fit is good.

(c): For QDA, from the confusion matrix, we see that sensitivity and specificity for training data is 0.86 and 0.58 respectively. Whereas for test data it is 0.62 and 0.66. Our training data here has high sensitivity but lower than

LDA, which indicates that this model is a good fit for our data. But it is lower than what we received in LDA for test data. Misclassification rates for training and test data are 0.23 and 0.36 respectively. This indicates that training data's rate is lower than that of test data, which is what we want to achieve.

(d): For Naïve Bayes:

	Sensitivity	Specificity	Misclassification Rate
Training Data	0.843	0.605	0.238 (low => good fit)
Test Data	0.775 (smaller than training data => desired)	0.630	0.274

(e): For KNN fit from 1-100, we received three optimal K's as can be seen from the output. The training error rate is increasing as K increases. Fortunately, the test error rate is all the same indicating that our model's performance is consistent with new data. Hence, we choose the smallest value of K as optimal K and K = 12.

	ks <dbl>	err.rate.train <dbl>	err.rate.test <dbl>
12	12	0.1942857	0.25
14	14	0.2014286	0.25
25	25	0.2200000	0.25

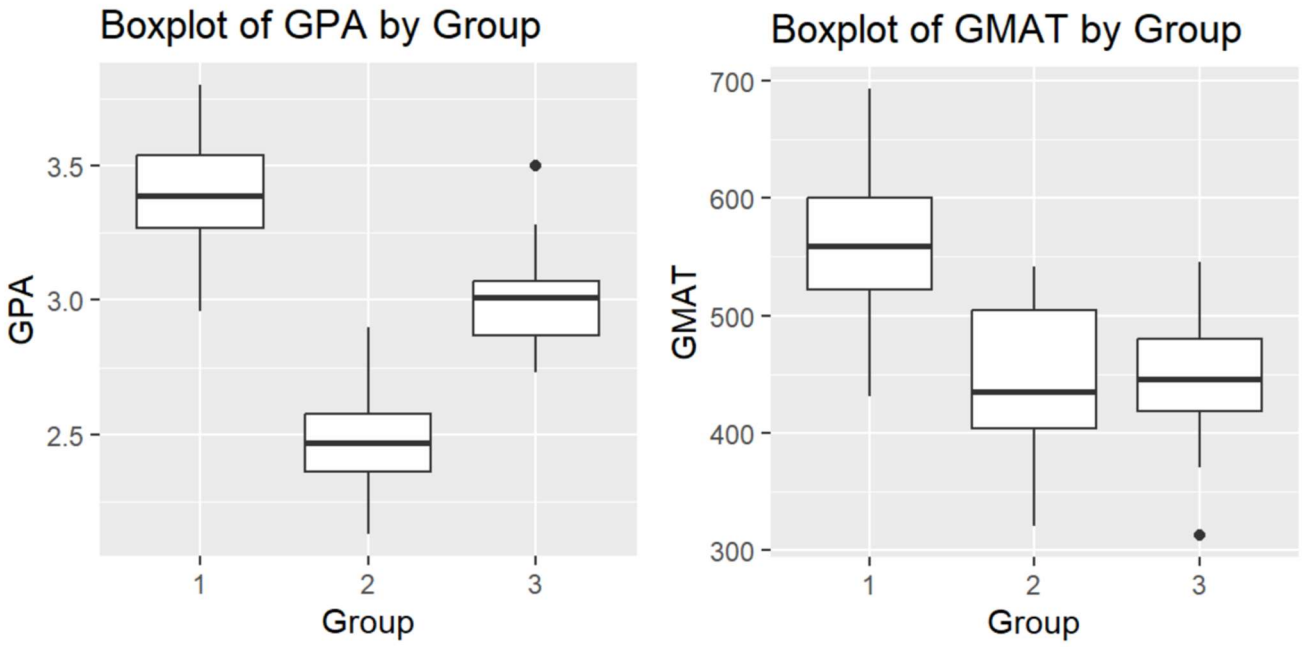
3 rows

But the issue is that when we choose this K = 12, we received three different K values which could imply overfitting because smaller K values can overfit our training data.

(f): After comparing the confusion matrices and misclassification rates I would recommend we use LDA because it has the most least misclassification rate for training data out of all the other methods like QDA, Naïve Bayes and KNN even though all these methods' misclassification rates are very close to each other.

3

(a): Exploratory Analysis:



Both the boxplots for GPA vs Group show presence of an outlier which makes sense because GPA can be affected by grade inflation and grading differences and similar for GMAT results. Group 1 has higher GPA and GMAT scores implying better performances if a student is admitted in Group 1, whereas Group 2's GMAT's boxplot looks wider implying there is more variability. Similarly, Group 3 has more variability in terms of GPA.

(b): The LDA decision boundary clearly separates Group 2 and group 3 but there is some misclassification between group 1 and group 3 which makes sense with respect to how we have separated our training and test data

(Borderline decisions on whether to offer admission or not). Hence, the decision boundary is mostly sensible because logically it has made sense.

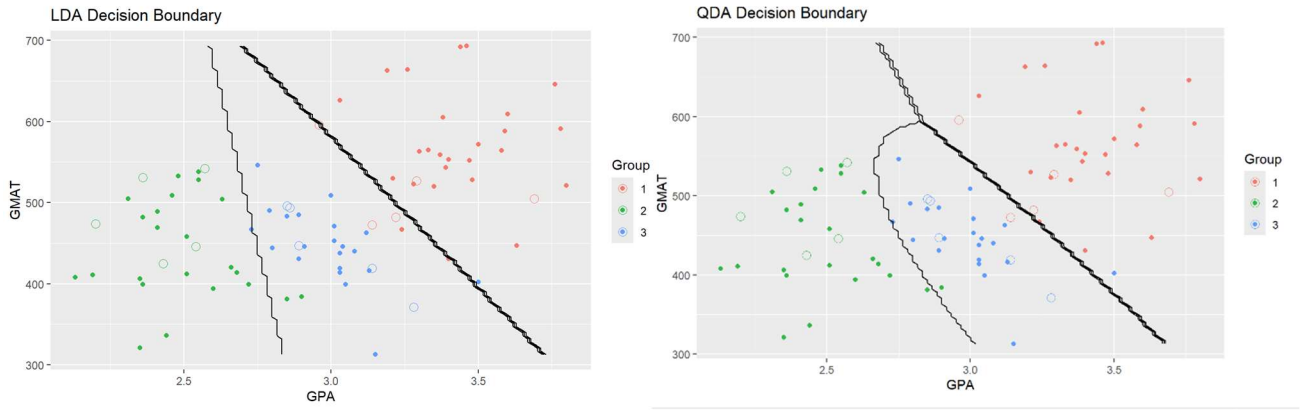
For training data. Sensitivity for the 3 groups are 0.92, 0.91 and 0.90 respectively which is very high and specificity is 0.97, 0.97 and 0.91 which is a good thing because the LDA model correctly identified most students who should be admitted and most students who should not be admitted. And for test data sensitivity is 1, 0.9, 0.95 and specificity is 0.97, 1, 0.95 which indicates the same conclusion as training data. Misclassification rates are 0.85 and 0.04 for training and test data which indicates that we might have overfit our data because the rate is high.

(c): The QDA decision boundary clearly separates Group 2 and 3 and mostly clearly separated Group 1 and 3 which aligns with the logic on which students to admit and which students to not admit. Hence, the decision boundary seems sensible. Confusion matrix results:

Group	1		2		3	
Data	Training	Test	Training	Test	Training	Test
Sensitivity	1	1	0.956	0.956	0.952	0.71
Specificity	0.977	0.88	1	0.967	0.979	1

From the above table, we can see that for group 1 sensitivity and specificity are high and =1 => the QDA model does not favor either type of error which is useful to detect false positives and false negatives. But for groups 2 and 3, sensitivity is lower than specificity which implies that our model correctly identifies students who must be admitted based on the GMAT scores and GPA. This high specificity is crucial to reduce the number of unqualified students being offered admission. The misclassification rate for training data is 0.028 and test data is 0.1.

(d): Plots of LDA and QDA decision boundary:



Comparing the misclassification rates and the decision boundaries of LDA and QDA can help us decide which classifier is best for this scenario.

Classifier	Misclassification Rates	
	Training data	Test data
LDA	0.85	0.04
QDA	0.028	0.1

According to misclassification rate, QDA has the lower rate which implies it is a better fit and can be used for new data. The decision boundary of LDA separates groups but not clearly. There are two group 2 points misclassified as group 3 and four group 1 points misclassified as group 3. Which is important because it plays an important role in decision making to offer admission. QDA clearly separated all groups (just one misclassified points in each group).

Hence, we should use the QDA classifier.

Section 2: Code

```
#Load Required Libraries
library(mvtnorm)
library(MASS)
library(e1071)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)
library(ggcorrplot)
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

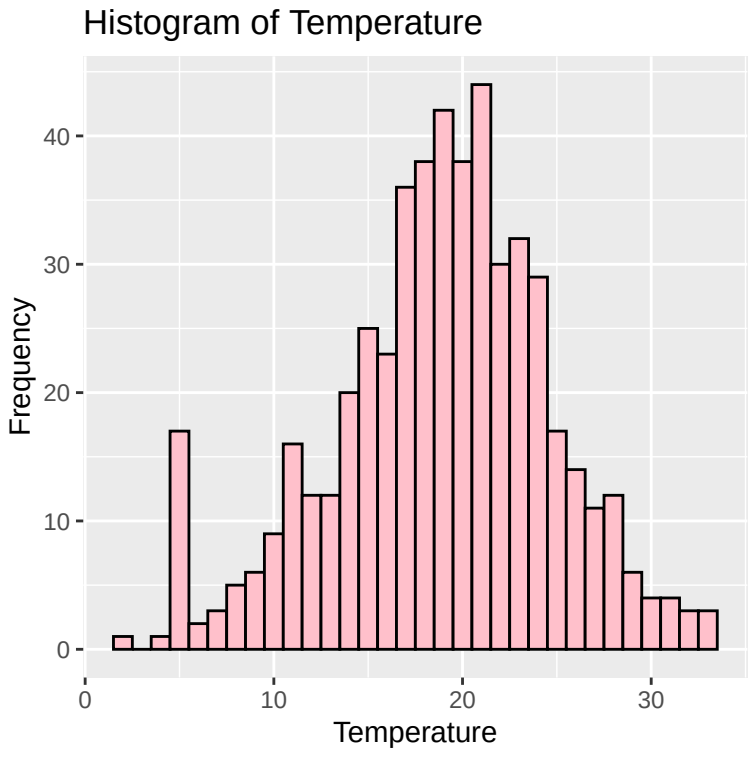
```
## Loading required package: lattice
```

```r
library(class)
library(reshape2)
```

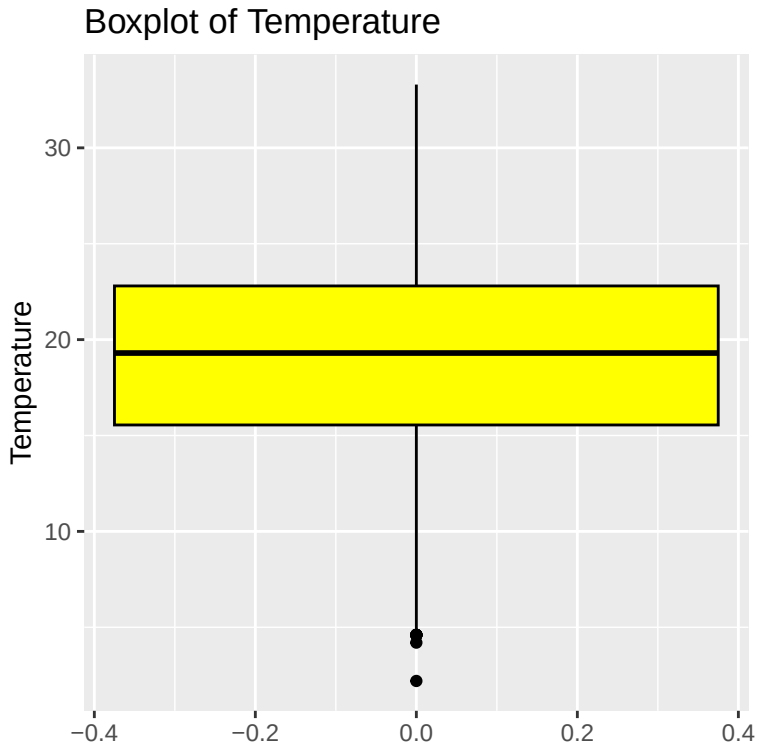
1

(a):

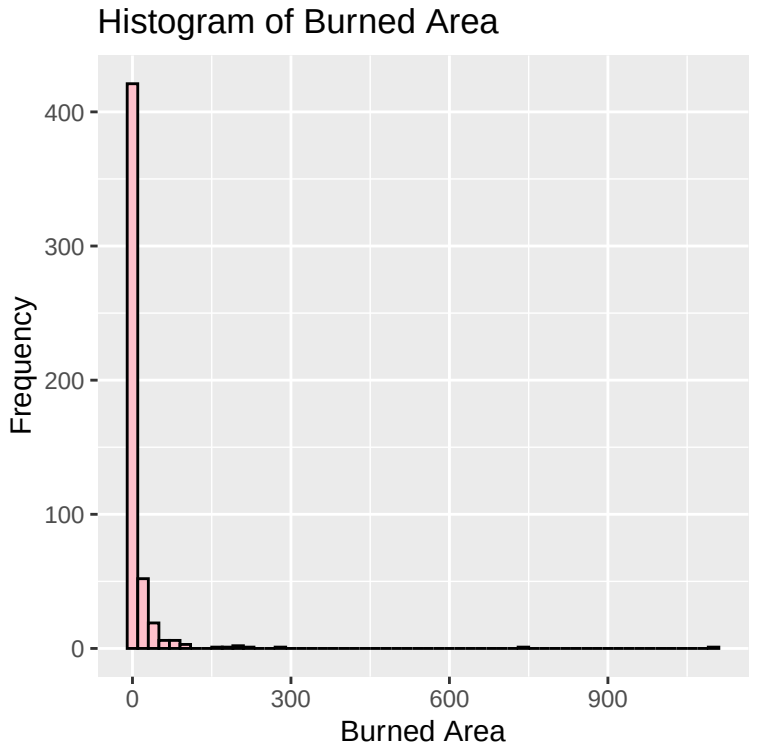
```r
#Read the data and convert qualitative variables 'day' and 'month' to factors
trainforest <- read.csv("forestfires.csv")
trainforest$day <- as.factor(trainforest$day)
trainforest$month <- as.factor(trainforest$month)

#---------DATA ANALYSIS : ------------------------
#Check for missing values
#sum(is.na(trainforest))

#Check the structure of the data we are working with to know what variables we have, etc.
#str(trainforest)
#summary(trainforest)

#-------------------------PLOTS : ---------------------------------------

#Histograms and boxplots of area with other predictors that are generally known to cause Forestfires are plott

#Indicates that 'temp' might be normally distributed because of symmetry
ggplot(trainforest, aes(x=temp)) + geom_histogram(binwidth=1, fill="pink", color="black") +
  labs(title="Histogram of Temperature", x="Temperature", y="Frequency")
```



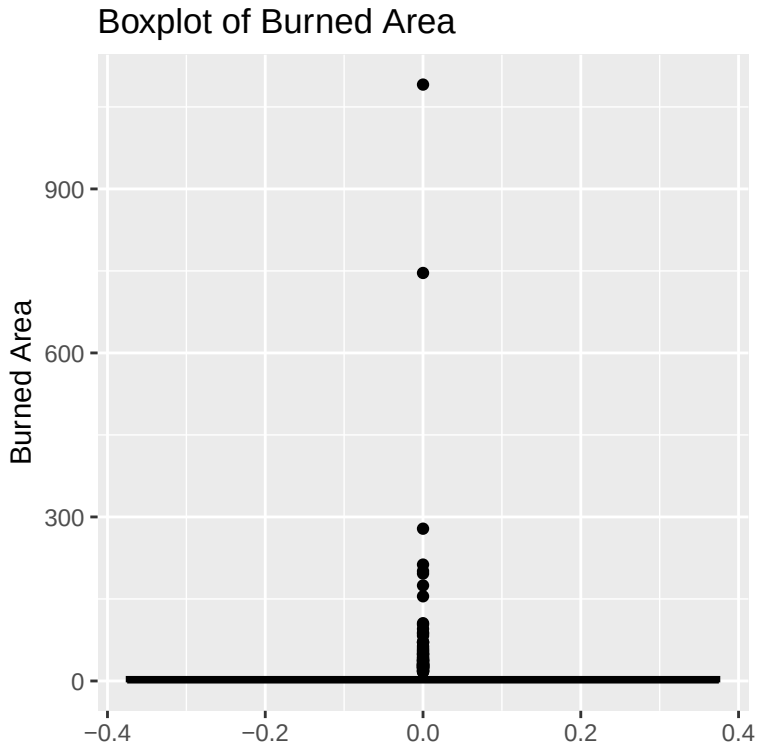
```
ggplot(trainforest, aes(y=temp)) + geom_boxplot(fill="yellow", color="black") +
  labs(title="Boxplot of Temperature", y="Temperature")
```



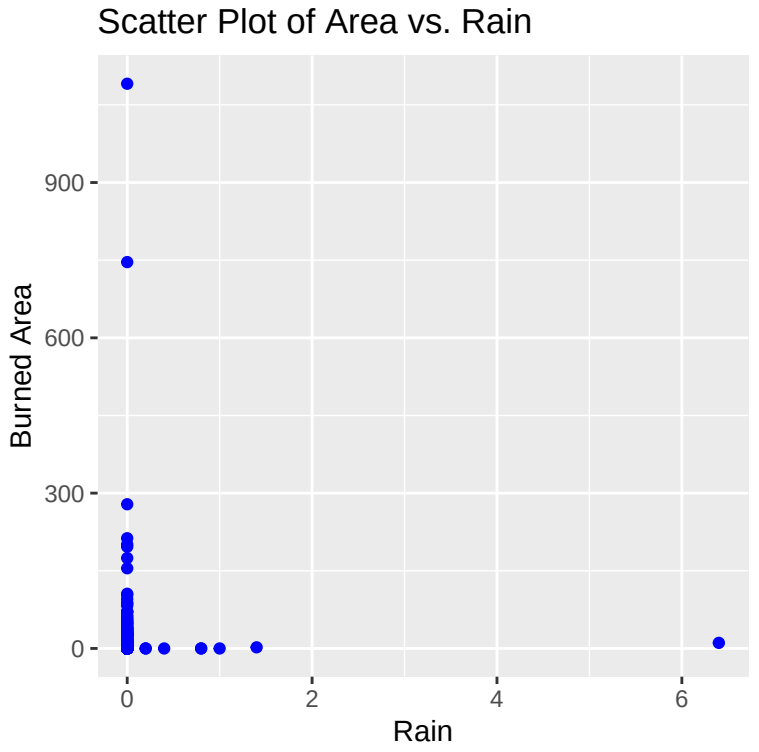

```
#Very left skewed, so not normal and may require transformation to make data analysis more insightful.
ggplot(trainforest, aes(x=area)) + geom_histogram(binwidth=20, fill="pink", color="black") +
  labs(title="Histogram of Burned Area", x="Burned Area", y="Frequency")
```



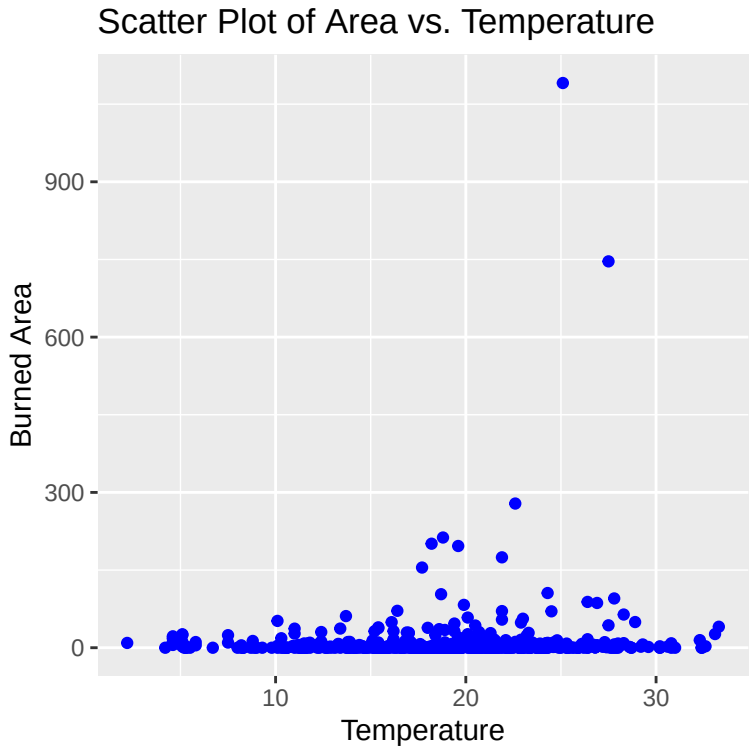
```
ggplot(trainforest, aes(y=area)) + geom_boxplot(fill="yellow", color="black") +
  labs(title="Boxplot of Burned Area", y="Burned Area")
```



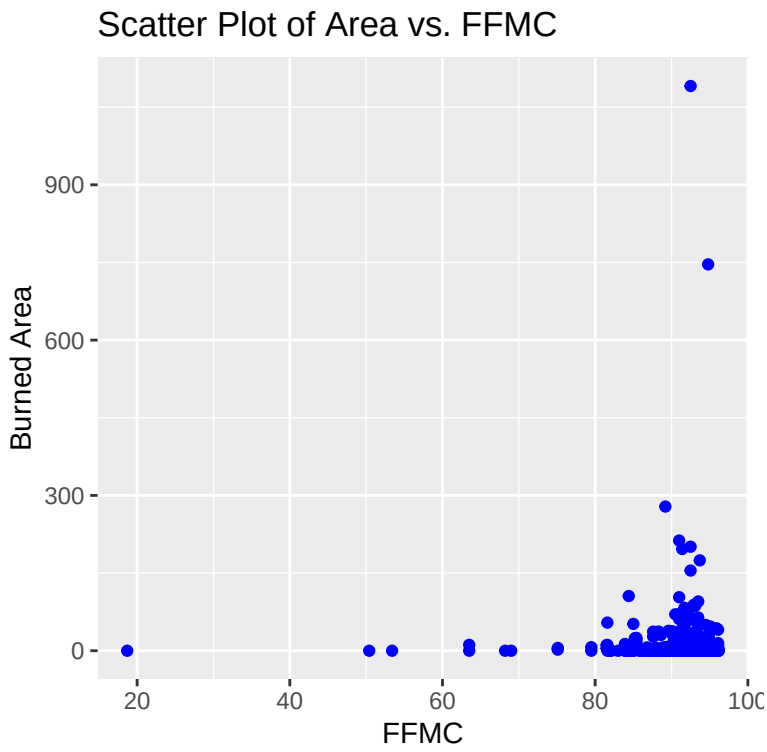

```
#Scatter plots to see realationship between variables like rain, temperature and FFMC
ggplot(trainforest, aes(x=rain, y=area)) + geom_point(color="blue") +
  labs(title="Scatter Plot of Area vs. Rain", x="Rain", y="Burned Area")
```



```
ggplot(trainforest, aes(x=temp, y=area)) + geom_point(color="blue") +
  labs(title="Scatter Plot of Area vs. Temperature", x="Temperature", y="Burned Area")
```



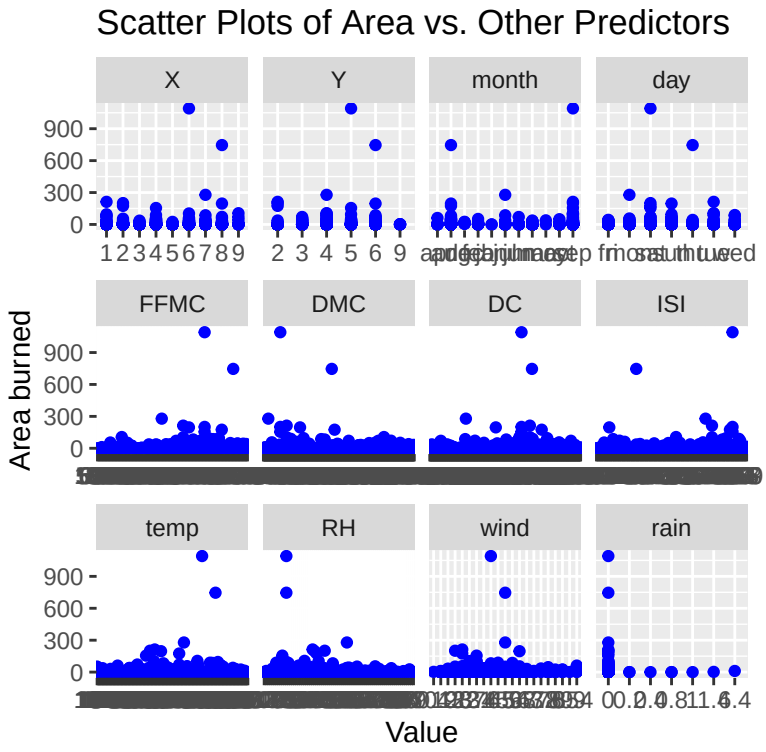

```
ggplot(trainforest, aes(x=FFMC, y=area)) + geom_point(color="blue") +
  labs(title="Scatter Plot of Area vs. FFMC", x="FFMC", y="Burned Area")
```



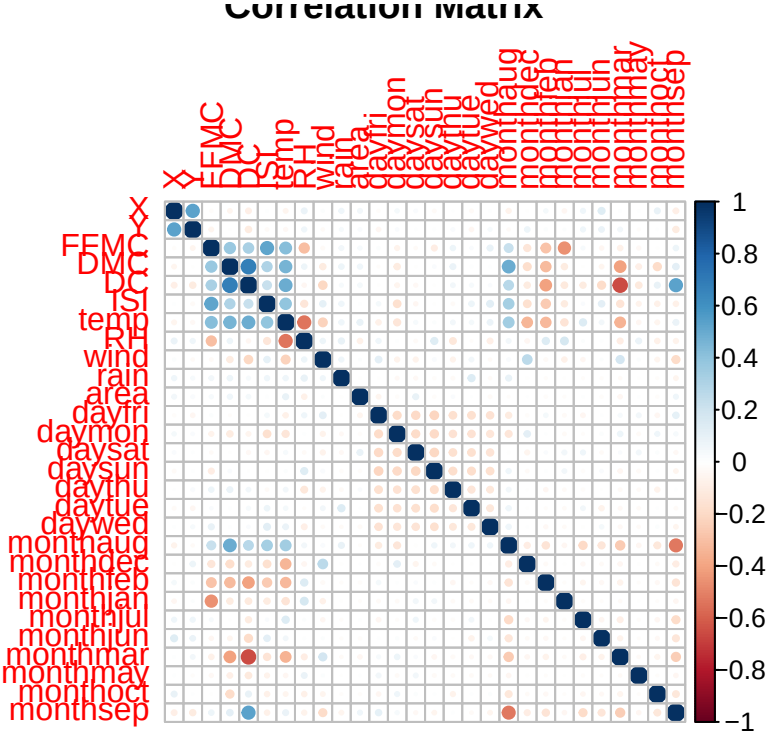
```
# Melting the dataset for faceting to see all histograms across all predictors -- makes it easier to analyse a
trainforest_melted <- melt(trainforest, id.vars = "area")
```

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
# Create faceted scatter plots to see realtionship to determine which predictor is strong for our response 'ar
ggplot(trainforest_melted, aes(x=value, y=area)) + geom_point(color="blue") +
  facet_wrap(~variable, scales="free_x") +
  labs(title="Scatter Plots of Area vs. Other Predictors", x="Value", y="Area burned")
```

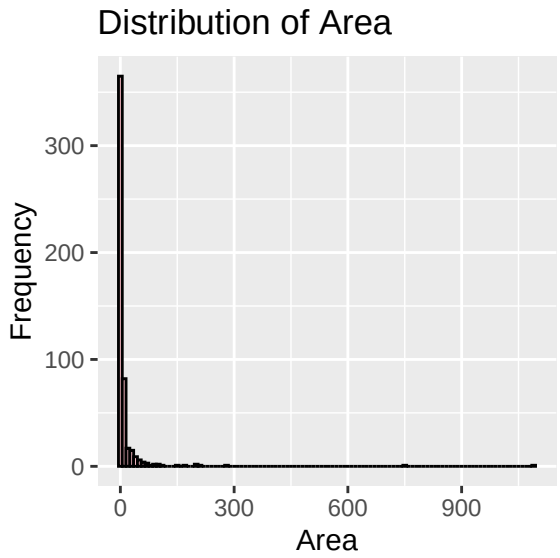



```
# Clean data to see correlation matrix to get more insight on how variables are correlated with each other to
trainforest_encoded <- cbind(trainforest, model.matrix(~day + month - 1, data=trainforest))
trainforest_encoded <- trainforest_encoded[, !(names(trainforest_encoded) %in% c("day", "month"))] # Remove or

# Calculate the correlation matrix and plot it
cor_matrix <- cor(trainforest_encoded)
corrplot(cor_matrix, method="circle", title="Correlation Matrix")
```

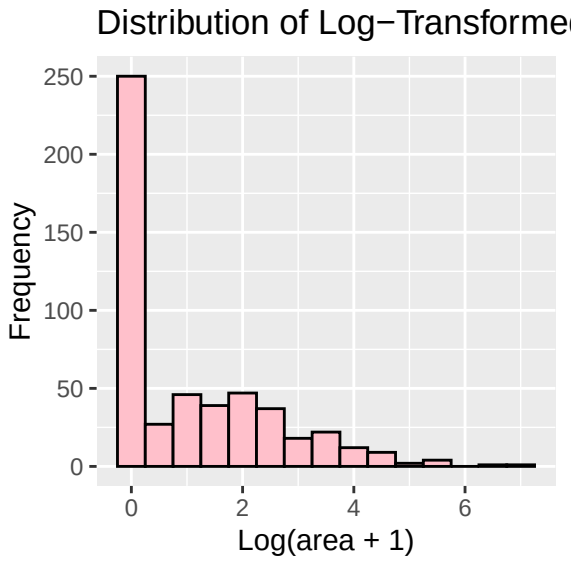


(b):

```
# Histogram of the original distribution of area
ggplot(trainforest, aes(x = area)) + geom_histogram(binwidth = 10, fill = "pink", color = "black") +
  labs(title = "Distribution of Area", x = "Area", y = "Frequency")
```




```
# Apply log transformation mentioned above
trainforest$area_new <- log(trainforest$area + 1)

# Histogram of the log-transformed area: Still skewed to the left but clearly the height of the bins have been
ggplot(trainforest, aes(x = area_new)) + geom_histogram(binwidth = 0.5, fill = "pink", color = "black") +
  labs(title = "Distribution of Log-Transformed Area", x = "Log(area + 1)", y = "Frequency")
```



(c):

```
# Build a simple linear regression model for each predictor variable to predict burned area. From now on, we w

model_temp <- lm(area_new ~ temp, trainforest)
summary(model_temp) #not sig
```

```
##
## Call:
## lm(formula = area_new ~ temp, data = trainforest)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2506 -1.0983 -0.7178  0.9151  5.8236
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.90190    0.20859   4.324 1.84e-05 ***
## temp         0.01076    0.01056   1.019    0.309
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.388 on 513 degrees of freedom
## Multiple R-squared:  0.002021,   Adjusted R-squared:  7.607e-05
## F-statistic: 1.039 on 1 and 513 DF,  p-value: 0.3085
```

```
model_RH <- lm(area_new ~ RH, trainforest)
#summary(model_RH) #not sig

model_X <- lm(area_new ~ X, trainforest)
#summary(model_X) #not sig

model_Y <- lm(area_new ~ Y, trainforest)
#summary(model_Y) #not sig
```

```r
model_FFMC <- lm(area_new ~ FFMC, trainforest)
#summary(model_FFMC) #not sig

model_DMC <- lm(area_new ~ DMC, trainforest)
#summary(model_DMC) #not sig

model_DC <- lm(area_new ~ DC, trainforest)
#summary(model_DC) #not sig

model_ISI <- lm(area_new ~ ISI, trainforest)
#summary(model_ISI) #not sig

model_month <- lm(area_new ~ month, trainforest)
summary(model_month) #SIG
```

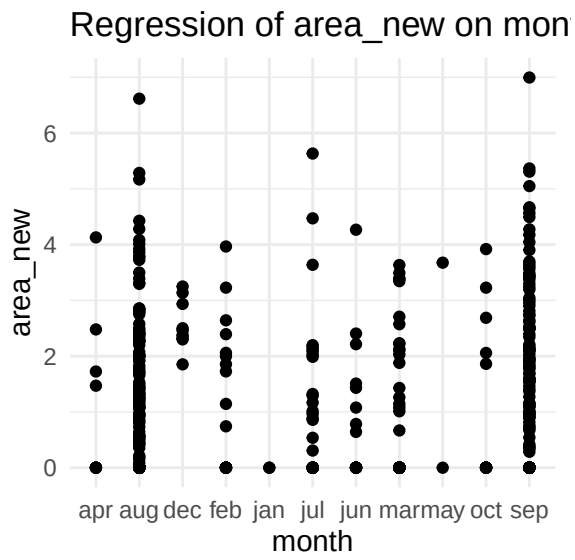
```
##
## Call:
## lm(formula = area_new ~ month, data = trainforest)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8379 -1.0223 -0.6036  0.8363  5.7211
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.089224   0.458635   2.375   0.0179 *
## monthaug    -0.066894   0.469777  -0.142   0.8868
## monthdec     1.482446   0.648607   2.286   0.0227 *
## monthfeb    -0.001390   0.552269  -0.003   0.9980
## monthjan    -1.089224   1.075593  -1.013   0.3117
## monthjul    -0.005583   0.519139  -0.011   0.9914
## monthjun    -0.246133   0.567191  -0.434   0.6645
## monthmar    -0.316659   0.495382  -0.639   0.5230
## monthmay     0.748674   1.075593   0.696   0.4867
## monthoct    -0.172172   0.580132  -0.297   0.7668
## monthsep     0.185314   0.470481   0.394   0.6938
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.376 on 504 degrees of freedom
## Multiple R-squared:  0.0371, Adjusted R-squared:  0.01799
## F-statistic: 1.942 on 10 and 504 DF,  p-value: 0.03777
```

```r
model_day <- lm(area_new ~ day, trainforest)
#summary(model_day) #not sig

model_wind <- lm(area_new ~ wind, trainforest)
#summary(model_wind) #not sig
```

```r
model_rain <- lm(area_new ~ rain, trainforest)
#summary(model_rain) #not sig

# Function to create scatter plots with regression lines
plot_regression <- function(predictor, response, data) {
  ggplot(data, aes_string(x = predictor, y = response)) +
    geom_point() +
    geom_smooth(method = "lm", col = "blue") +
    labs(title = paste("Regression of", response, "on", predictor),
         x = predictor, y = response) +
    theme_minimal()
}

# Create regression plots to see that month affects area burned.
#plot_regression("temp", "area_new", trainforest)
#plot_regression("RH", "area_new", trainforest)
#plot_regression("X", "area_new", trainforest)
#plot_regression("Y", "area_new", trainforest)
#plot_regression("FFMC", "area_new", trainforest)
#plot_regression("DMC", "area_new", trainforest)
#plot_regression("DC", "area_new", trainforest)
#plot_regression("ISI", "area_new", trainforest)
plot_regression("month", "area_new", trainforest)
```

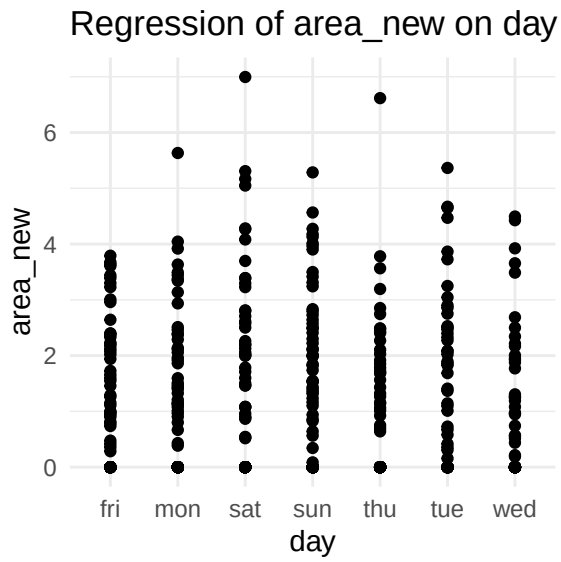
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



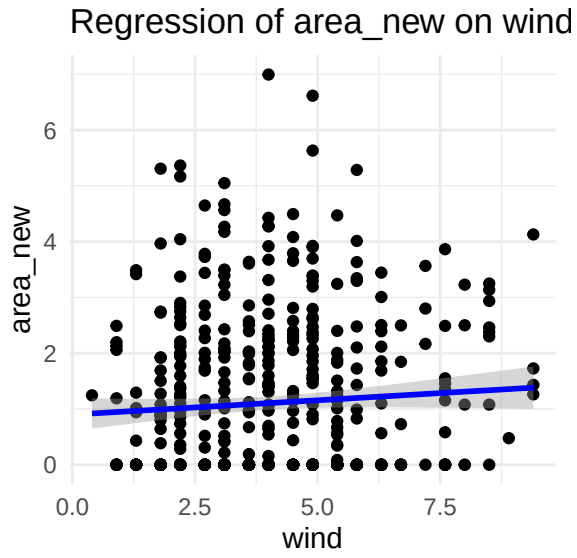
```
plot_regression("day", "area_new", trainforest)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
plot_regression("wind", "area_new", trainforest)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
#plot_regression("rain", "area_new", trainforest)
```

(d):

```
multi_all_lm <- lm(area_new ~ temp + X + Y + month  + DMC + DC + RH + ISI + day + FFMC + wind + rain , data =
summary(multi_all_lm)
```

```
##
```

```
## Call:
## lm(formula = area_new ~ temp + X + Y + month + DMC + DC + RH +
##     ISI + day + FFMC + wind + rain, data = trainforest)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9959 -1.0299 -0.4997  0.8434  5.2001
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.4918597  1.6460189  -0.299  0.76521
## temp         0.0340979  0.0221701   1.538  0.12469
## X            0.0520233  0.0321987   1.616  0.10681
## Y           -0.0355402  0.0609138  -0.583  0.55986
## monthaug     0.2719539  0.8189967   0.332  0.73999
## monthdec     2.1904025  0.7942663   2.758  0.00604 **
## monthfeb     0.1842463  0.5577891   0.330  0.74130
## monthjan    -0.3390164  1.2125469  -0.280  0.77991
## monthjul     0.0845391  0.7105195   0.119  0.90534
## monthjun    -0.2845657  0.6521446  -0.436  0.66277
## monthmar    -0.3385076  0.5033235  -0.673  0.50156
## monthmay     0.7101212  1.0944982   0.649  0.51677
## monthoct     0.7603841  0.9766298   0.779  0.43660
## monthsep     0.9360814  0.9176266   1.020  0.30818
## DMC          0.0039404  0.0018681   2.109  0.03543 *
## DC          -0.0018874  0.0012629  -1.494  0.13571
## RH           0.0007592  0.0062007   0.122  0.90260
## ISI         -0.0120540  0.0178924  -0.674  0.50083
## daymon       0.1479392  0.2253147   0.657  0.51175
## daysat       0.3170128  0.2163143   1.466  0.14342
## daysun       0.2188758  0.2104396   1.040  0.29881
## daythu       0.0824648  0.2387867   0.345  0.72998
## daytue       0.3244884  0.2339426   1.387  0.16606
## daywed       0.1337699  0.2462862   0.543  0.58728
## FFMC         0.0076552  0.0165489   0.463  0.64387
## wind         0.0564641  0.0382512   1.476  0.14055
## rain         0.0382507  0.2133982   0.179  0.85782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.373 on 488 degrees of freedom
## Multiple R-squared:  0.07149,    Adjusted R-squared:  0.02202
## F-statistic: 1.445 on 26 and 488 DF,  p-value: 0.07353
```

(e):

```
goodmodel_lm <- lm(area_new ~ temp + month + DMC + DC + RH + wind + rain +ISI , data = trainforest)
summary(goodmodel_lm)
```

```
##
```

```
## Call:
## lm(formula = area_new ~ temp + month + DMC + DC + RH + wind +
##     rain + ISI, data = trainforest)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7757 -1.0327 -0.5881  0.7971  5.4021
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.420344   0.704166   0.597  0.55082
## temp         0.036137   0.021674   1.667  0.09609 .
## monthaug     0.252652   0.801793   0.315  0.75281
## monthdec     2.125392   0.774178   2.745  0.00626 **
## monthfeb     0.144833   0.552663   0.262  0.79338
## monthjan    -0.648752   1.091940  -0.594  0.55270
## monthjul     0.111603   0.695775   0.160  0.87263
## monthjun    -0.297106   0.647483  -0.459  0.64653
## monthmar    -0.369484   0.495622  -0.745  0.45633
## monthmay     0.679747   1.083608   0.627  0.53075
## monthoct     0.845521   0.956215   0.884  0.37700
## monthsep     0.899853   0.897660   1.002  0.31662
## DMC          0.003943   0.001794   2.197  0.02846 *
## DC          -0.001911   0.001238  -1.543  0.12340
## RH           0.001187   0.005867   0.202  0.83978
## wind         0.054112   0.037415   1.446  0.14874
## rain         0.081603   0.209795   0.389  0.69747
## ISI         -0.008275   0.015784  -0.524  0.60033
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.369 on 497 degrees of freedom
## Multiple R-squared:  0.05957,    Adjusted R-squared:  0.0274
## F-statistic: 1.852 on 17 and 497 DF,  p-value: 0.02008
```

(g):

```r
# Set quantitative predictors 'temp', 'DMC', 'DC', 'RH', 'wind', 'rain', and 'ISI' to their sample means.
mean_temp <- mean(trainforest$temp, na.rm = TRUE)
mean_DMC <- mean(trainforest$DMC, na.rm = TRUE)
mean_DC <- mean(trainforest$DC, na.rm = TRUE)
mean_RH <- mean(trainforest$RH, na.rm = TRUE)
mean_wind <- mean(trainforest$wind, na.rm = TRUE)
mean_rain <- mean(trainforest$rain, na.rm = TRUE)
mean_ISI <- mean(trainforest$ISI, na.rm = TRUE)

# Set qualitative predictor 'month' to the most frequent category
most_freq_month <- names(sort(table(trainforest$month), decreasing = TRUE))[1]

# Create a new data frame with these values to include in predict function
```

```r
mean_freq_data <- data.frame(temp = mean_temp, month = most_freq_month, DMC = mean_DMC,
  DC = mean_DC, RH = mean_RH, wind = mean_wind, rain = mean_rain, ISI = mean_ISI)

# Use the model to make a prediction for burned area. We take exponent of the burned area prediction and subtr
predicted_area_new <- predict(goodmodel_lm, mean_freq_data)
predicted_area <- exp(predicted_area_new) - 1
predicted_area
```

```
##       1
## 1.56415
```

(h):

```r
# Residuals
residuals <- residuals(goodmodel_lm)

# Diagnostic plots
par(mfrow = c(2, 2))
plot(goodmodel_lm, which= 1)

# Identify outliers using Cook's distance
cooksd <- cooks.distance(goodmodel_lm)
plot(cooksd, pch="*", cex=2, main="Cook's Distance")
abline(h = 4/(nrow(trainforest)-length(coef(goodmodel_lm))), col="blue")
text(x=1:length(cooksd), y=cooksd, labels=ifelse(cooksd>4/(nrow(trainforest)-length(coef(goodmodel_lm))), name

# Identify influential points (Cook's distance > 4/n)
influential_points <- which(cooksd > (4 / nrow(trainforest)))
print(influential_points)
```

```
##  23 197 211 212 224 227 230 235 239 285 294 305 396 416 445 469 470 471 473 479
##  23 197 211 212 224 227 230 235 239 285 294 305 396 416 445 469 470 471 473 479
## 499 513
## 499 513
```

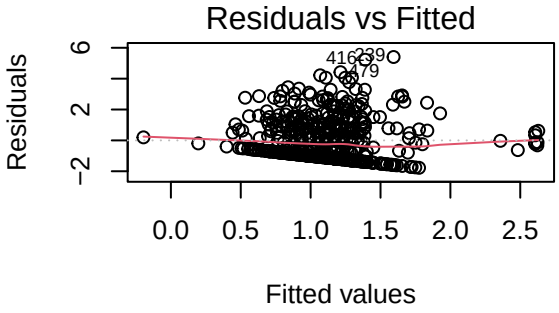
```r
# Remove influential points (outliers) and refit model after removing
trainforest_no_outliers <- trainforest[-influential_points, ]

# Refit the model
goodmodel_lm_no_outliers <- lm(area_new ~ temp + month + DMC + DC + RH + wind + rain + ISI, data = trainforest
summary(goodmodel_lm_no_outliers)
```

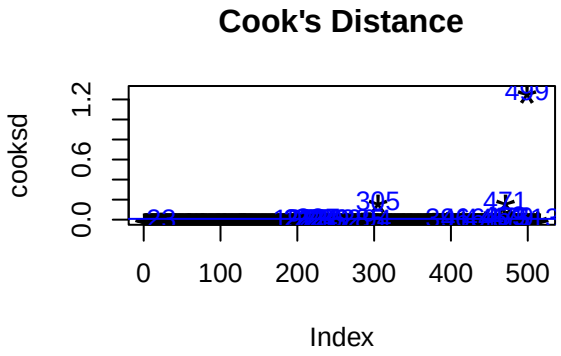
```
##
## Call:
## lm(formula = area_new ~ temp + month + DMC + DC + RH + wind +
##     rain + ISI, data = trainforest_no_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -1.6395 -0.8953 -0.5037  0.8214  4.3805
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.319067   0.718678  -0.444  0.65727
## temp         0.042943   0.020385   2.107  0.03567 *
## monthaug     0.137544   0.838827   0.164  0.86982
## monthdec     2.512351   0.786441   3.195  0.00149 **
## monthfeb     0.365509   0.581654   0.628  0.53005
## monthjan    -0.320310   1.020016  -0.314  0.75364
## monthjul    -0.125838   0.729755  -0.172  0.86317
## monthjun    -0.204650   0.669582  -0.306  0.76001
## monthmar     0.183503   0.537881   0.341  0.73313
## monthoct     0.269892   0.986919   0.273  0.78461
## monthsep     0.668989   0.935627   0.715  0.47495
## DMC          0.002805   0.001693   1.657  0.09825 .
## DC          -0.000809   0.001266  -0.639  0.52308
## RH           0.004796   0.005597   0.857  0.39200
## wind         0.028951   0.034919   0.829  0.40747
## rain        -0.833987   0.615457  -1.355  0.17604
## ISI         -0.003060   0.017018  -0.180  0.85739
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.232 on 476 degrees of freedom
## Multiple R-squared:  0.07837,    Adjusted R-squared:  0.04739
## F-statistic:  2.53 on 16 and 476 DF,  p-value: 0.0009483
```

```r
# Identify levels in the original model and refitted model
original_levels <- names(coef(goodmodel_lm))
refitted_levels <- names(coef(goodmodel_lm_no_outliers))

# Find missing levels in the refitted model
missing_levels <- setdiff(original_levels, refitted_levels)
#missing_levels

# Create a named vector of zeros for missing levels to combine the refitted coefficients with the missing coef
missing_coefficients <- setNames(rep(0, length(missing_levels)), missing_levels)
refitted_coefficients_final <- c(coef(goodmodel_lm_no_outliers), missing_coefficients)

# Ensure the order matches the original coefficients
refitted_coefficients_final <- refitted_coefficients_final[original_levels]

# Compare coefficients
comparison <- data.frame(
  Original = coef(goodmodel_lm),
  Refitted = refitted_coefficients_final
)
comparison
```

```
##                 Original      Refitted
## (Intercept)  0.420343710 -0.3190669609
## temp         0.036136806  0.0429432940
## monthaug     0.252651975  0.1375436326
## monthdec     2.125392278  2.5123513426
## monthfeb     0.144833207  0.3655091368
## monthjan    -0.648752363 -0.3203097939
## monthjul     0.111603239 -0.1258380029
## monthjun    -0.297106239 -0.2046502930
## monthmar    -0.369483704  0.1835032524
## monthmay     0.679746782  0.0000000000
## monthoct     0.845521248  0.2698920735
## monthsep     0.899853116  0.6689887108
## DMC          0.003942744  0.0028048333
## DC          -0.001910552 -0.0008090071
## RH           0.001186755  0.0047957425
## wind         0.054111949  0.0289509727
## rain         0.081603230 -0.8339872767
## ISI         -0.008275069 -0.0030595695
```

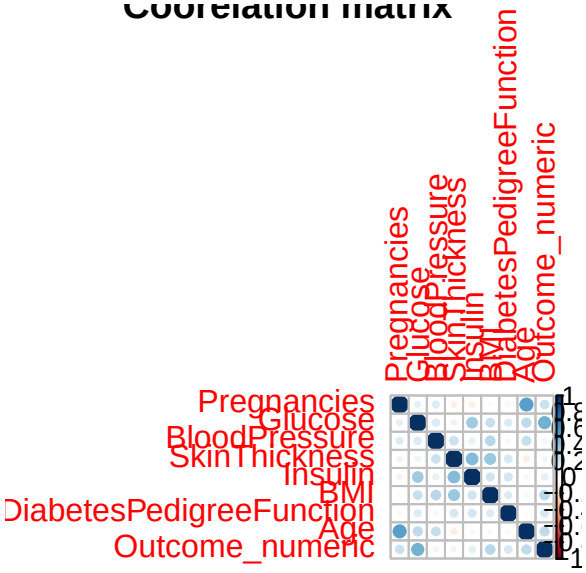




2

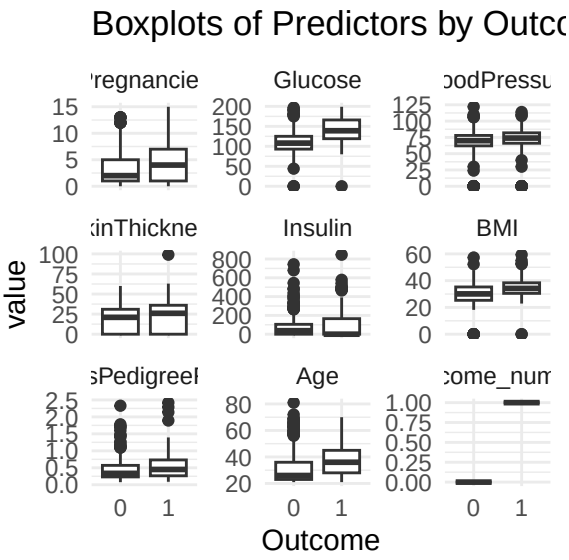
(a): Exploratory analysis

```r
# Read the data
traindiabetes <- read.csv("diabetes_train.csv")
testdiabetes <- read.csv("diabetes_test.csv")

# Check for missing values
#sum(is.na(traindiabetes))

#Convert outcome to a factor variable because it is qualitative
traindiabetes$Outcome <- as.factor(traindiabetes$Outcome)
#table(traindiabetes$Outcome)

# Convert Outcome to numeric in order to calculate correlation matrix and plot
traindiabetes$Outcome_numeric <- as.numeric(traindiabetes$Outcome) - 1

# Calculate the correlation matrix including Outcome_numeric
cor_matrix_with_outcome <- cor(traindiabetes[, sapply(traindiabetes, is.numeric)])
corrplot(cor_matrix_with_outcome, method = "circle",title = "Coorelation matrix ")
```




```r
#pregnancies, glucose, bmi, age, blood pressure look like they are strong predictors of diabetes

# Boxplots of various predictors (inferred from the correlation plot) against Outcome
#ggplot(traindiabetes, aes(x = Outcome, y = Pregnancies)) + geom_boxplot() + ggtitle("Boxplot of Pregnancies b
#ggplot(traindiabetes, aes(x = Outcome, y = Glucose)) + geom_boxplot() + ggtitle("Boxplot of Glucose by Outcom
#ggplot(traindiabetes, aes(x = Outcome, y = BMI)) + geom_boxplot() + ggtitle("Boxplot of BMI by Outcome")
#ggplot(traindiabetes, aes(x = Outcome, y = Age)) + geom_boxplot() + ggtitle("Boxplot of Age by Outcome")
#ggplot(traindiabetes, aes(x = Outcome, y = BloodPressure)) + geom_boxplot() + ggtitle("Boxplot of BP by Outco
#ggplot(traindiabetes, aes(x = Outcome, y = Insulin)) + geom_boxplot() + ggtitle("Boxplot of Insulin by Outcom

# Melt the data for faceting to avoid having many boxplots.
melted_data_facetwrap <- melt(traindiabetes, id.vars = "Outcome")

# Create the faceted boxplot of predictors vs outcome
ggplot(melted_data_facetwrap, aes(x = Outcome, y = value)) +
  geom_boxplot() +
```

```
  facet_wrap(~ variable, scales = "free_y") +
  ggtitle("Boxplots of Predictors by Outcome") +
  theme_minimal()
```




```
#------------------STANDARDIZING TRAINING AND TEST VARIABLES--------------------------------
# Function to standardize the training and test variable
standardize <- function(x) {
  (x - mean(x)) / sd(x)
}

# Standardize all columns except "Outcome"
standardize_trainD <- traindiabetes %>%
  mutate(across(-Outcome, standardize))

standardize_testD <- testdiabetes %>%
  mutate(across(-Outcome, standardize))

# View the standardized data
#standardize_trainD
#standardize_testD
```

(b):

```
#LDA on training and test data for all predictors
trainD_lda <- lda(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPe
#trainD_lda

testD_lda <- lda(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPed
#testD_lda

# Get predictions for test and training data
lda_pred_testD <- predict(testD_lda, standardize_trainD)$class
lda_pred_trainD <- predict(trainD_lda, standardize_trainD)$class
```

```r
# Confusion matrix for training and test data using the caret package and using confusionMatrix function
conf_matrix_trainD <- confusionMatrix(lda_pred_trainD, standardize_trainD$Outcome)
conf_matrix_trainD
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 405 106
##          1  54 135
##
##                Accuracy : 0.7714
##                  95% CI : (0.7385, 0.8021)
##     No Information Rate : 0.6557
##     P-Value [Acc > NIR] : 1.815e-11
##
##                   Kappa : 0.4664
##
##  Mcnemar's Test P-Value : 5.533e-05
##
##             Sensitivity : 0.8824
##             Specificity : 0.5602
##          Pos Pred Value : 0.7926
##          Neg Pred Value : 0.7143
##              Prevalence : 0.6557
##          Detection Rate : 0.5786
##    Detection Prevalence : 0.7300
##       Balanced Accuracy : 0.7213
##
##        'Positive' Class : 0
##
```

```r
conf_matrix_testD <- confusionMatrix(lda_pred_testD, standardize_trainD$Outcome)
conf_matrix_testD
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 378  94
##          1  81 147
##
##                Accuracy : 0.75
##                  95% CI : (0.7162, 0.7817)
##     No Information Rate : 0.6557
##     P-Value [Acc > NIR] : 4.55e-08
##
##                   Kappa : 0.4391
##
##  Mcnemar's Test P-Value : 0.3643
```

```
##
##             Sensitivity : 0.8235
##             Specificity : 0.6100
##          Pos Pred Value : 0.8008
##          Neg Pred Value : 0.6447
##              Prevalence : 0.6557
##          Detection Rate : 0.5400
##    Detection Prevalence : 0.6743
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 0
##
```

```r
# Misclassification rate for training and test data
misclass_rate_trainD <- 1 - conf_matrix_trainD$overall['Accuracy']
misclass_rate_trainD
```

```
##  Accuracy
## 0.2285714
```

```r
misclass_rate_testD <- 1 - conf_matrix_testD$overall['Accuracy']
misclass_rate_testD
```

```
## Accuracy
##     0.25
```

(c):

```r
#QDA on training and test data for all predictors
trainD_qda <- qda(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPe
#trainD_qda

testD_qda <-  qda(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPe
#testD_qda

# Get predictions for test data and training data
qda_pred_testD <- predict(testD_qda, standardize_trainD)$class
qda_pred_trainD <- predict(trainD_qda, standardize_trainD)$class

# Confusion matrix for training data and test data
conf_matrix_trainD_qda <- confusionMatrix(qda_pred_trainD, standardize_trainD$Outcome)
conf_matrix_trainD_qda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 397  99
##          1  62 142
##
```

```
##                Accuracy : 0.77
##                  95% CI : (0.737, 0.8007)
##     No Information Rate : 0.6557
##     P-Value [Acc > NIR] : 3.23e-11
##
##                   Kappa : 0.4713
##
##  Mcnemar's Test P-Value : 0.004551
##
##             Sensitivity : 0.8649
##             Specificity : 0.5892
##          Pos Pred Value : 0.8004
##          Neg Pred Value : 0.6961
##              Prevalence : 0.6557
##          Detection Rate : 0.5671
##    Detection Prevalence : 0.7086
##       Balanced Accuracy : 0.7271
##
##        'Positive' Class : 0
##
```

```
conf_matrix_testD_qda <- confusionMatrix(qda_pred_testD, standardize_trainD$Outcome)
conf_matrix_testD_qda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 285  80
##          1 174 161
##
##                Accuracy : 0.6371
##                  95% CI : (0.6003, 0.6728)
##     No Information Rate : 0.6557
##     P-Value [Acc > NIR] : 0.8584
##
##                   Kappa : 0.2645
##
##  Mcnemar's Test P-Value : 5.368e-09
##
##             Sensitivity : 0.6209
##             Specificity : 0.6680
##          Pos Pred Value : 0.7808
##          Neg Pred Value : 0.4806
##              Prevalence : 0.6557
##          Detection Rate : 0.4071
##    Detection Prevalence : 0.5214
##       Balanced Accuracy : 0.6445
##
##        'Positive' Class : 0
```

```
##
```

```
# Misclassification rate for training data and test data
misclass_rate_trainD_qda <- 1 - conf_matrix_trainD_qda$overall['Accuracy']
misclass_rate_trainD_qda
```

```
## Accuracy
##     0.23
```

```
misclass_rate_testD_qda <- 1 - conf_matrix_testD_qda$overall['Accuracy']
misclass_rate_testD_qda
```

```
##  Accuracy
## 0.3628571
```

(d):

```
#Naive Bayes for training and test data
nb_fit_trainD <- naiveBayes(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + 
#nb_fit_trainD

nb_fit_testD <- naiveBayes(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + D
#nb_fit_testD

# Get predicted classes for test and training data
nb_pred_testD <- predict(nb_fit_testD, standardize_trainD)
#table(nb_pred_testD, standardize_trainD$Outcome)

nb_pred_trainD <- predict(nb_fit_trainD, standardize_trainD)
#table(nb_pred_trainD, standardize_trainD$Outcome)

# Confusion matrix for training data and test data
conf_matrix_trainD_nb <- confusionMatrix(nb_pred_trainD, standardize_trainD$Outcome)
conf_matrix_trainD_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 387  95
##          1  72 146
##
##                Accuracy : 0.7614
##                  95% CI : (0.7281, 0.7926)
##     No Information Rate : 0.6557
##     P-Value [Acc > NIR] : 8.681e-10
##
##                   Kappa : 0.4594
##
##  Mcnemar's Test P-Value : 0.08868
```

```
##
##             Sensitivity : 0.8431
##             Specificity : 0.6058
##          Pos Pred Value : 0.8029
##          Neg Pred Value : 0.6697
##              Prevalence : 0.6557
##          Detection Rate : 0.5529
##    Detection Prevalence : 0.6886
##       Balanced Accuracy : 0.7245
##
##        'Positive' Class : 0
##
```

```
conf_matrix_testD_nb <- confusionMatrix(nb_pred_testD, standardize_trainD$Outcome)
conf_matrix_testD_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 356  89
##          1 103 152
##
##                Accuracy : 0.7257
##                  95% CI : (0.691, 0.7585)
##     No Information Rate : 0.6557
##     P-Value [Acc > NIR] : 4.334e-05
##
##                   Kappa : 0.4008
##
##  Mcnemar's Test P-Value : 0.3481
##
##             Sensitivity : 0.7756
##             Specificity : 0.6307
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.5961
##              Prevalence : 0.6557
##          Detection Rate : 0.5086
##    Detection Prevalence : 0.6357
##       Balanced Accuracy : 0.7032
##
##        'Positive' Class : 0
##
```

```
# Misclassification rate for training data and test data
misclass_rate_trainD_nb <- 1 - conf_matrix_trainD_nb$overall['Accuracy']
misclass_rate_trainD_nb
```

```
##  Accuracy
## 0.2385714
```

```r
misclass_rate_testD_nb <- 1 - conf_matrix_testD_nb$overall['Accuracy']
misclass_rate_testD_nb
```

```
##  Accuracy
## 0.2742857
```

(e):

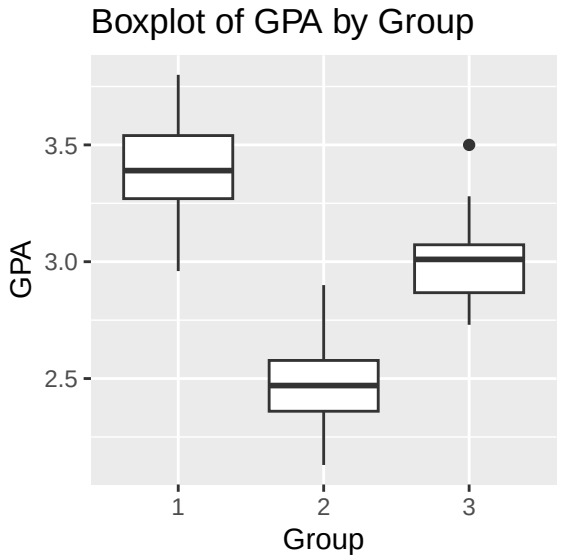
```r
set.seed(1)

#To do KNN, we use code from Mini Project 1

#Set outcome as response variable
train_resp <- as.numeric(as.factor(traindiabetes$Outcome))
test_resp <- as.numeric(as.factor(testdiabetes$Outcome))

#create a vector containing predictors for both training and test data
trainingset.X <- cbind(standardize_trainD$Pregnancies, standardize_trainD$Glucose, standardize_trainD$BloodPre

testset.X <- cbind(standardize_testD$Pregnancies, standardize_testD$Glucose, standardize_testD$BloodPressure,

# Now, we fit KNN for several values of K (1-100)
ks <- c(seq(1, 30, by = 1), seq(35, 100, by = 5))
nks <- length(ks)

# Store training and test error rate for each value of k ranging from 1 to 100.
err.rate.train <- numeric(length = nks)
err.rate.test <- numeric(length = nks)

#Set the names of the error rate so that they correspond to the right value of k
names(err.rate.train) <- names(err.rate.test) <- ks
for (i in seq(along = ks)) {
  mod.train <- knn(trainingset.X, trainingset.X, train_resp, k = ks[i])    # KNN for training data
  mod.test <- knn(trainingset.X, testset.X, train_resp, k = ks[i])         # KNN for test data
  err.rate.train[i] <- 1 - sum(mod.train == train_resp)/length(train_resp) # Training error rate (1 - accuracy
  err.rate.test[i] <- 1 - sum(mod.test == test_resp)/length(test_resp)      # Test error rate
}

# Calculate optimal value of k and obtain its corresponding test and training error rates
result <- data.frame(ks, err.rate.train, err.rate.test)
result[err.rate.test == min(result$err.rate.test), ]
```

```
##    ks err.rate.train err.rate.test
## 12 12      0.1942857          0.25
## 14 14      0.2014286          0.25
## 25 25      0.2200000          0.25
```

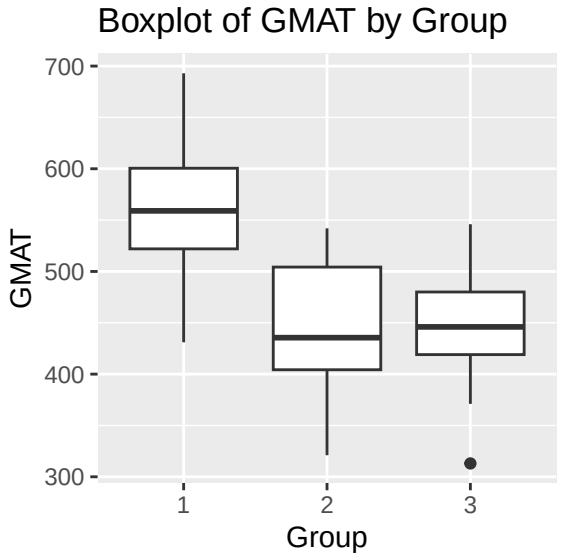
3:

(a): Exploratory data analysis

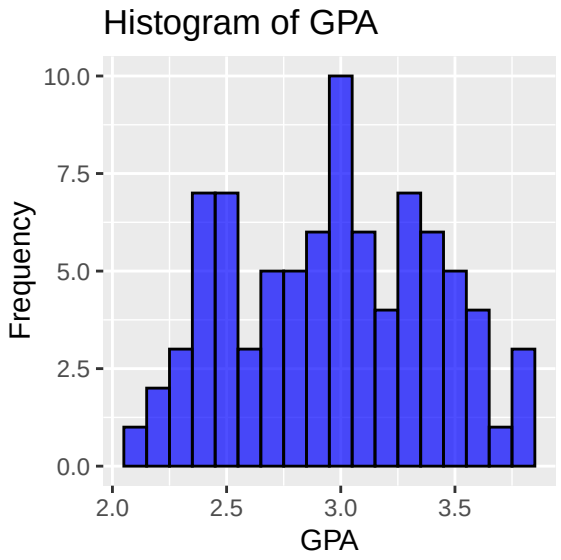
```r
#Read the data and convert qualitative variable group to factor
admission <- read.csv("admission.csv")
admission$Group <- as.factor(admission$Group)

#Obtain test and training data from original dataset according to specifications in the question
admit <- which(admission$Group==1)
dont_admit <- which(admission$Group==2)
borderline <- which(admission$Group==3)

admission_test <- rbind(admission[admit,][1:5,], admission[dont_admit,][1:5,], admission[borderline,][1:5,]) #
admission_train <- rbind(admission[admit,][-(1:5),], admission[dont_admit,][-(1:5),], admission[borderline,][-

# Boxplot for GPA by Group
ggplot(admission, aes(x = Group, y = GPA)) + geom_boxplot() + labs(title = "Boxplot of GPA by Group", x = "Gro
```



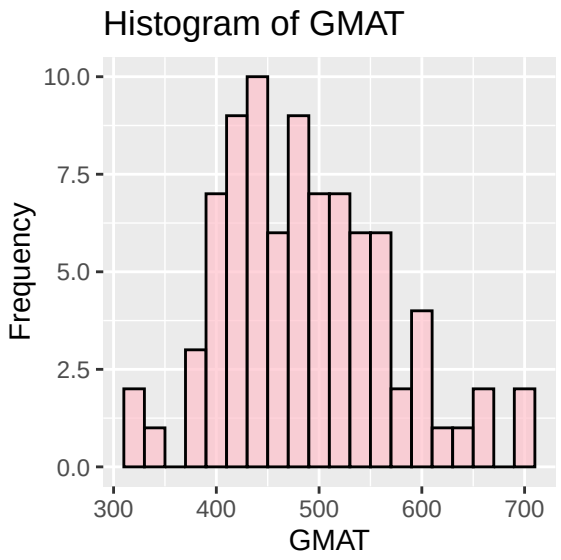

```r
# Boxplot for GMAT by Group
ggplot(admission, aes(x = Group, y = GMAT)) + geom_boxplot() + labs(title = "Boxplot of GMAT by Group", x = "G
```



```r
# Histogram for GPA
ggplot(admission, aes(x = GPA)) + geom_histogram(binwidth = 0.1, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histogram of GPA", x = "GPA", y = "Frequency")
```



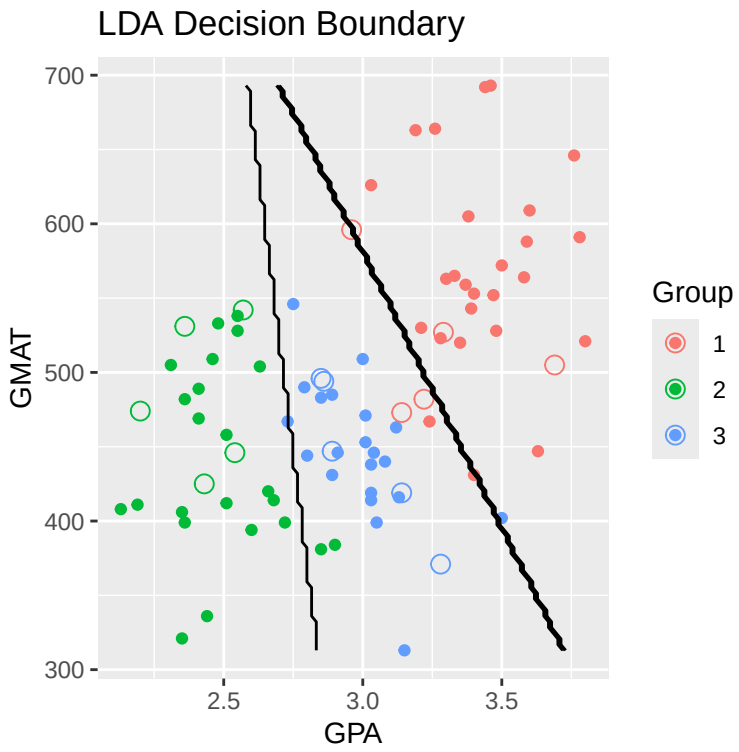

```r
# Histogram for GMAT
ggplot(admission, aes(x = GMAT)) + geom_histogram(binwidth = 20, fill = "pink", color = "black", alpha = 0.7) +
  labs(title = "Histogram of GMAT", x = "GMAT", y = "Frequency")
```



(b):

```r
#LDA on training and test data
admit_lda <- lda(Group ~ GPA + GMAT, admission_train)
admit_test_lda <- lda(Group ~ GPA + GMAT, admission_test)

# Create a grid of values for GPA and GMAT to predict the decision boundary
gpa_range <- seq(min(admission$GPA), max(admission$GPA), length.out = 100)
gmat_range <- seq(min(admission$GMAT), max(admission$GMAT), length.out = 100)
grid <- expand.grid(GPA = gpa_range, GMAT = gmat_range)
```

```r
# Predict the class for each data point in the grid created above
grid$Group <- predict(admit_lda, newdata = grid)$class

# Plot the decision boundary for training data
ggplot(admission_train, aes(x = GPA, y = GMAT, color = Group)) +
  geom_point() +
  geom_point(data = admission_test, shape = 1, size = 3) +
  geom_contour(data = grid, aes(z = as.numeric(Group)), color = "black", bins = 3) + #decision boundary
  labs(title = "LDA Decision Boundary", x = "GPA", y = "GMAT")
```




```r
# Get predictions for test data and training data
admit_lda_pred_test <- predict(admit_test_lda, admission_train)$class
admit_lda_pred_train <- predict(admit_lda, admission_train)$class

# Confusion matrix for training data and test data
conf_matrix_admTrain <- confusionMatrix(admit_lda_pred_train, admission_train$Group)
conf_matrix_admTrain
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 24  0  1
##          2  0 21  1
##          3  2  2 19
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9143
##                  95% CI : (0.8227, 0.9679)
##     No Information Rate : 0.3714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8712
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9231   0.9130   0.9048
## Specificity            0.9773   0.9787   0.9184
## Pos Pred Value         0.9600   0.9545   0.8261
## Neg Pred Value         0.9556   0.9583   0.9574
## Prevalence             0.3714   0.3286   0.3000
## Detection Rate         0.3429   0.3000   0.2714
## Detection Prevalence   0.3571   0.3143   0.3286
## Balanced Accuracy      0.9502   0.9459   0.9116
```

```
conf_matrix_admTest <- confusionMatrix(admit_lda_pred_test, admission_train$Group)
conf_matrix_admTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 26  0  1
##          2  0 21  0
##          3  0  2 20
##
## Overall Statistics
##
##                Accuracy : 0.9571
##                  95% CI : (0.8798, 0.9911)
##     No Information Rate : 0.3714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9354
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9130   0.9524
## Specificity            0.9773   1.0000   0.9592
## Pos Pred Value         0.9630   1.0000   0.9091
## Neg Pred Value         1.0000   0.9592   0.9792
```

```
## Prevalence             0.3714   0.3286   0.3000
## Detection Rate         0.3714   0.3000   0.2857
## Detection Prevalence   0.3857   0.3000   0.3143
## Balanced Accuracy      0.9886   0.9565   0.9558
```

```r
# Misclassification rate for training and test data
misclass_rate_admTrain <- 1 - conf_matrix_admTrain$overall['Accuracy']
misclass_rate_admTrain
```

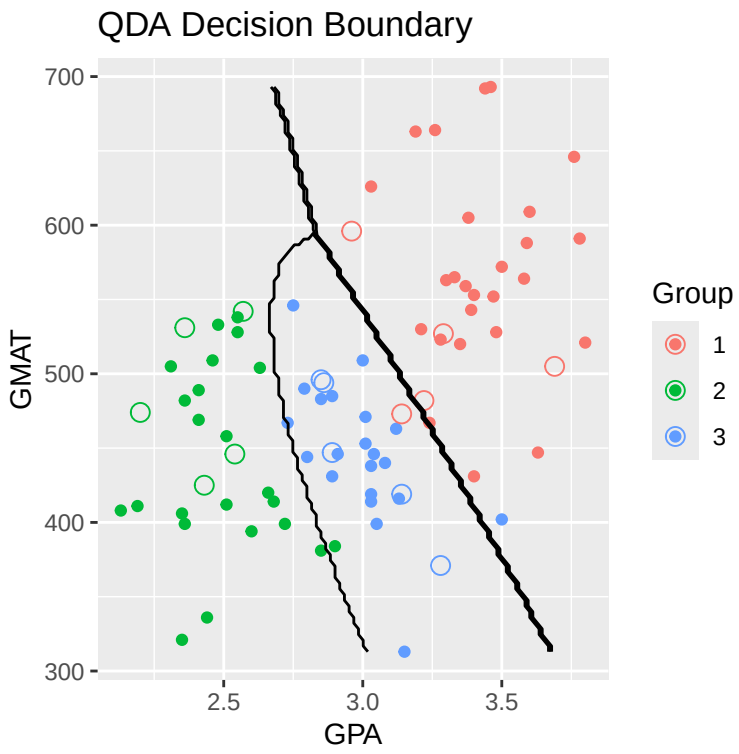
```
##   Accuracy
## 0.08571429
```

```r
misclass_rate_admTest <- 1 - conf_matrix_admTest$overall['Accuracy']
misclass_rate_admTest
```

```
##   Accuracy
## 0.04285714
```

(c):

```r
#QDA on training and test data
admit_qda <- qda(Group ~ GPA + GMAT, admission_train)
admit_test_qda <- qda(Group ~ GPA + GMAT, admission_test)

# Predict the class for each data point in the grid using the grid created for LDA in 2(b) above
grid$Group <- predict(admit_qda, newdata = grid)$class

# Plot the decision boundary of training data
ggplot(admission_train, aes(x = GPA, y = GMAT, color = Group)) +
  geom_point() + # Plot training data points
  geom_point(data = admission_test, shape = 1, size = 3) +
  geom_contour(data = grid, aes(z = as.numeric(Group)), color = "black", bins = 3) + #decision boundary
  labs(title = "QDA Decision Boundary", x = "GPA", y = "GMAT")
```




```
# Get predictions for test and training data
admit_qda_pred_test <- predict(admit_test_qda, admission_train)$class
admit_qda_pred_train <- predict(admit_qda, admission_train)$class

# Confusion matrix for training  and test data
conf_matrix_admTrain_qda <- confusionMatrix(admit_qda_pred_train, admission_train$Group)
conf_matrix_admTrain_qda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 26  0  1
##          2  0 22  0
##          3  0  1 20
##
## Overall Statistics
##
##                Accuracy : 0.9714
##                  95% CI : (0.9006, 0.9965)
##     No Information Rate : 0.3714
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9569
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9565   0.9524
## Specificity            0.9773   1.0000   0.9796
## Pos Pred Value         0.9630   1.0000   0.9524
## Neg Pred Value         1.0000   0.9792   0.9796
## Prevalence             0.3714   0.3286   0.3000
## Detection Rate         0.3714   0.3143   0.2857
## Detection Prevalence   0.3857   0.3143   0.3000
## Balanced Accuracy      0.9886   0.9783   0.9660
```

```
conf_matrix_admTest_qda <- confusionMatrix(admit_qda_pred_test, admission_train$Group)
conf_matrix_admTest_qda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 26  1  4
##          2  0 22  2
##          3  0  0 15
##
## Overall Statistics
##
##                Accuracy : 0.9
##                  95% CI : (0.8048, 0.9588)
##     No Information Rate : 0.3714
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8482
##
##  Mcnemar's Test P-Value : 0.0719
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9565   0.7143
## Specificity            0.8864   0.9574   1.0000
## Pos Pred Value         0.8387   0.9167   1.0000
## Neg Pred Value         1.0000   0.9783   0.8909
## Prevalence             0.3714   0.3286   0.3000
## Detection Rate         0.3714   0.3143   0.2143
## Detection Prevalence   0.4429   0.3429   0.2143
## Balanced Accuracy      0.9432   0.9570   0.8571
```

```
# Misclassification rate for training and test data
misclass_rate_admTrain_qda <- 1 - conf_matrix_admTrain_qda$overall['Accuracy']
misclass_rate_admTrain_qda
```

```
##   Accuracy
## 0.02857143
```

```r
misclass_rate_admTest_qda <- 1 - conf_matrix_admTest_qda$overall['Accuracy']
misclass_rate_admTest_qda
```

```
## Accuracy
##      0.1
```