Predictive maintenane

February 28, 2018

```r
knitr::opts_chunk$set(echo = TRUE, tidy.opts = list(width.cutoff = 60),
    tidy = TRUE, warning = FALSE, message = FALSE, echo = TRUE,
    tidy = TRUE, size = "small")

suppressPackageStartupMessages({
    library(knitr)
    library(dplyr)
    library(ggplot2)
    library(corrplot)
    library(caret)
    library(caTools)
    library(rpart)
    library(e1071)
    library(RColorBrewer)
    library(rattle)
    library(party)
    library(partykit)
    library(ISLR)
    library(class)
})
```

1. Data Munging In this section, we will load the data, and slice and dice it to see if there are any treatments that we need to do on the dataset. This is an important step to make the data good enough to be modelled.

Descriptive statistics

```r
# Load the data
pred_data <- read.csv("maintenance_data.csv", header = TRUE)

head(pred_data)
```

```
##   lifetime broken pressureInd moistureInd temperatureInd  team  provider
## 1       56      0    92.17885   104.23020       96.51716 TeamA Provider4
## 2       81      1    72.07594   103.06570       87.27106 TeamC Provider4
## 3       60      0    96.27225    77.80138      112.19617 TeamA Provider1
## 4       86      1    94.40646   108.49361       72.02537 TeamC Provider2
## 5       34      0    97.75290    99.41349      103.75627 TeamB Provider1
## 6       30      0    87.67880   115.71226       89.79210 TeamA Provider1
```

```r
str(pred_data)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ lifetime      : int  56 81 60 86 34 30 68 65 23 81 ...
##  $ broken        : int  0 1 0 1 0 0 0 1 0 1 ...
##  $ pressureInd   : num  92.2 72.1 96.3 94.4 97.8 ...
##  $ moistureInd   : num  104.2 103.1 77.8 108.5 99.4 ...
##  $ temperatureInd: num  96.5 87.3 112.2 72 103.8 ...
##  $ team          : Factor w/ 3 levels "TeamA","TeamB",..: 1 3 1 3 2 1 2 2 2 3 ...
##  $ provider      : Factor w/ 4 levels "Provider1","Provider2",..: 4 4 1 2 1 1 2 3 2 4 ...
```

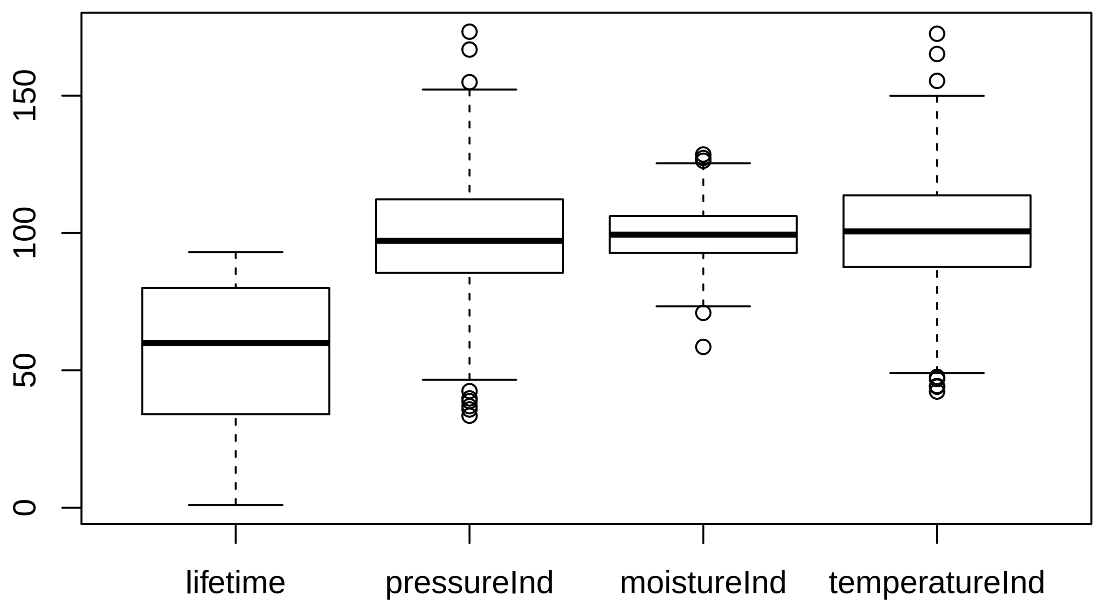
```r
summary(pred_data)
```

```
##     lifetime        broken        pressureInd      moistureInd
##  Min.   : 1.0   Min.   :0.000   Min.   : 33.48   Min.   : 58.55
##  1st Qu.:34.0   1st Qu.:0.000   1st Qu.: 85.56   1st Qu.: 92.77
##  Median :60.0   Median :0.000   Median : 97.22   Median : 99.43
##  Mean   :55.2   Mean   :0.397   Mean   : 98.60   Mean   : 99.38
##  3rd Qu.:80.0   3rd Qu.:1.000   3rd Qu.:112.25   3rd Qu.:106.12
##  Max.   :93.0   Max.   :1.000   Max.   :173.28   Max.   :128.60
##  temperatureInd      team          provider
##  Min.   : 42.28   TeamA:336   Provider1:254
##  1st Qu.: 87.68   TeamB:356   Provider2:266
##  Median :100.59   TeamC:308   Provider3:242
##  Mean   :100.63               Provider4:238
##  3rd Qu.:113.66
##  Max.   :172.54
```

Understanding continuous data

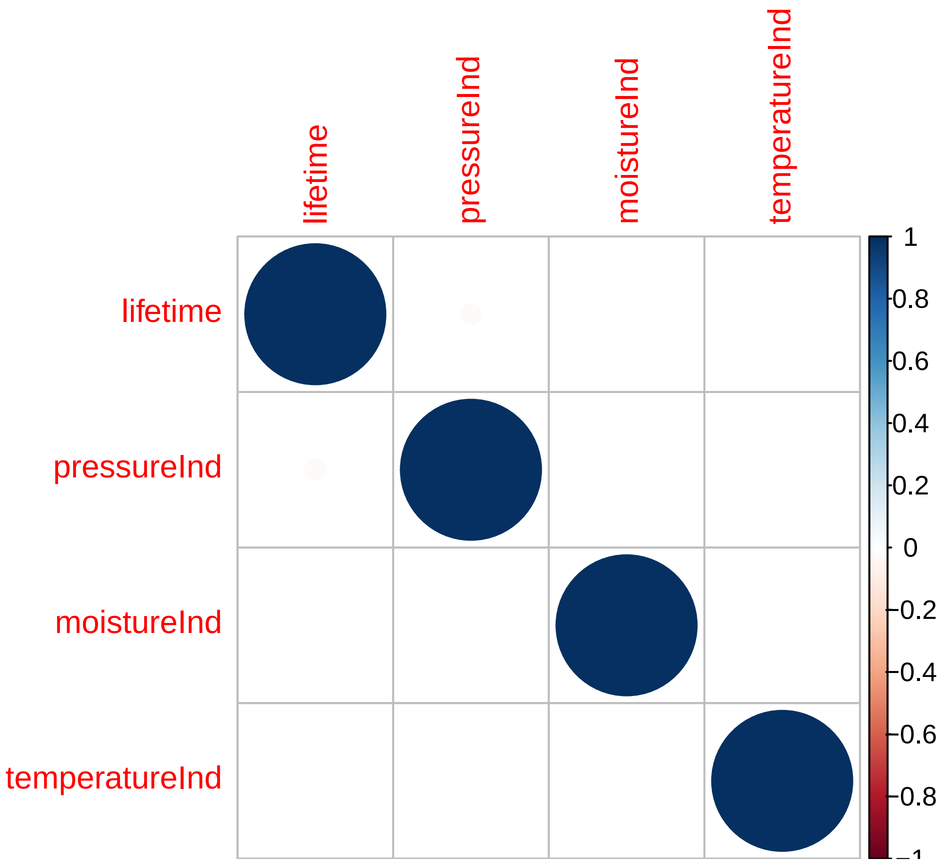
```r
# Using boxplot to get a sense of the medians, quartiles and
# outliers for continuous variables

boxplot(pred_data[, c("lifetime", "pressureInd", "moistureInd",
    "temperatureInd")])
```

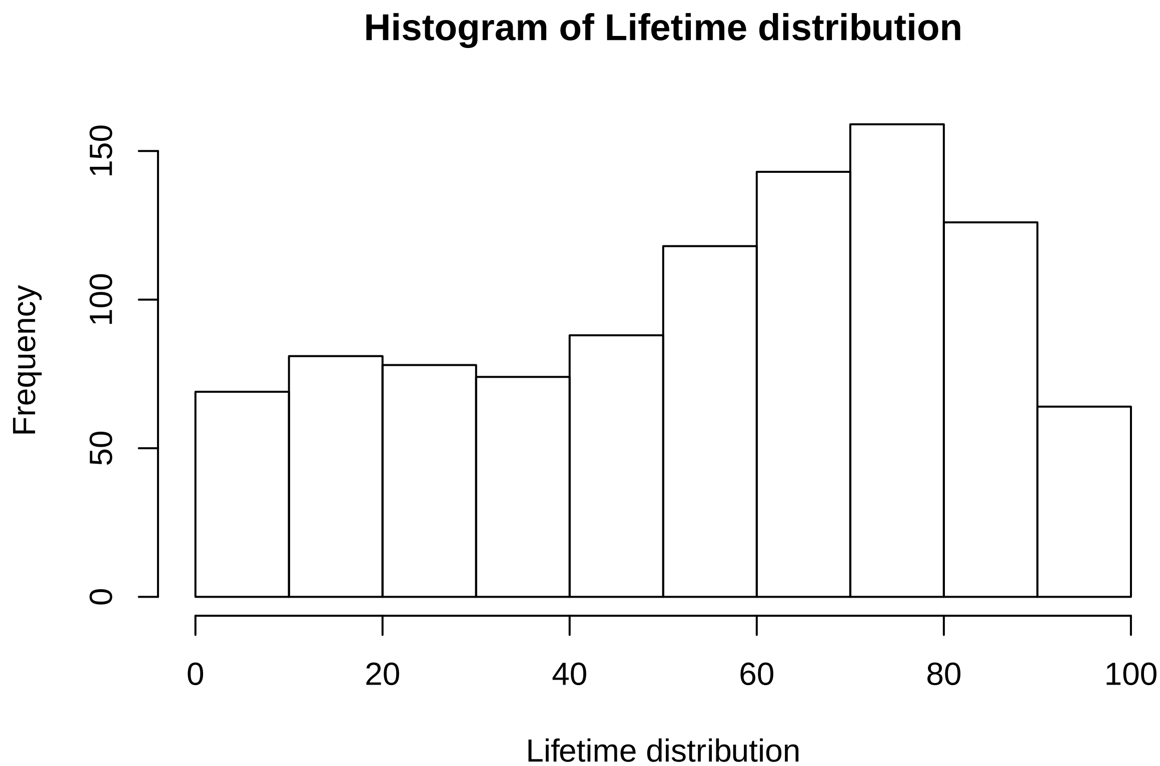



```r
# Plotting correlation matrix
mat <- pred_data[, c("lifetime", "pressureInd", "moistureInd",
```

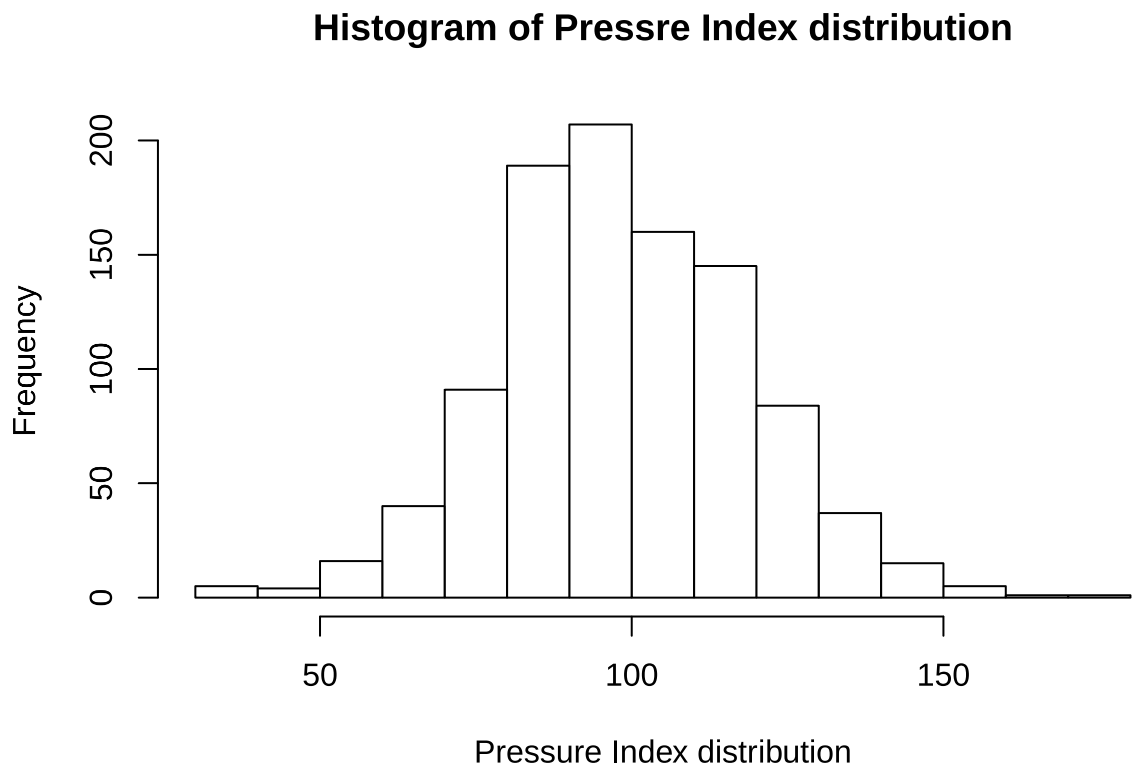
```
    "temperatureInd")]
corr_mat = cor(mat, method = "s")
corrplot(corr_mat)
```



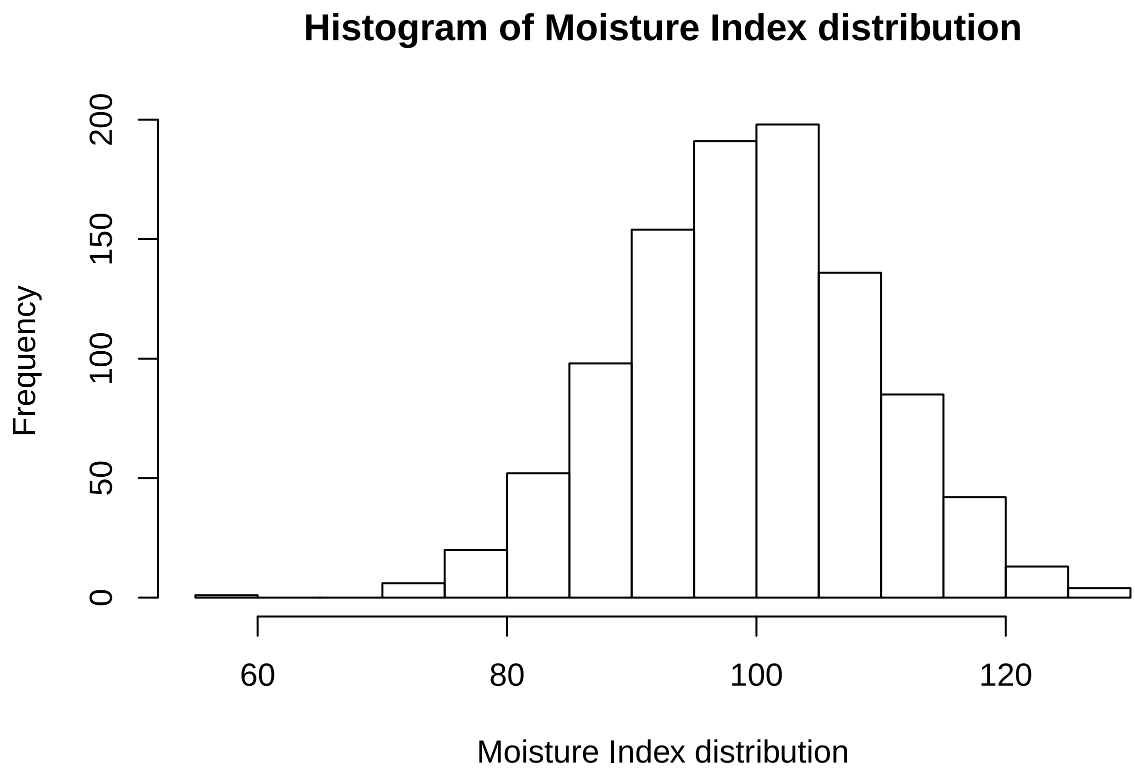

```
# Checking for normality of distribution for all continuous
# variables
hist(pred_data$lifetime, xlab = "Lifetime distribution", main = "Histogram of Lifetime distribution")
```



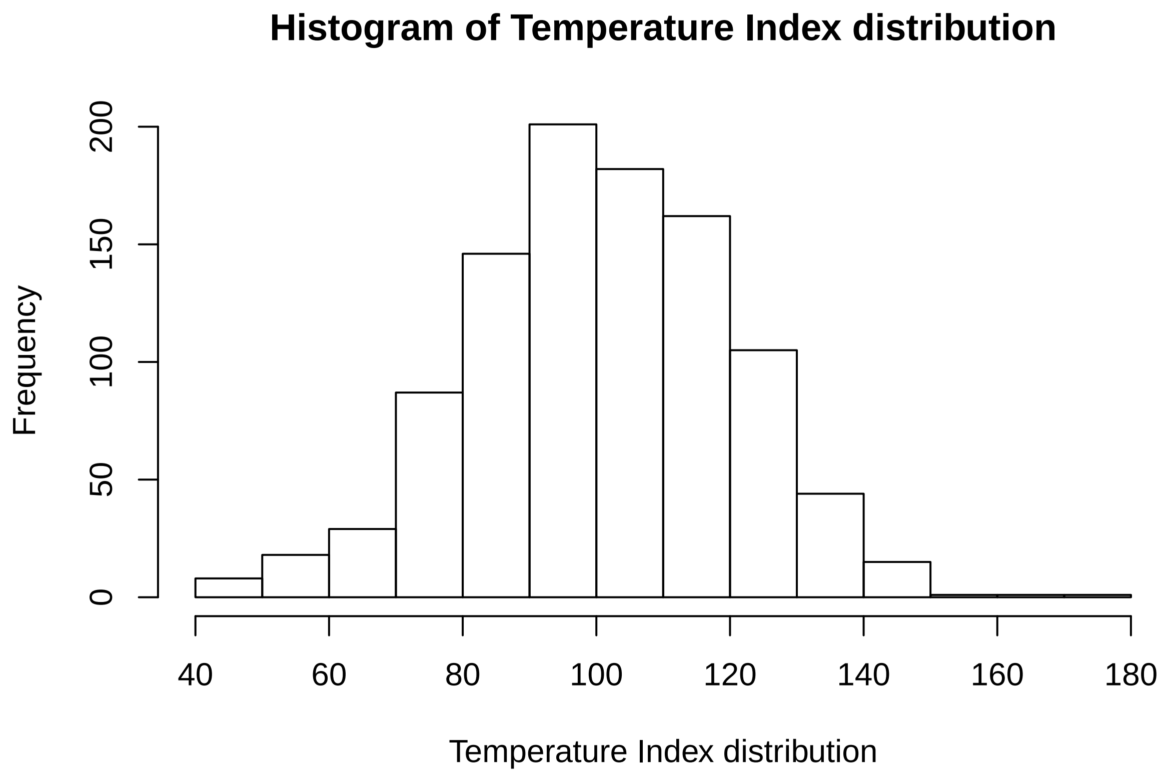

```r
hist(pred_data$pressureInd, xlab = "Pressure Index distribution",
    main = "Histogram of Pressre Index distribution")
```




```
hist(pred_data$moistureInd, xlab = "Moisture Index distribution",
    main = "Histogram of Moisture Index distribution")
```




```r
hist(pred_data$temperatureInd, xlab = "Temperature Index distribution",
     main = "Histogram of Temperature Index distribution")
```



Understanding categorical data

```r
# Converting all categorical variables to factor
pred_data$broken <- as.factor(pred_data$broken)
pred_data$team <- as.factor(pred_data$team)
pred_data$provider <- as.factor(pred_data$provider)

# Looking at all values for the variables
table(pred_data$broken)
```

```
##
##   0   1
## 603 397
```

```r
table(pred_data$team)
```

```
##
## TeamA TeamB TeamC
##   336   356   308
```

```r
table(pred_data$provider)
```

```
##
## Provider1 Provider2 Provider3 Provider4
##       254       266       242       238
```

Checking for a statistical difference between features of machines that broke down that those that didnt

```r
t.test(pred_data[pred_data$broken == 0, ]$lifetime, pred_data[pred_data$broken ==
    1, ]$lifetime)
```

```
##
##  Welch Two Sample t-test
##
## data:  pred_data[pred_data$broken == 0, ]$lifetime and pred_data[pred_data$broken == 1, ]$lifetime
## t = -35.625, df = 915.68, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -40.09221 -35.90551
## sample estimates:
## mean of x mean of y
##  40.10945  78.10831
```

```r
# very small p-value, hence there is a difference between \
# lifetimes of machines that break down and those that don't

t.test(pred_data[pred_data$broken == 0, ]$pressureInd, pred_data[pred_data$broken ==
    1, ]$pressureInd)
```

```
##
##  Welch Two Sample t-test
##
## data:  pred_data[pred_data$broken == 0, ]$pressureInd and pred_data[pred_data$broken == 1, ]$pressur
## t = 0.91364, df = 844.09, p-value = 0.3612
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.355404  3.716115
## sample estimates:
## mean of x mean of y
##  99.06794  97.88758
```

```r
# signifacnt p-value, hence there we cannot reject null
# hypothesis, and hence cannot be sure if there is a
# difference between the pressureInd of machines that break
# down against those that don't

t.test(pred_data[pred_data$broken == 0, ]$moistureInd, pred_data[pred_data$broken ==
    1, ]$moistureInd)
```

```
##
##  Welch Two Sample t-test
##
## data:  pred_data[pred_data$broken == 0, ]$moistureInd and pred_data[pred_data$broken == 1, ]$moistur
## t = 0.61648, df = 845.99, p-value = 0.5377
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.8698651  1.6664955
## sample estimates:
## mean of x mean of y
##  99.53485  99.13654
```

```r
## signifacnt p-value, hence there we cannot reject null
## hypothesis, and hence cannot be sure if there is a
## difference between the moistureInd of machines that break
```

```
## down against those that don't

t.test(pred_data[pred_data$broken == 0, ]$temperatureInd, pred_data[pred_data$broken ==
    1, ]$temperatureInd)
```

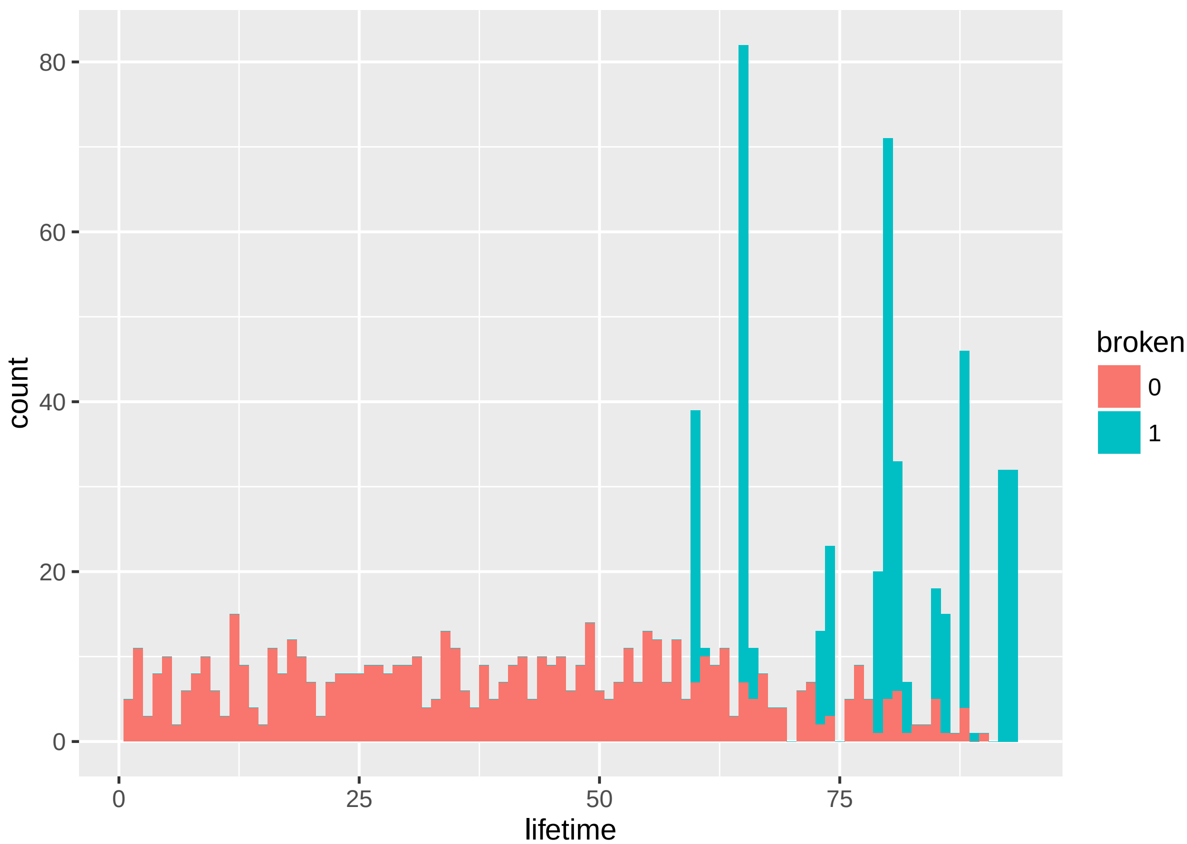
```
##
##  Welch Two Sample t-test
##
## data:  pred_data[pred_data$broken == 0, ]$temperatureInd and pred_data[pred_data$broken == 1, ]$tempe
## t = -0.48401, df = 839.1, p-value = 0.6285
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.114977  1.882610
## sample estimates:
## mean of x mean of y
##  100.3839  101.0001
```

```
# signifacnt p-value, hence there we cannot reject null
# hypothesis, and hence cannot be sure if there is a
# difference between the temperatureInd of machines that
# break down against those that don't
```

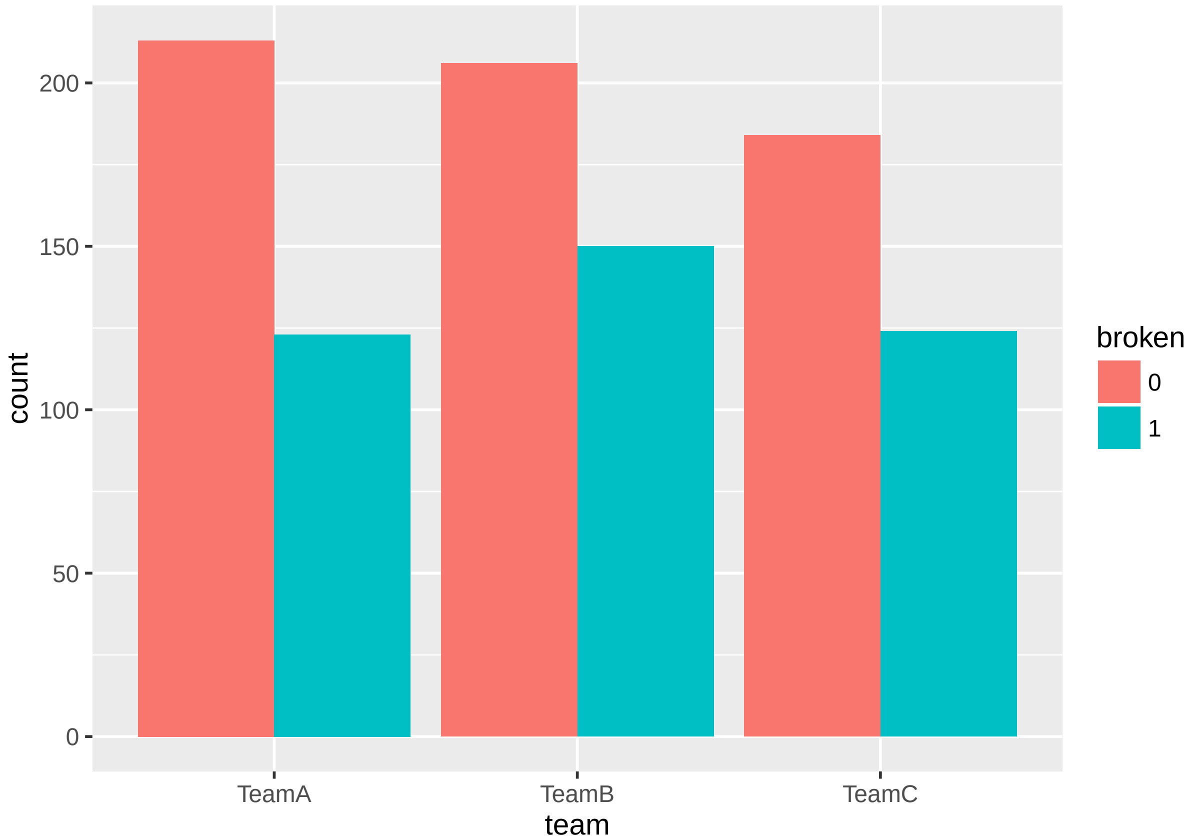
This makes intuitive sense, as older the machine, more likely it is to break. I thought that the operating conditions would show some difference in machines that breakdown and those that don't, but it does not look like there is any difference.

Generating ggplots for looking at distribution of variables

```
ggplot(pred_data) + geom_histogram(aes(x = lifetime, fill = broken),
    stat = "bin", binwidth = 1)
```

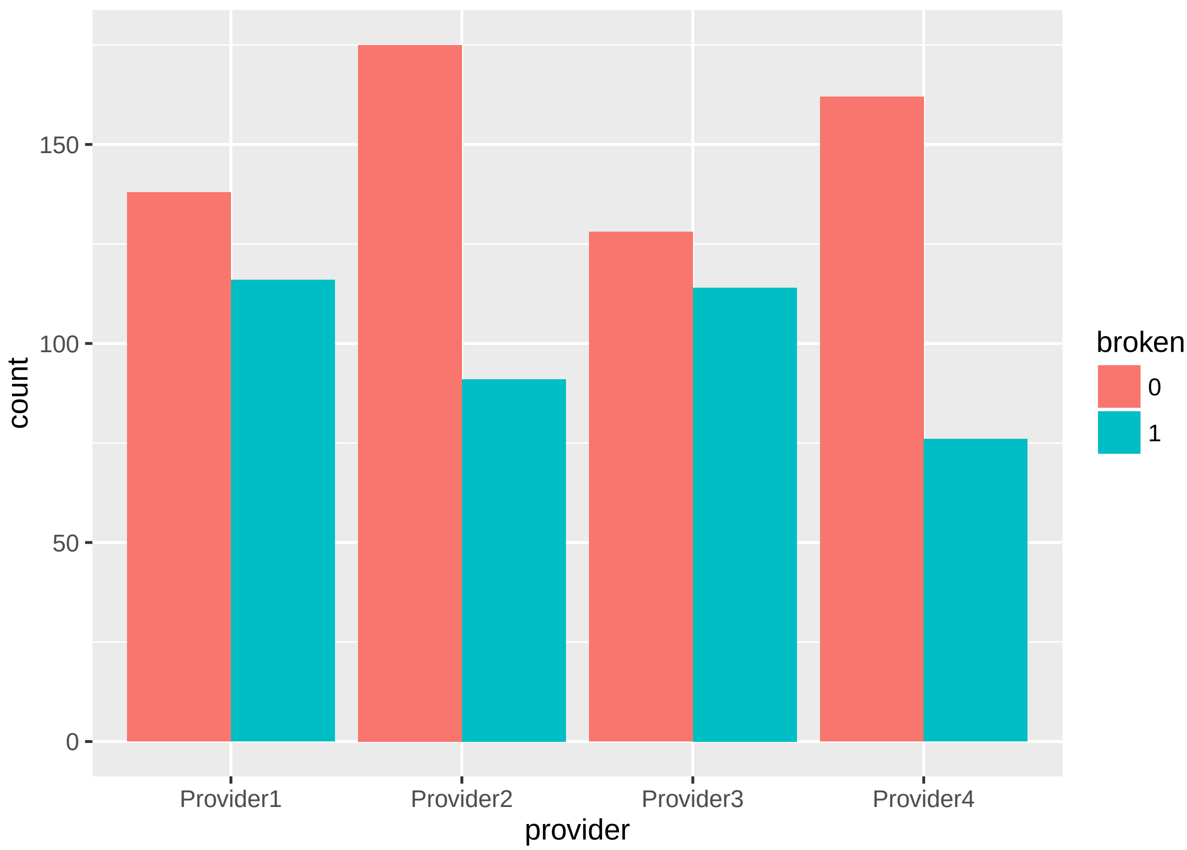



```r
ggplot(pred_data) + geom_bar(aes(x = team, fill = broken), position = "dodge")
```




```
# Checking the distribution of machines break downs across
# teams It does look like TeamB causes more breakdowns than
# the rest TeamA performs the best with the least number of
# breakdowns

ggplot(pred_data) + geom_bar(aes(x = provider, fill = broken),
    position = "dodge")
```




```
# Checking to see if any of the providers of the machine
# stands out for breakdowns.  Provider1 has the most
# breakdown machines, but it does not look to be an outlier.
# Provider2 does very well with the least number of
# breakdowns.
```

2. Modelling

Till now we looked at the data in various ways to see if anything stood out in terms of what was causing breakdowns of the machines. But nothing stood out, so we don't have any particular pattern which we can use to say with confidence that a machine will break down. Hence, we turn to machine learning models. Here, I have split the data into training and testing datasets and build classification models using the following algorithms: i. Logistic regression ii. Classfication Tree iii. Support Vector Machines iv. Naive Bayes

I will compare the performance of all these models and use the one which gives the best accuracy for the model. We can also use other criteria like Precision or Recall to select model depending on our usecase. In this case, we want to predict with higher certainity before a machine breaks down. So it is imporatant to flag a machine that is likely to break, i.e., to reduce the false negatives in our model. So we should also select a model that minimizes Recall.

Train test split

```
set.seed(84)

# Train 75% of the data and test on 25%
sample = sample.split(pred_data, SplitRatio = 0.75)
train = subset(pred_data, sample == TRUE)
test = subset(pred_data, sample == FALSE)
```

i. Logistic regression

```r
log_fit <- glm(broken ~ ., family = binomial, data = train)
summary(log_fit)
```

```
##
## Call:
## glm(formula = broken ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.367   0.000   0.000   0.000   1.135
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -4.294e+03  1.087e+05  -0.039    0.968
## lifetime             5.441e+01  1.035e+03   0.053    0.958
## pressureInd          1.131e-01  9.259e-02   1.222    0.222
## moistureInd         -5.201e-02  7.757e-02  -0.671    0.503
## temperatureInd       1.325e-01  8.594e-02   1.541    0.123
## teamTeamB           -7.610e+00  7.164e+04   0.000    1.000
## teamTeamC            3.371e+02  7.195e+04   0.005    0.996
## providerProvider2 -6.857e+02  1.300e+04  -0.053    0.958
## providerProvider3  7.679e+02  1.484e+04   0.052    0.959
## providerProvider4 -4.644e+02  8.933e+03  -0.052    0.959
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 952.736  on 713  degrees of freedom
## Residual deviance:  13.117  on 704  degrees of freedom
## AIC: 33.117
##
## Number of Fisher Scoring iterations: 25
```

```r
log_pred <- predict(log_fit, test, type = c("response"))
log_pred <- factor(ifelse(log_pred > 0.5, "1", "0"))
cm_log <- confusionMatrix(log_pred, test$broken, mode = "prec_recall")
cm_log
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 162   1
##          1   3 120
##
##                Accuracy : 0.986
##                  95% CI : (0.9646, 0.9962)
##     No Information Rate : 0.5769
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9714
##  Mcnemar's Test P-Value : 0.6171
##
##               Precision : 0.9939
```

```
##                  Recall : 0.9818
##                      F1 : 0.9878
##              Prevalence : 0.5769
##          Detection Rate : 0.5664
##    Detection Prevalence : 0.5699
##       Balanced Accuracy : 0.9868
##
##        'Positive' Class : 0
##
```

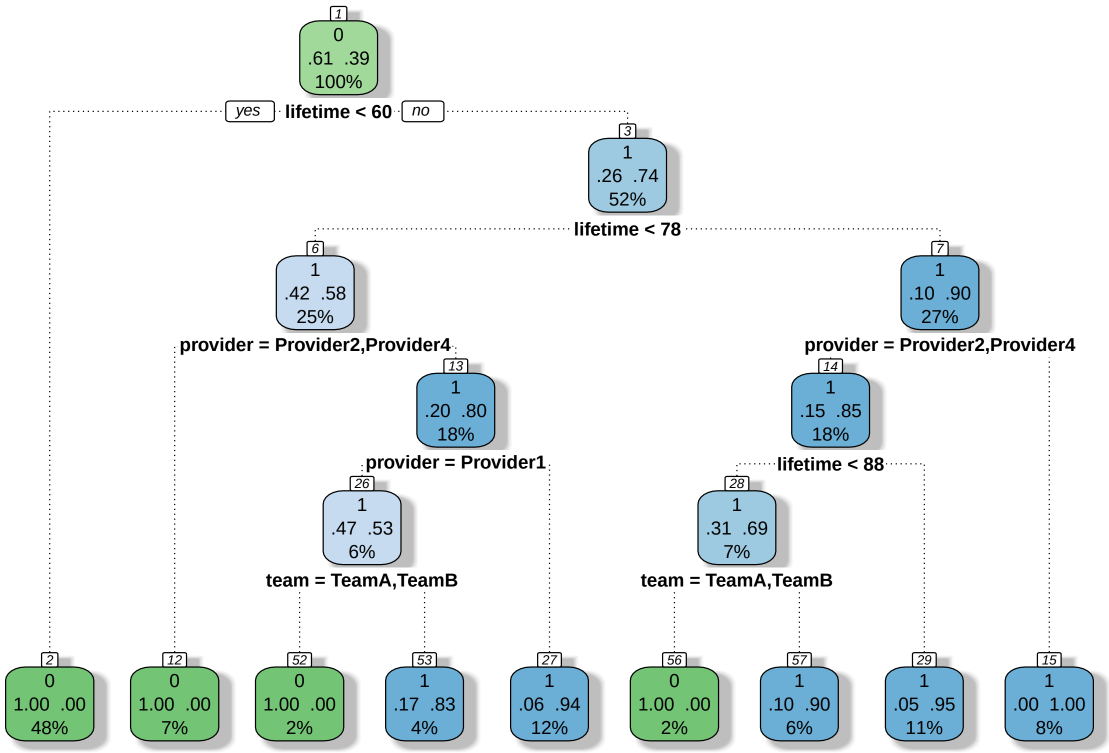
ii. Classification Tree

```
tree_fit <- rpart(broken ~ ., method = "class", data = train)
tree_pred <- predict(tree_fit, newdata = test, type = c("class"))


cm_tree <- confusionMatrix(tree_pred, test$broken, mode = "prec_recall")
cm_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 155   0
##          1  10 121
##
##                Accuracy : 0.965
##                  95% CI : (0.9366, 0.9831)
##     No Information Rate : 0.5769
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9292
##  Mcnemar's Test P-Value : 0.004427
##
##               Precision : 1.0000
##                  Recall : 0.9394
##                      F1 : 0.9688
##              Prevalence : 0.5769
##          Detection Rate : 0.5420
##    Detection Prevalence : 0.5420
##       Balanced Accuracy : 0.9697
##
##        'Positive' Class : 0
##
```

```
fancyRpartPlot(tree_fit)
```



Rattle 2018−Mar−31 23:20:37 Monica Jain

iii. SVM

```
svm_fit <- svm(broken ~ ., data = train)
svm_pred <- predict(svm_fit, newdata = test, type = c("class"))
cm_svm <- confusionMatrix(svm_pred, test$broken, mode = "prec_recall")
cm_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 143   0
##          1  22 121
##
##                Accuracy : 0.9231
##                  95% CI : (0.8859, 0.9512)
##     No Information Rate : 0.5769
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8462
##  Mcnemar's Test P-Value : 7.562e-06
##
##               Precision : 1.0000
##                  Recall : 0.8667
##                      F1 : 0.9286
##              Prevalence : 0.5769
##          Detection Rate : 0.5000
```

```
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.9333
##
##        'Positive' Class : 0
##
```

iv. Naive Bayes

```r
nb_fit <- naiveBayes(broken ~ ., data = train)
nb_pred <- predict(nb_fit, newdata = test, type = c("class"))
cm_nb <- confusionMatrix(nb_pred, test$broken, mode = "prec_recall")
cm_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 141   9
##          1  24 112
##
##                Accuracy : 0.8846
##                  95% CI : (0.8418, 0.9192)
##     No Information Rate : 0.5769
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7675
##  Mcnemar's Test P-Value : 0.01481
##
##               Precision : 0.9400
##                  Recall : 0.8545
##                      F1 : 0.8952
##              Prevalence : 0.5769
##          Detection Rate : 0.4930
##    Detection Prevalence : 0.5245
##       Balanced Accuracy : 0.8901
##
##        'Positive' Class : 0
##
```

```r
a <- data.frame(c("Logistic Regression", "Classifciation Tree",
    "Support Vector Machine", "Naive Bayes"))

colnames(a) <- "Model"

a$Accuracy <- c(cm_log$overall[1] * 100, (cm_tree$overall[1] *
    100), (cm_svm$overall[1] * 100), (cm_nb$overall[1] * 100))

a
```

```
##                    Model Accuracy
## 1    Logistic Regression 98.60140
## 2    Classifciation Tree 96.50350
## 3 Support Vector Machine 92.30769
## 4            Naive Bayes 88.46154
```

We see that out of all the models trained, Logistic regression performs the best, with an accuracy of 98.6, followed by Classification tree giving an accuracy of 96.5%.

This makes sense as this data is not to complex, with only 6 predictor variables and 1000 rows. Also, decision tree makes a more complex model, increasing the chances of overfitting, making logistic regression the best model for this usecase.