HarvardX Data Science Capstone Project

Vincent

May 2019

1 Introduction

This project is based on the "Rain in Australia" datasaset available on kaggle: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

The goal is to predict wether or not it will rain tomorrow by training a binary classification model.

2 Preparing data

2.1 Downloading and loading data

The base datafile is downloaded from Kaggle if not already available.

```
# Download base dataset
DataURL <- "https://www.kaggle.com/jsphyg/weather-dataset-rattle-package/downloads/
weather-dataset-rattle-package.zip/2"
DataFile <- "weather-dataset-rattle-package.zip"
if (!file.exists(DataFile)) {
  download.file(DataURL, destfile = DataFile)
  unzip(DataFile)
}
```

```
rain <- read.csv("weatherAUS.csv")
str(rain)
```

```
## 'data.frame':    142193 obs. of  24 variables:
##  $ Date         : Factor w/ 3436 levels "2007-11-01","2007-11-02",..: 397 398 399 400 401 402 403 404
##  $ Location     : Factor w/ 49 levels "Adelaide","Albany",..: 3 3 3 3 3 3 3 3 3 3 ...
```

```
##  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
##  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
##  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
##  $ Evaporation  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Sunshine     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ WindGustDir  : Factor w/ 16 levels "E","ENE","ESE",..: 14 15 16 5 14 15 14 14 7 14 ...
##  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindDir9am   : Factor w/ 16 levels "E","ENE","ESE",..: 14 7 14 10 2 14 13 11 10 9 ...
##  $ WindDir3pm   : Factor w/ 16 levels "E","ENE","ESE",..: 15 16 16 1 8 14 14 14 8 11 ...
##  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
##  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
##  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
##  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
##  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
##  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
##  $ RainToday    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 2 ...
##  $ RISK_MM      : num  0 0 0 1 0.2 0 0 0 1.4 0 ...
##  $ RainTomorrow : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 2 1 ...
```

2.2 Understanding and cleaning data

2.2.1 Available columns

column	description
Date	The date of observation
Location	The common name of the location of the weather station
MinTemp	The minimum temperature in degrees celsius
MaxTemp	The maximum temperature in degrees celsius
Rainfall	The amount of rainfall recorded for the day in mm
Evaporation	The so-called Class A pan evaporation (mm) in the 24 hours to 9am
Sunshine	The number of hours of bright sunshine in the day
WindGustDir	The direction of the strongest wind gust in the 24 hours to midnight
WindGustSpeed	The speed (km/h) of the strongest wind gust in the 24 hours to midnight
WindDir9am	Direction of the wind at 9am
WindDir3pm	Direction of the wind at 3pm
WindSpeed9am	Wind speed (km/hr) averaged over 10 minutes prior to 9am
WindSpeed3pm	Wind speed (km/hr) averaged over 10 minutes prior to 3pm
Humidity9am	Humidity (percent) at 9am
Humidity3pm	Humidity (percent) at 3pm
Pressure9am	Atmospheric pressure (hpa) reduced to mean sea level at 9am
Pressure3pm	Atmospheric pressure (hpa) reduced to mean sea level at 3pm
Cloud9am	Fraction of sky obscured by cloud at 9am. This is measured in “oktas”.
Cloud3pm	Fraction of sky obscured by cloud (in “oktas”: eighths) at 3pm.
Temp9am	Temperature (degrees C) at 9am
Temp3pm	Temperature (degrees C) at 3pm
RainToday	Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
RISK_MM	The amount of next day rain in mm. Used to create response variable RainTomorrow.
RainTomorrow	The target variable. Did it rain tomorrow?

The outcome variable, RainTomorrow, is a factor variable with two levels. This is a binary classification problem

2.2.2 Date

Date is stored as a factor variable and can be converted to an actual date.

```
rain$Date <- as.Date(rain$Date)
summary(rain$Date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.
## "2007-11-01" "2011-01-06" "2013-05-27" "2013-04-01" "2015-06-12"
##         Max.
## "2017-06-25"
```

However extracting the month CAN give can be a better predictor because it will have the same value for the same season over the years.

```
library(lubridate)
rain$Month <- month(ymd(rain$Date))
```

2.2.3 RISK_MM and RainTomorrow

This predictor is directly linked to the outcome : "The amount of next day rain in mm. Used to create response variable RainTomorrow.". And in fact there is a perfect corelation between those variables. RainTomorrow is True if RISK_MM > 1.

```
table(RainTomorrow = rain$RainTomorrow, RISK_MM_above_1 = rain$RISK_MM > 1)
```

```
##             RISK_MM_above_1
## RainTomorrow  FALSE   TRUE
##          No  110316      0
##          Yes      0  31877
```

This predictor must be removed because otherwise it would leak the outcome variable in the dataset.

```
rain$RISK_MM <- NULL
```

2.2.4 Rainfall and RainToday

RainToday is also a binary representation of the numeric variable Rainfall. Raintoday is true when rainfall $>$ 1mm.

```
table(RainToday = rain$RainToday, rainfall_above_1 = rain$Rainfall > 1)
```

```
##          rainfall_above_1
## RainToday  FALSE   TRUE
##       No  109332      0
##       Yes      0  31455
```

Since those variables are prefectly correlated, we will keep only one of them. Rainfall contains more variability so may have a better predictive value.

```
rain$RainToday <- NULL
```

2.2.5 Missing values

Some variables contain a high percentage of missing values.

```
sapply(rain, function(x) round(sum(is.na(x)) / length(x) * 100))
```

```
##          Date      Location       MinTemp       MaxTemp      Rainfall
##             0             0             0             0             1
##   Evaporation      Sunshine   WindGustDir WindGustSpeed    WindDir9am
##            43            48             7             7             7
##    WindDir3pm  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm
##             3             1             2             1             3
##   Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am
##            10            10            38            40             1
##       Temp3pm  RainTomorrow         Month
##             2             0             0
```

for some of them, this percentage is so high that trying to impute those values is useless, therefore we keep only those with up to 10% of missing values and eliminate the others: Evaporation, Sunshine, Cloud9am, Cloud3pm

```
rain$Evaporation <- NULL
rain$Sunshine <- NULL
rain$Cloud9am <- NULL
rain$Cloud3pm <- NULL
```

For the remaining data, we could replace NA with the mean for numerical variable or with the mode for factor variables, but since we are dealing with weather data and we have a date, it's better to assign the last non missing values for the same location.

```
library(dplyr)
library(tidyr)
rain <- rain %>%
  arrange(Date) %>%
  group_by(Location) %>%
  fill(WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Rainfall, Pressure9am,
       Pressure3pm, MinTemp, MaxTemp, Temp9am, Temp3pm, WindGustDir, WindDir9am, WindDir3pm) %>%
  fill(WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Rainfall, Pressure9am,
       Pressure3pm, MinTemp, MaxTemp, Temp9am, Temp3pm, WindGustDir, WindDir9am, WindDir3pm, .direction
rain <- rain %>% ungroup()
rain$Date <- NULL
sapply(rain, function(x) sum(is.na(x)))
```

```
##      Location       MinTemp       MaxTemp      Rainfall   WindGustDir
##             0             0             0             0          5971
## WindGustSpeed    WindDir9am    WindDir3pm  WindSpeed9am  WindSpeed3pm
##          5971             0             0             0             0
##   Humidity9am   Humidity3pm   Pressure9am   Pressure3pm       Temp9am
##             0             0         11781         11781             0
##       Temp3pm  RainTomorrow         Month
##             0             0             0
```

After this imputation, some values are still missing because certain variables where not collected in some locations, and when all values are missing, we cannot impute the previous value. We can deal with those special cases. For WindGustDir we impute the value of WindDir9am for the same observation For WindGustSpeed, we impute the maximum of WindSpeed9am and WindSpeed3pm for the same observation For Pressure9am and Pressure3pm we have no value in the same observation thta would make sense, so we impute the mean of these variables in the entire dataset.

```
rain$WindGustDir[is.na(rain$WindGustDir)] <- rain$WindDir9am[is.na(rain$WindGustDir)]
rain$WindGustSpeed[is.na(rain$WindGustSpeed)] <- max(rain$WindSpeed9am[is.na(rain$WindGustSpeed)],
```

```
                                                    rain$WindSpeed3pm[is.na(rain$WindGustSpeed)])
rain$Pressure9am[is.na(rain$Pressure9am)] <- mean(rain$Pressure9am[!is.na(rain$Pressure9am)])
rain$Pressure3pm[is.na(rain$Pressure3pm)] <- mean(rain$Pressure3pm[!is.na(rain$Pressure3pm)])
```

At this stage, we no longer have missing values in the dataset, and we still have 142193 observations. If we had simply ignored all observations with missing values, we would only have 56420 left. This simple imputation allows us to consider almost three times more observations and thus keep more signal in the dataset, hopefully leading to more insight.

3 Exploratory Data Analysis

3.1 Prevalence

Let's check the prevalence of the outcome variable

```
prop.table(table(rain$RainTomorrow))
```

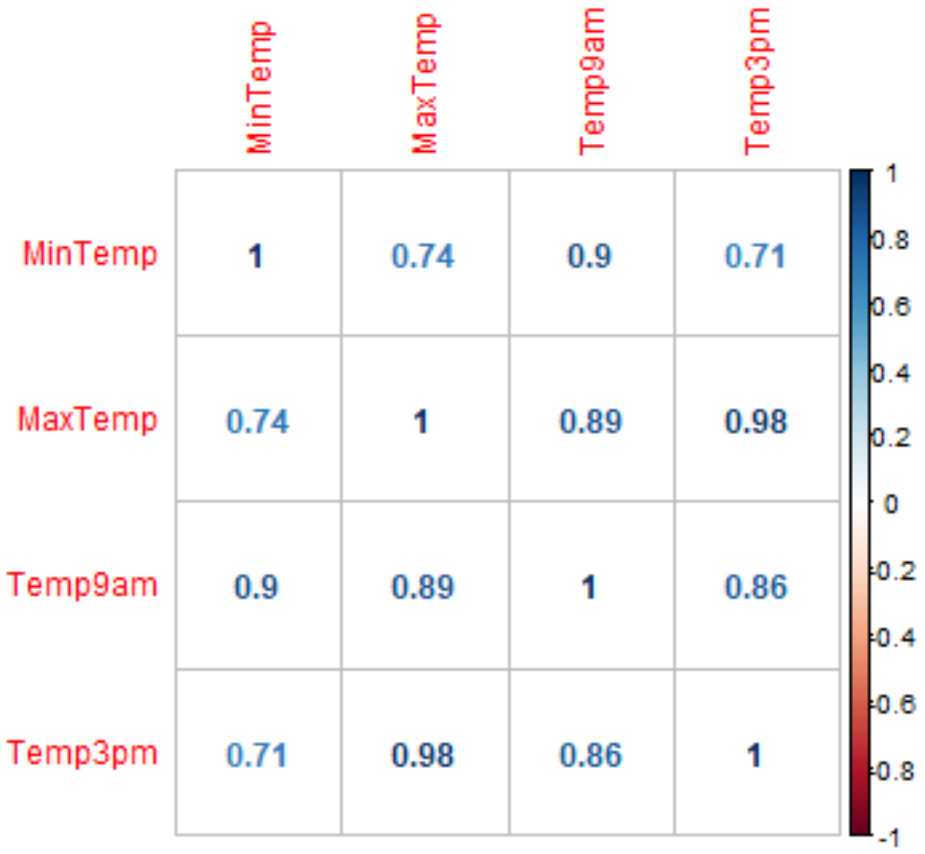
```
##
##        No       Yes
## 0.7758188 0.2241812
```

We have to take prevalence into account because in 77.5% of cases, the outcome is NO (no rain). So a dummy model always predicting no would be close to 77.5% accurate. We have also to consider sensitivity and specificity. Sensitivity is the ability to predict a positive outcome when the actual outcome is positive. Specificity is the ability not to predict a positive outcome when the actual outcome is not positive.

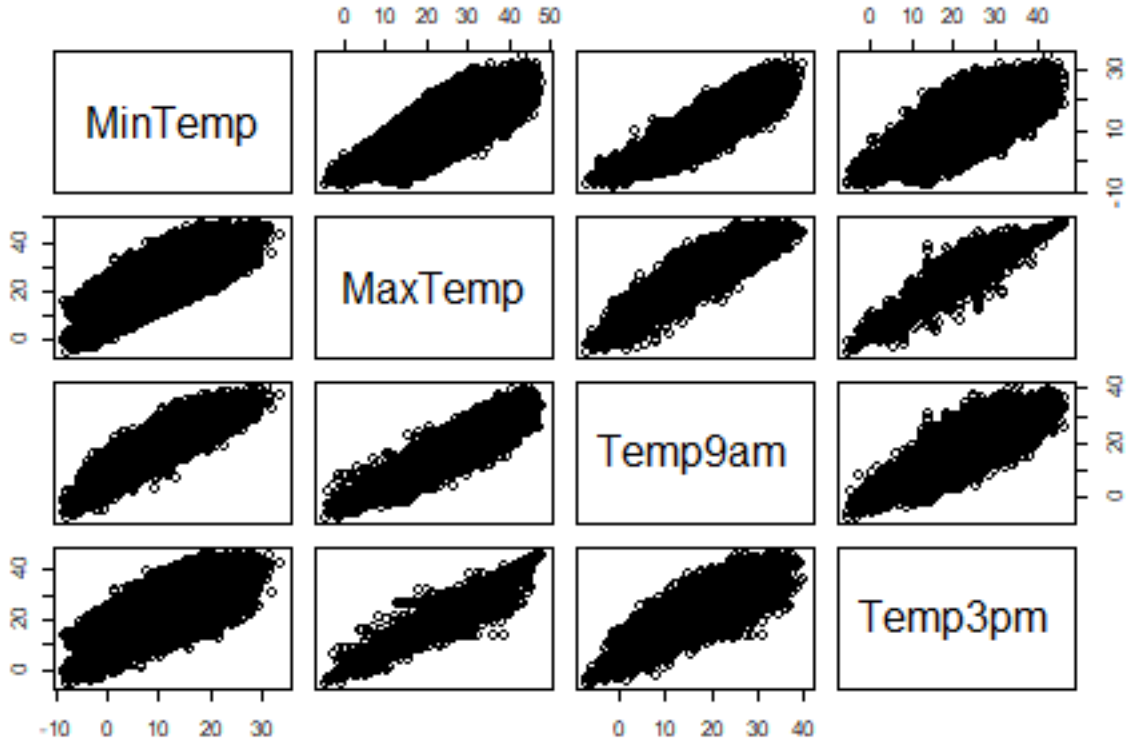
3.2 Correlation of explanatory variables

The explanotory variables contains different measures of the same quantity. For example we have 4 measures of temperature: min, max, at 9am and at 6pm. As can be expected, there is a relatively strong correlation between these measures.

```
library(dplyr)
library(corrplot)
corrplot(cor(select(rain, contains("Temp"))), method = "number")
```

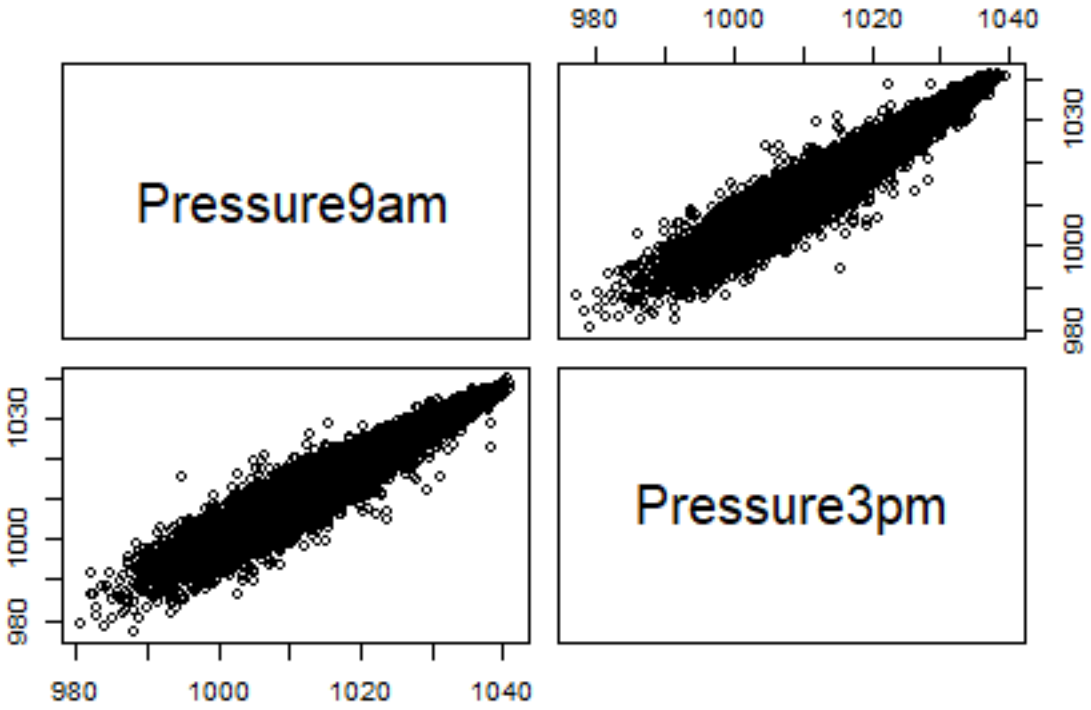



```
pairs(select(rain, contains("Temp")))
```



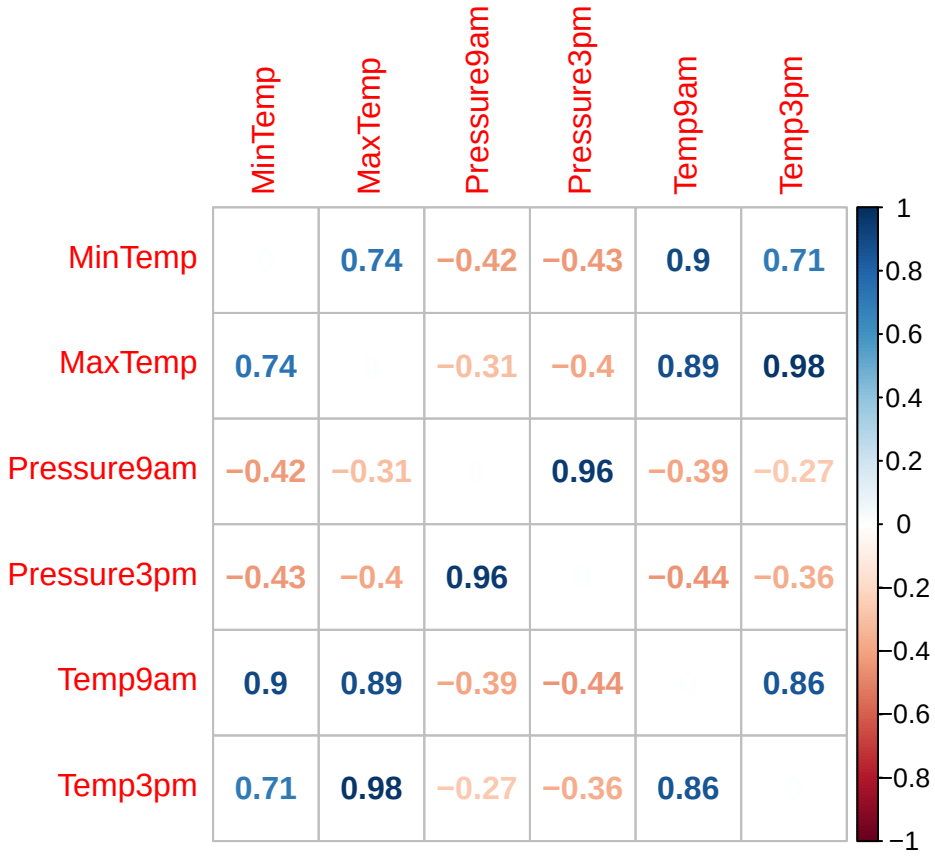
This is also true for the two measures of pressure.

```
pairs(select(rain, contains("Pressure")))
```



We can outline the highest correlations among the available predictors.

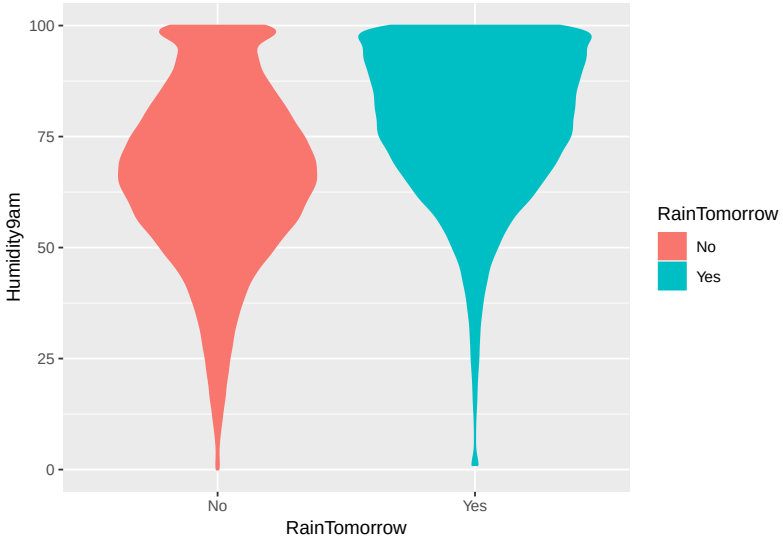
```
numeric <- sapply(rain, is.numeric)
correlations <- cor(rain[,numeric])
diag(correlations) <- 0
high <- apply(abs(correlations) >= 0.7, 2, any)
corrplot(correlations[high, high], method = "number")
```



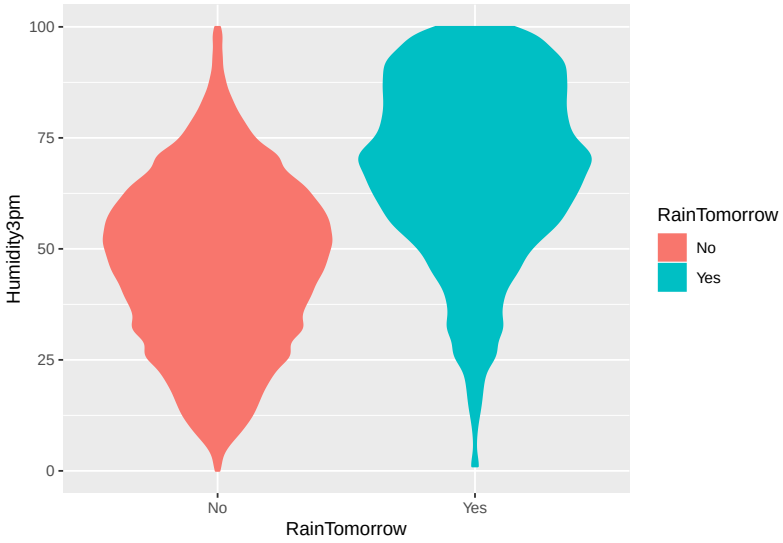
Some predictors are indeed highly correlated. For instance, the most obvious corelations are between Temp9am and MinTemp, Temp 3pm and MaxTemp. Pressure9am and Pressure3pm also seem to bring the same information whith a near perfect corelation of 0.96. Later, in models that are sensitive to corelated predictors, we could choose to keep only MinTemp and MaxTemp and only one measure of Pressure. We would keep most of the signal with 3 predictors instead of 6.

3.3 Humidity

```
library(ggplot2)
rain %>%
  ggplot(aes(x=RainTomorrow, y=Humidity9am, colour = RainTomorrow, fill= RainTomorrow)) + geom_violin()
```




```
rain %>%
  ggplot(aes(x=RainTomorrow, y=Humidity3pm, colour = RainTomorrow, fill= RainTomorrow)) + geom_violin()
```



We notice that, even if in both cases we are measuring Humidity, there difference is greater in the Humidity3pm measure. It could be interesting to keep only Humidity3pm and eliminate Humidity9am with has a substantial corelation of `round(cor(rain$Humidity9am, rain$Humidity3pm), digits = 2`.

4 Models

4.1 Training and testing set

```
library(caret)
set.seed(1971)
test_index <- createDataPartition(rain$RainTomorrow, times = 1, p = 0.2, list = FALSE)
test_set <- rain[test_index, ]
train_set <- rain[-test_index, ]
```

4.2 Baseline model

A simple and naïve model would be to always predict the most frequent outcome. It allows us to have a baseline to evaluate more elaborate models.

```
pred_naive <- rep("No", nrow(test_set))
```

4.2.1 Overall Accuracy

```
mean(pred_naive == test_set$RainTomorrow)
```

```
## [1] 0.7758087
```

As expected, the overall accuracy is close to the prevalence of the most common outcome. However, if we compute the accuracy by outcome, the weakness of our simple approach is revealed. Specificity is perfect but there is no sensitivity.

```
library(dplyr)
test_set %>%
  mutate(y_hat = pred_naive) %>%
  group_by(RainTomorrow) %>%
  summarize(accuracy = mean(y_hat == RainTomorrow))
```

```
## # A tibble: 2 x 2
##   RainTomorrow accuracy
##   <fct>           <dbl>
## 1 No                  1
## 2 Yes                 0
```

4.2.2 Confusion Matrix

```
pred_naive <- as.factor(pred_naive)
levels(pred_naive) <- levels(test_set$RainTomorrow)
table(predicted = pred_naive, actual = test_set$RainTomorrow)
```

```
##          actual
## predicted    No   Yes
##       No  22064  6376
##       Yes     0     0
```

```
cm <- confusionMatrix(data = pred_naive, reference = test_set$RainTomorrow, positive = "Yes")
all_results <- cbind(data.frame(Model = 'Fixed No Model'), as.data.frame(t(c(cm$byClass[c(1,2,7)], cm$o
all_results
```

```
##            Model Sensitivity Specificity F1  Accuracy
## 1 Fixed No Model           0           1 NA 0.7758087
```

Of course our naive approach leads to a perfect specificity but without any sensitivity.

Since computing all these models can be computationally intensive, we first enable parallel computing to speed-up model building.

4.3 Enable parallel computation

```
library(parallel)
library(doParallel)
```

```
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
```

4.4 Logistic Regression

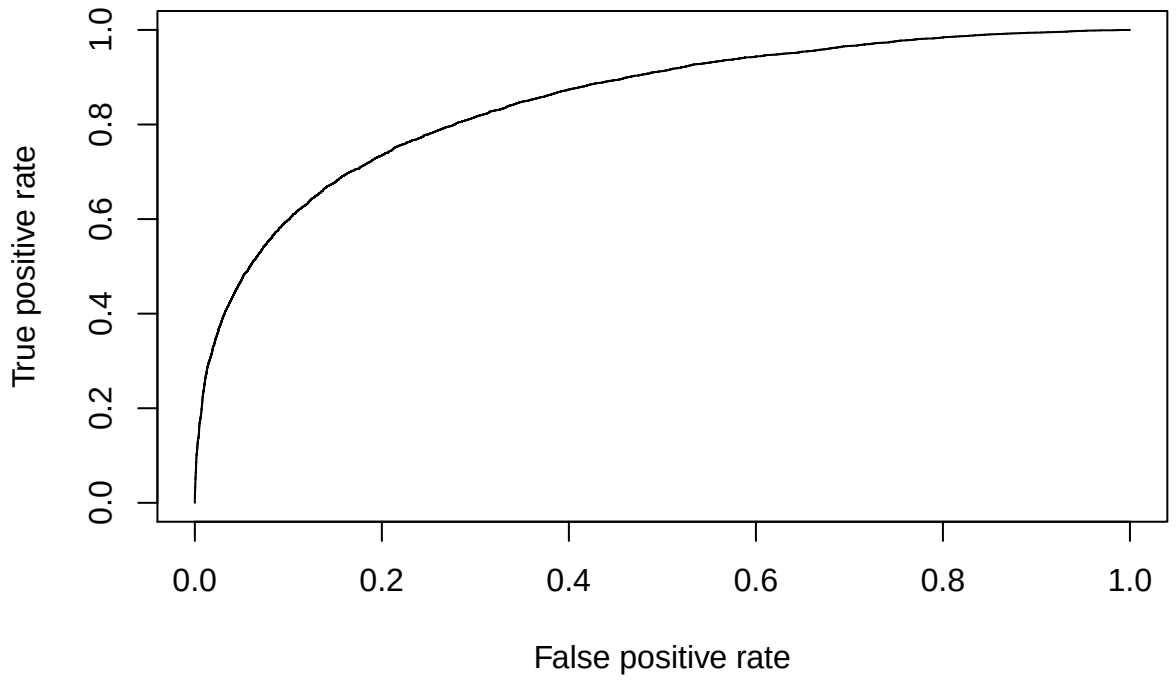
To build a logistic regression model, we use knowledge we gained during explanatory analysis and eliminate highly correlated predictors.

```
mod_glm <- glm(RainTomorrow ~ WindGustSpeed + WindSpeed9am + Humidity3pm + Pressure3pm + MinTemp + MaxT
#summary(mod_glm)
```

All of the selected predictors are highly significant.

```
pred_glm <- predict(mod_glm, type = 'response', newdata = test_set)
```

```
library(ROCR)
pr <- prediction(pred_glm, test_set$RainTomorrow)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8479651
```

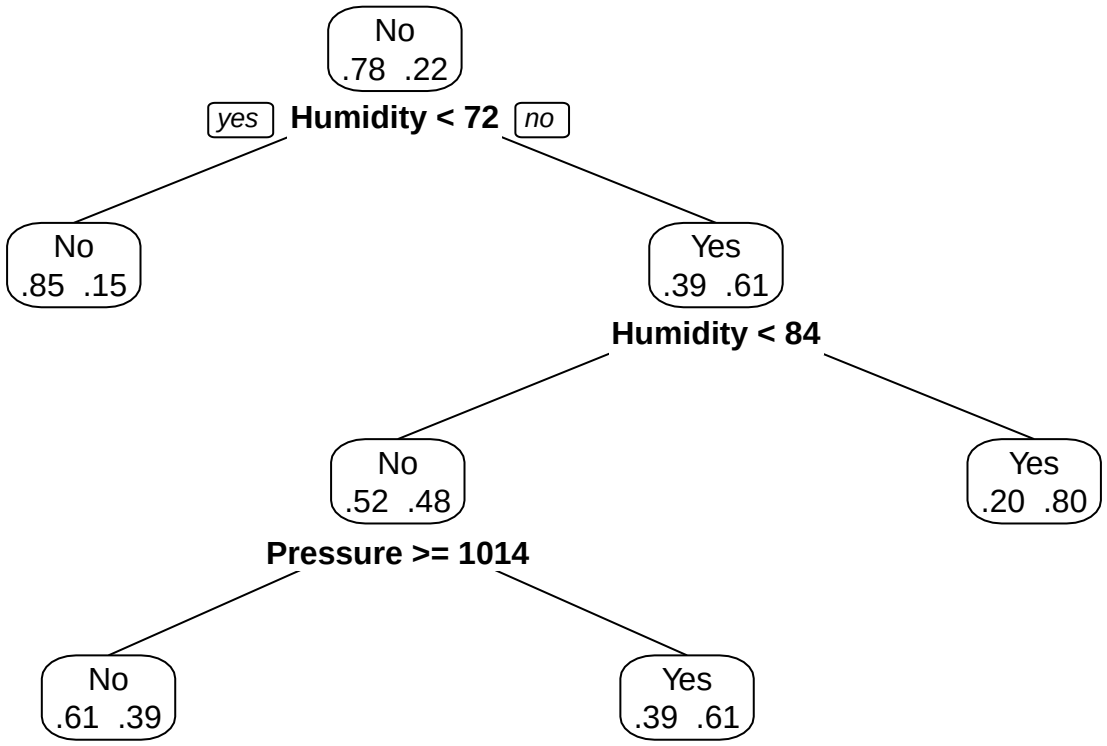
```
pred_glm <- predict(mod_glm, type = 'response', newdata = test_set)
pred_glm2 <- as.factor(ifelse (pred_glm < 0.3, "No", "Yes"))
```

```
cm <- confusionMatrix(data = pred_glm2, reference = test_set$RainTomorrow, positive = "Yes")
all_results <- rbind(all_results, cbind(data.frame(Model = 'Logistic Regression Model'), as.data.frame(t
all_results[2,]
```

```
##                         Model Sensitivity Specificity        F1  Accuracy
## 2 Logistic Regression Model   0.6755019   0.8518854 0.6174468 0.8123418
```

4.5 Tree

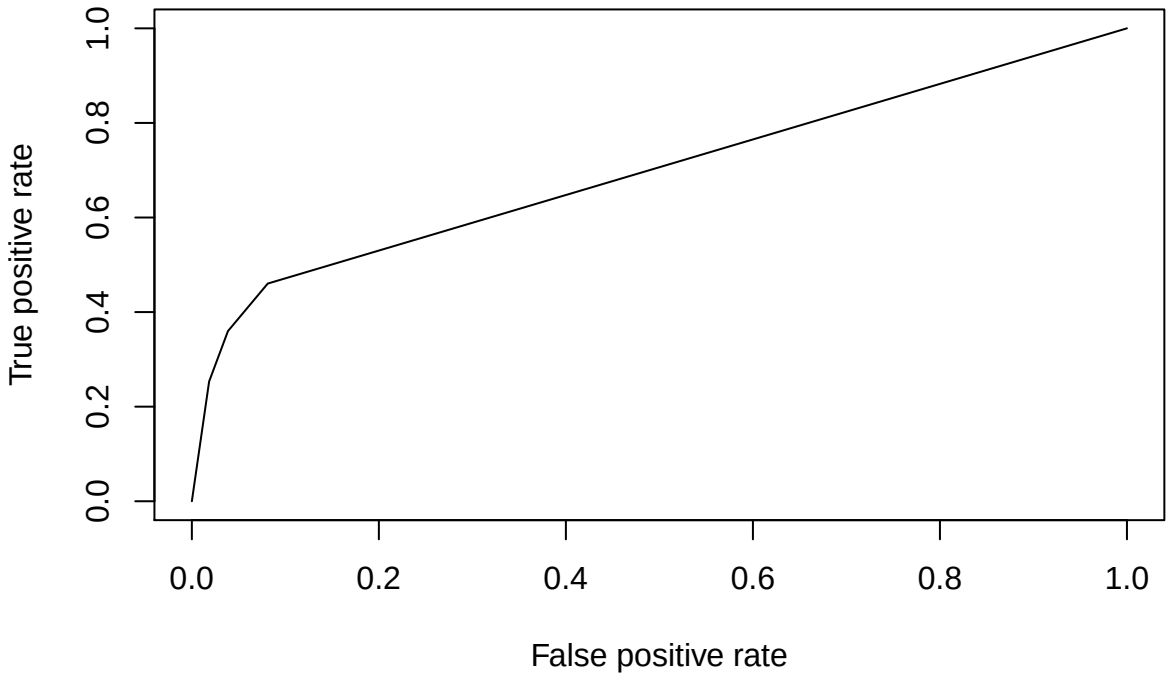
```
library(rpart)
mod_tree <- rpart(RainTomorrow ~ WindGustSpeed + WindSpeed9am + Humidity3pm + Pressure3pm + MinTemp + Ma
library(rpart.plot)
prp(mod_tree, type = 2, extra = 4)
```




```
pred_tree <- predict(mod_tree, newdata = test_set, type = "class")
cm <- confusionMatrix(pred_tree, test_set$RainTomorrow, positive = "Yes")
all_results <- rbind(all_results, cbind(data.frame(Model = 'Simple Tree Model'), as.data.frame(t(c(cm$by
all_results[3,]
```

```
##               Model Sensitivity Specificity        F1 Accuracy
## 3 Simple Tree Model   0.3597867   0.9613851 0.4818315 0.826512
```

```
pred_tree_prob <- predict(mod_tree, newdata = test_set)
pr <- prediction(pred_tree_prob[,2], test_set$RainTomorrow)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.6968523
```

4.6 Random Forest

Correlated variables are less of a problem for random Forests, so we used all predictors instead of selected ones.

```
library(randomForest)
mod_rf <- randomForest(RainTomorrow ~ ., data = train_set, family = binomial)
pred_rf <- predict(mod_rf, type = 'response', newdata = test_set)
cm <- confusionMatrix(data = pred_rf, reference = test_set$RainTomorrow, positive = "Yes")
all_results <- rbind(all_results, cbind(data.frame(Model = 'Simple Random Forest Model'), as.data.frame
all_results[4,]
```

```
##                         Model Sensitivity Specificity        F1 Accuracy
## 4 Simple Random Forest Model   0.5864178   0.9396302 0.6532716 0.860443
```

4.7 Using the caret package

The caret package provides consistency in order to train models from different packages. Available methods are described here : https://topepo.github.io/caret/train-models-by-tag.html

Caret also allows to easily use cross validation and to optimize hyperparameters for each type of model.

```
library(caret)
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)
```

4.8 Random Forest

```
Gridrf <-  expand.grid(mtry = c(5, 7, 9))
fit_caret_rf <- train(RainTomorrow ~ ., data = train_set, method = "rf", trControl = fitControl, tuneGr
predict_caret_rf <- predict(fit_caret_rf, test_set)
cm <- confusionMatrix(data = predict_caret_rf, reference = test_set$RainTomorrow, positive = "Yes")
all_results <- rbind(all_results, cbind(data.frame(Model = 'Caret rf Model'), as.data.frame(t(c(cm$byCl
all_results[5,]
```

```
##            Model Sensitivity Specificity        F1  Accuracy
## 5 Caret rf Model   0.4962359   0.9598441 0.6069442 0.8559072
```

4.9 Bayesian Generalized Linear Model

```
fit_caret_bayesglm <- train(RainTomorrow ~ WindGustSpeed + WindSpeed9am + Humidity3pm + Pressure3pm + M
predict_caret_bayesglm <- predict(fit_caret_bayesglm, test_set)
cm <- confusionMatrix(data = predict_caret_bayesglm, reference = test_set$RainTomorrow, positive = "Yes
all_results <- rbind(all_results, cbind(data.frame(Model = 'Bayesian Generalized Linear Model'), as.data
all_results[6,]
```

```
##                              Model Sensitivity Specificity       F1
## 6 Bayesian Generalized Linear Model   0.4716123    0.950281 0.573855
##    Accuracy
## 6 0.8429677
```

4.10 Stop parallel computation

```
stopCluster(cluster)
registerDoSEQ()
```

4.11 Ensemble model

Using several different models and combining them, with a majority vote in our case, can lead to better performance.

```
predictions <- data.frame(pred_glm2, pred_tree, pred_rf, predict_caret_rf, predict_caret_bayesglm)
library(prettyR)
final_prediction <- as.factor(apply(predictions, 1, Mode))
cm <- confusionMatrix(data = final_prediction, reference = test_set$RainTomorrow, positive = "Yes")
all_results <- rbind(all_results, cbind(data.frame(Model = 'Ensemble Majority Vote Model'), as.data.fra
all_results[7,]
```

```
##                          Model Sensitivity Specificity       F1  Accuracy
## 7 Ensemble Majority Vote Model   0.5244668   0.9489213 0.616576 0.8537623
```

5 Results

```
knitr::kable(all_results)
```

Model	Sensitivity	Specificity	F1	Accuracy
Fixed No Model	0.0000000	1.0000000	NA	0.7758087
Logistic Regression Model	0.6755019	0.8518854	0.6174468	0.8123418
Simple Tree Model	0.3597867	0.9613851	0.4818315	0.8265120
Simple Random Forest Model	0.5864178	0.9396302	0.6532716	0.8604430
Caret rf Model	0.4962359	0.9598441	0.6069442	0.8559072
Bayesian Generalized Linear Model	0.4716123	0.9502810	0.5738550	0.8429677
Ensemble Majority Vote Model	0.5244668	0.9489213	0.6165760	0.8537623

6 Conclusion