Weather Forecasting Project

Jeet Banik

09/01/2020

Introduction

This report is related to the "Choose Your Own Project" of the HarvardX: PH125.9x Data Science: Capstone course. In this project, the objective is to use Machine Learning to forecast the weather. The original data set can be found using the following link: https://www.kaggle.com/vonline9/weather-istanbul-data-20092019/data To provide satisfying results the weather is predicted by using different supervised machine learning algorithms and calculating its accuracy and thus finding the best model for forecasting weather more correctly based on the accuracy score.

Analysis

Getting Data

First step in any data analysis project is to get the data. In this project, a dataset "Istanbul Weather Data.csv" is downloaded from Kaggle and analysis is done on it. Required packages are loaded also beforehand.

```
# Dataset links downloadable from Kaggle #
# https://www.kaggle.com/vonline9/weather-istanbul-data-20092019/data
# https://www.kaggle.com/vonline9/weather-istanbul-data-20092019/download
# https://www.kaggle.com/vonline9/weather-istanbul-data-20092019/download/cr3DbJpST7Y7iCmTUn9R%2Fversio
# Need to login to Kaggle to download the dataset thus didn't download directly in R #

# Load required packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(Amelia)) install.packages("Amelia", repos = "http://cran.us.r-project.org")
if(!require(mice)) install.packages("mice", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(klaR)) install.packages("klaR", repos = "http://cran.us.r-project.org")
if(!require(httpuv)) install.packages("httpuv", repos = "http://cran.us.r-project.org")
if(!require(class)) install.packages("class", repos = "http://cran.us.r-project.org")
if(!require(Metrics)) install.packages("Metrics", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
```

```r
# Get the current working directory to copy the downloaded dataset here to import it
getwd()

# Read the data into a data frame 'df'
df <- read.csv("Istanbul Weather Data.csv")
View(df)
```

Basic Data Analysis

This gives the basic structure and summary statistics of the weather dataset.

```r
head(df)
```

```
##     DateTime            Condition Rain MaxTemp MinTemp  SunRise   SunSet MoonRise  MoonSet
## 1 02.09.2019        Partly cloudy  0.0      27      22 06:32:00 19:37:00  9:52:00 21:45:00
## 2 01.09.2019        Partly cloudy  0.0      27      22 06:31:00 19:38:00  8:37:00 21:13:00
## 3 31.08.2019 Patchy rain possible  0.5      26      22 06:30:00 19:40:00  7:21:00 20:40:00
## 4 30.08.2019        Partly cloudy  0.0      27      22 06:29:00 19:42:00   6:4:00  20:5:00
## 5 29.08.2019        Partly cloudy  0.0      27      23 06:27:00 19:43:00  4:47:00 19:26:00
## 6 28.08.2019                Sunny  0.0      28      24 06:26:00 19:44:00  3:34:00 18:41:00
##   AvgWind AvgHumidity AvgPressure
## 1      23          66        1012
## 2      21          66        1011
## 3      22          63        1015
## 4      20          64        1016
## 5      24          61        1015
## 6      27          58        1016
```

```r
str(df)
```

```
## 'data.frame':    3896 obs. of  12 variables:
##  $ DateTime   : chr  "02.09.2019" "01.09.2019" "31.08.2019" "30.08.2019" ...
##  $ Condition  : chr  "Partly cloudy" "Partly cloudy" "Patchy rain possible" "Partly cloudy" ...
##  $ Rain       : num  0 0 0.5 0 0 0 0 0 0 0 ...
##  $ MaxTemp    : int  27 27 26 27 27 28 30 30 30 30 ...
##  $ MinTemp    : int  22 22 22 22 23 24 24 24 24 24 ...
##  $ SunRise    : chr  "06:32:00" "06:31:00" "06:30:00" "06:29:00" ...
##  $ SunSet     : chr  "19:37:00" "19:38:00" "19:40:00" "19:42:00" ...
##  $ MoonRise   : chr  "9:52:00" "8:37:00" "7:21:00" "6:4:00" ...
##  $ MoonSet    : chr  "21:45:00" "21:13:00" "20:40:00" "20:5:00" ...
##  $ AvgWind    : int  23 21 22 20 24 27 27 25 20 19 ...
##  $ AvgHumidity: int  66 66 63 64 61 58 61 66 69 71 ...
##  $ AvgPressure: int  1012 1011 1015 1016 1015 1016 1016 1015 1015 1017 ...
```

```r
summary(df)
```

```
##    DateTime          Condition              Rain           MaxTemp         MinTemp
##  Length:3896        Length:3896        Min.   : 0.0000   Min.   :-3.00   Min.   :-5.00
##  Class :character   Class :character   1st Qu.: 0.0000   1st Qu.:12.00   1st Qu.: 8.00
##  Mode  :character   Mode  :character   Median : 0.0100   Median :18.00   Median :14.00
##                                        Mean   : 0.9468   Mean   :18.08   Mean   :13.77
```

```
##                                        3rd Qu.: 0.7200   3rd Qu.:25.00   3rd Qu.:20.00
##                                        Max.   :42.0000   Max.   :37.00   Max.   :26.00
##    SunRise             SunSet             MoonRise            MoonSet              AvgWind
##   Length:3896        Length:3896        Length:3896        Length:3896        Min.   : 2.00
##   Class :character   Class :character   Class :character   Class :character   1st Qu.:11.00
##   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median :16.00
##                                                                                Mean   :16.99
##                                                                                3rd Qu.:22.00
##                                                                                Max.   :56.00
##    AvgHumidity     AvgPressure
##   Min.   :40.00   Min.   : 992
##   1st Qu.:65.00   1st Qu.:1011
##   Median :71.00   Median :1015
##   Mean   :71.41   Mean   :1015
##   3rd Qu.:78.00   3rd Qu.:1019
##   Max.   :97.00   Max.   :1038
```

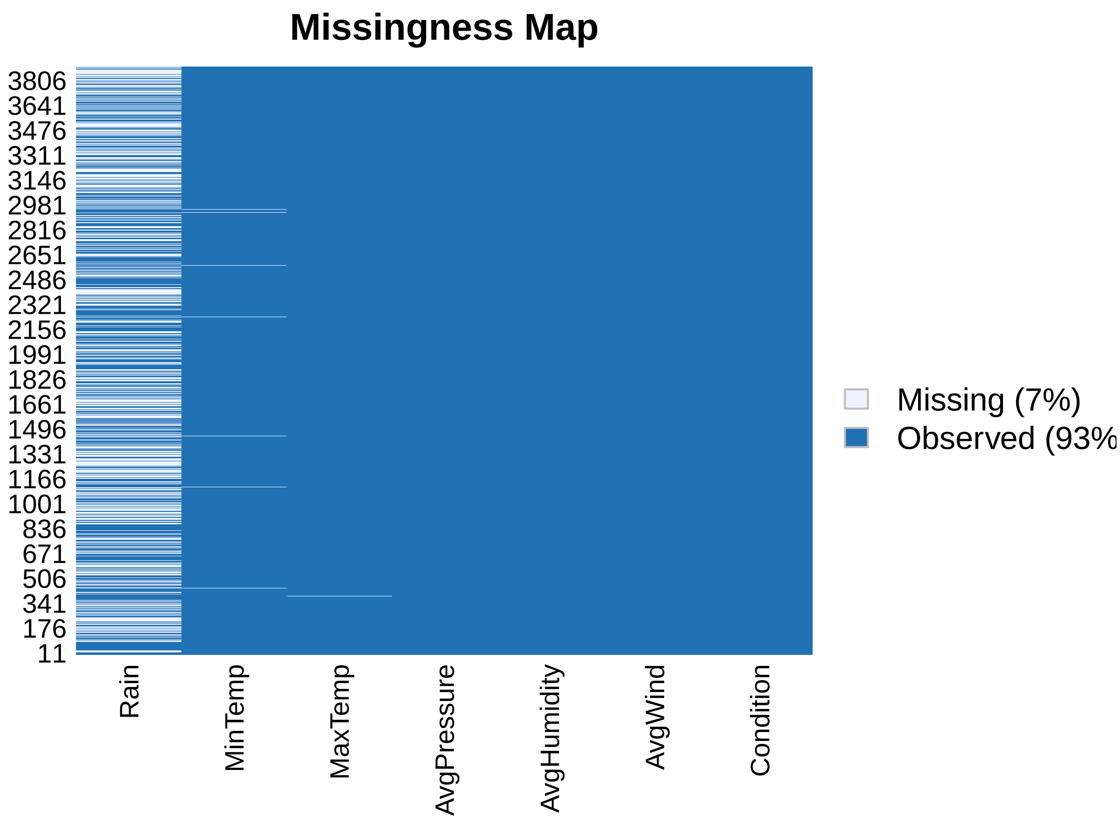
First Model: Naive Bayes Model

In machine learning, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector representing some n features (independent variables), it assigns to this instance probabilities, for each of k possible outcomes or classes. ### Data Exploration

```r
# Create a copy of the original data frame to work with in this model
dat1 <- df

# Data cleaning
dat1 <- dat1[,c(-1,-6:-9)]
dat1$Condition <- ifelse(dat1$Condition=="Sunny", T, F)
dat1$Condition <- factor(dat1$Condition, levels=c(F, T))

head(dat1)
```

```
##   Condition Rain MaxTemp MinTemp AvgWind AvgHumidity AvgPressure
## 1     FALSE  0.0      27      22      23          66        1012
## 2     FALSE  0.0      27      22      21          66        1011
## 3     FALSE  0.5      26      22      22          63        1015
## 4     FALSE  0.0      27      22      20          64        1016
## 5     FALSE  0.0      27      23      24          61        1015
## 6      TRUE  0.0      28      24      27          58        1016
```

```r
# Convert '0' values into NA
dat1[,2:4][dat1[,2:4]==0] <- NA

# Visualize the missing data NA
missmap(dat1)
```




```r
# Use mice function to predict the missing values
m <- mice(dat1[, c("Rain","MinTemp","MaxTemp")], method='rf')
```

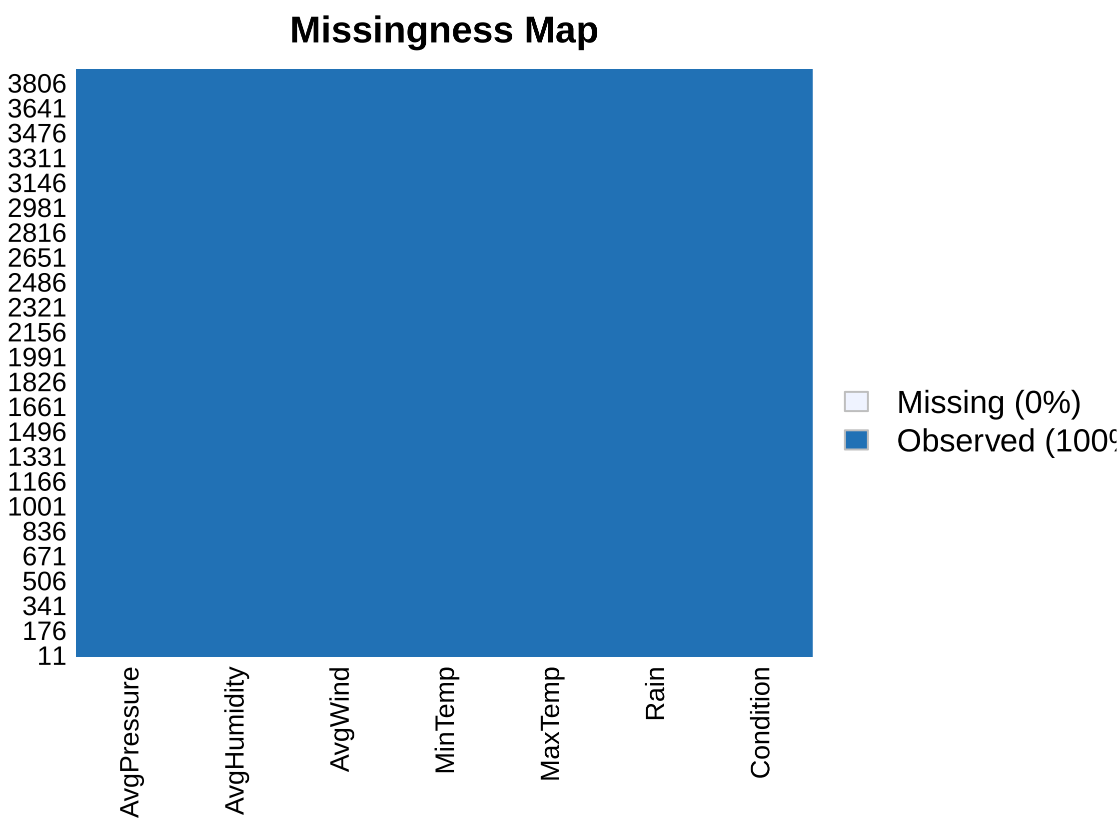
```
##
##  iter imp variable
##   1   1  Rain  MinTemp  MaxTemp
##   1   2  Rain  MinTemp  MaxTemp
##   1   3  Rain  MinTemp  MaxTemp
##   1   4  Rain  MinTemp  MaxTemp
##   1   5  Rain  MinTemp  MaxTemp
##   2   1  Rain  MinTemp  MaxTemp
##   2   2  Rain  MinTemp  MaxTemp
##   2   3  Rain  MinTemp  MaxTemp
##   2   4  Rain  MinTemp  MaxTemp
##   2   5  Rain  MinTemp  MaxTemp
##   3   1  Rain  MinTemp  MaxTemp
##   3   2  Rain  MinTemp  MaxTemp
##   3   3  Rain  MinTemp  MaxTemp
##   3   4  Rain  MinTemp  MaxTemp
##   3   5  Rain  MinTemp  MaxTemp
##   4   1  Rain  MinTemp  MaxTemp
##   4   2  Rain  MinTemp  MaxTemp
##   4   3  Rain  MinTemp  MaxTemp
##   4   4  Rain  MinTemp  MaxTemp
##   4   5  Rain  MinTemp  MaxTemp
##   5   1  Rain  MinTemp  MaxTemp
```

```
##   5   2  Rain  MinTemp  MaxTemp
##   5   3  Rain  MinTemp  MaxTemp
##   5   4  Rain  MinTemp  MaxTemp
##   5   5  Rain  MinTemp  MaxTemp
```

```r
m1 <- complete(m)

# Move the predicted missing values into the main dataset
dat1$Rain <- m1$Rain
dat1$MinTemp <- m1$MinTemp
dat1$MaxTemp <- m1$MaxTemp
```

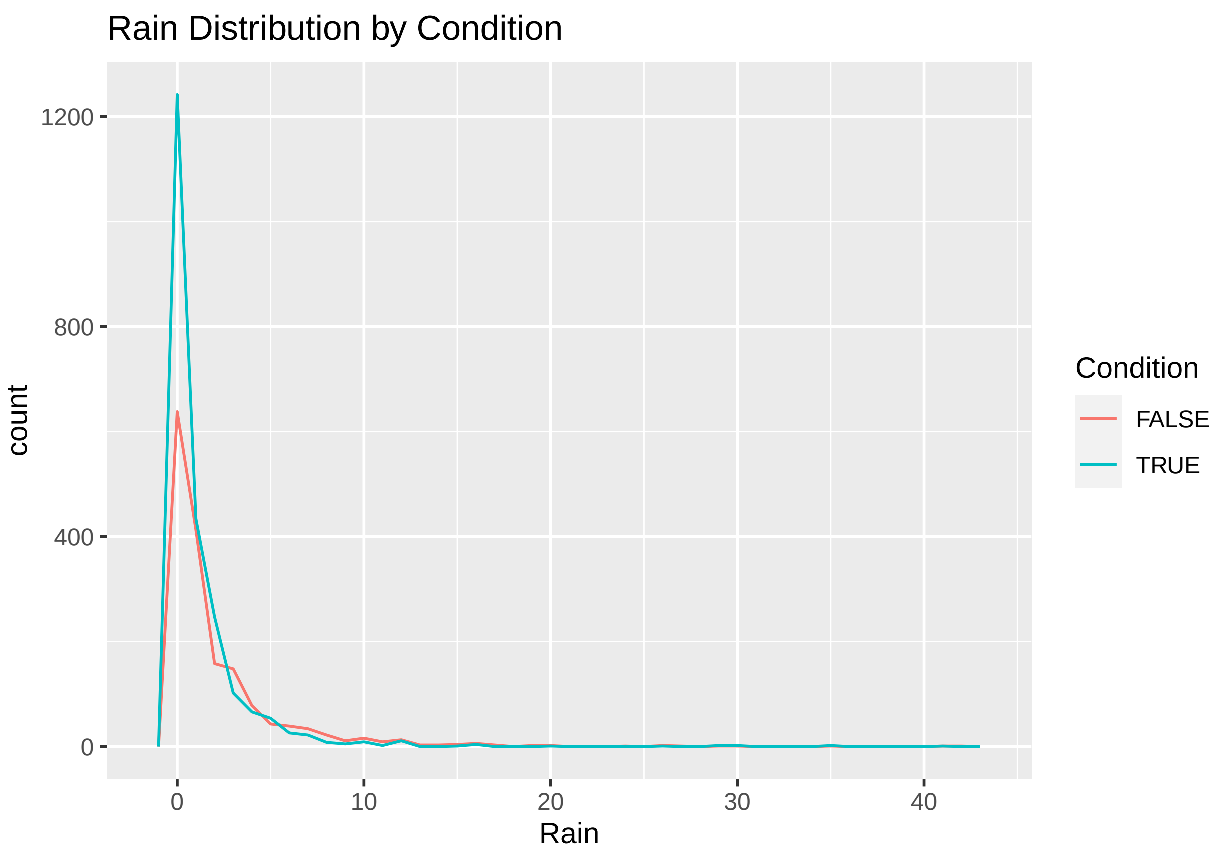
Data Visualization

```r
missmap(dat1)
```



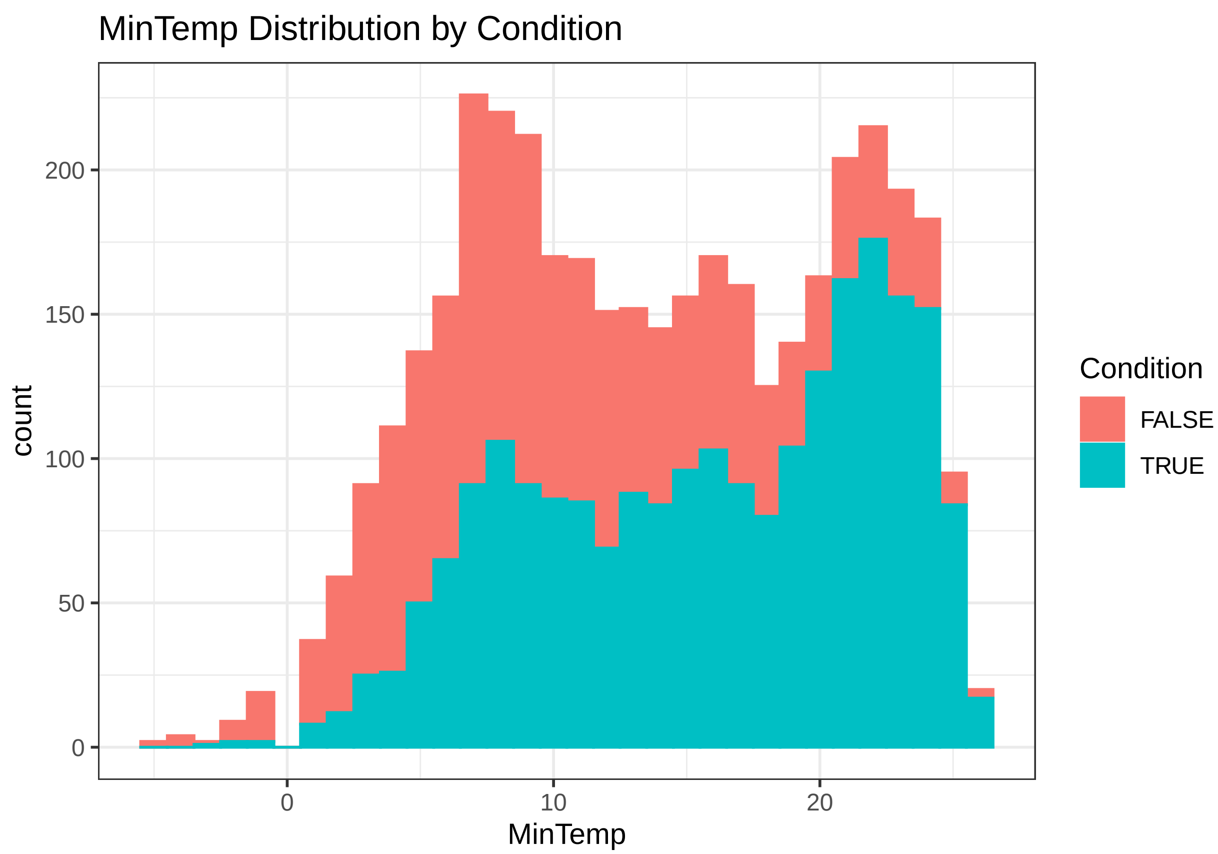
Above plot shows that there are no more missing values in the data frame. Distribution of rain by condition:

```r
ggplot(dat1, aes(Rain, colour=Condition)) +
  geom_freqpoly(binwidth=1) + labs(title="Rain Distribution by Condition")
```



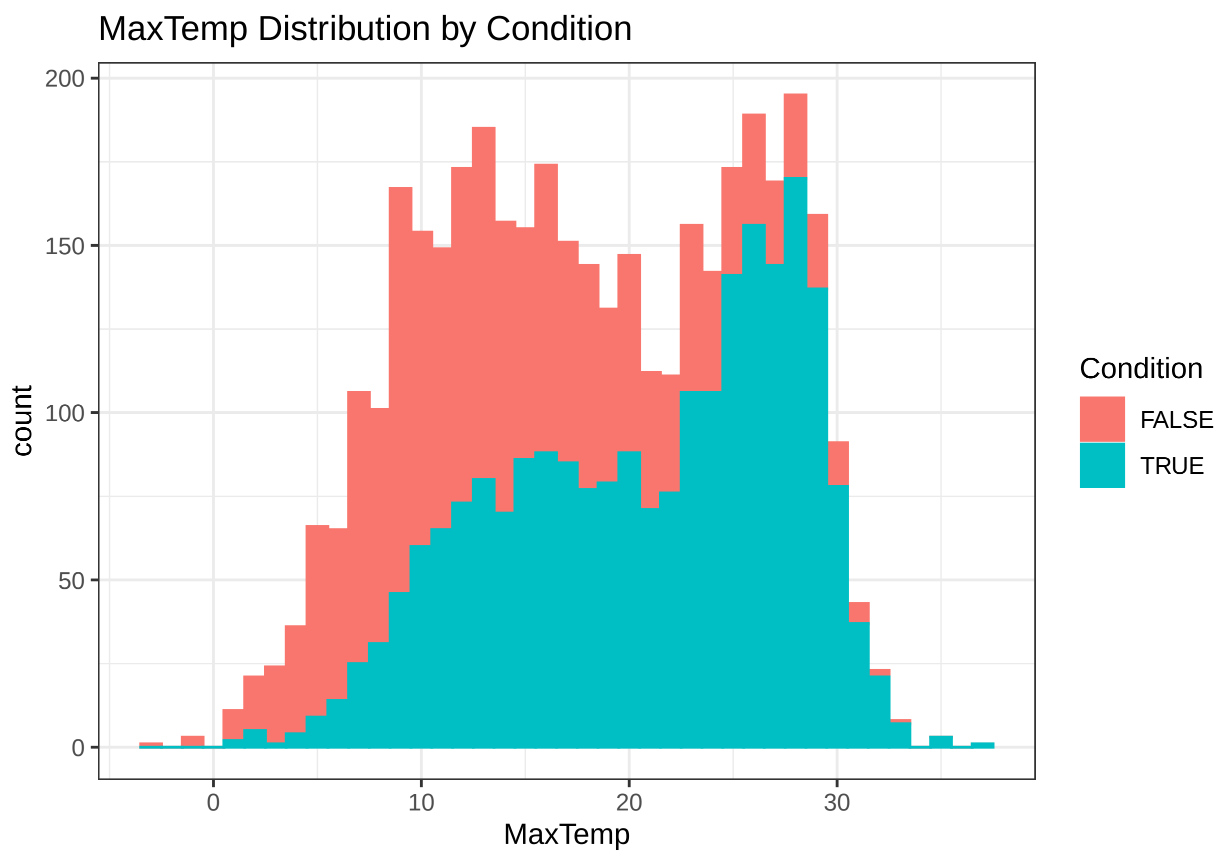
Distribution of minimum tempearture by condition:

```
ggplot(dat1, aes(x=MinTemp, fill=Condition, color=Condition)) +
  geom_histogram(binwidth=1) + labs(title="MinTemp Distribution by Condition") +
  theme_bw()
```



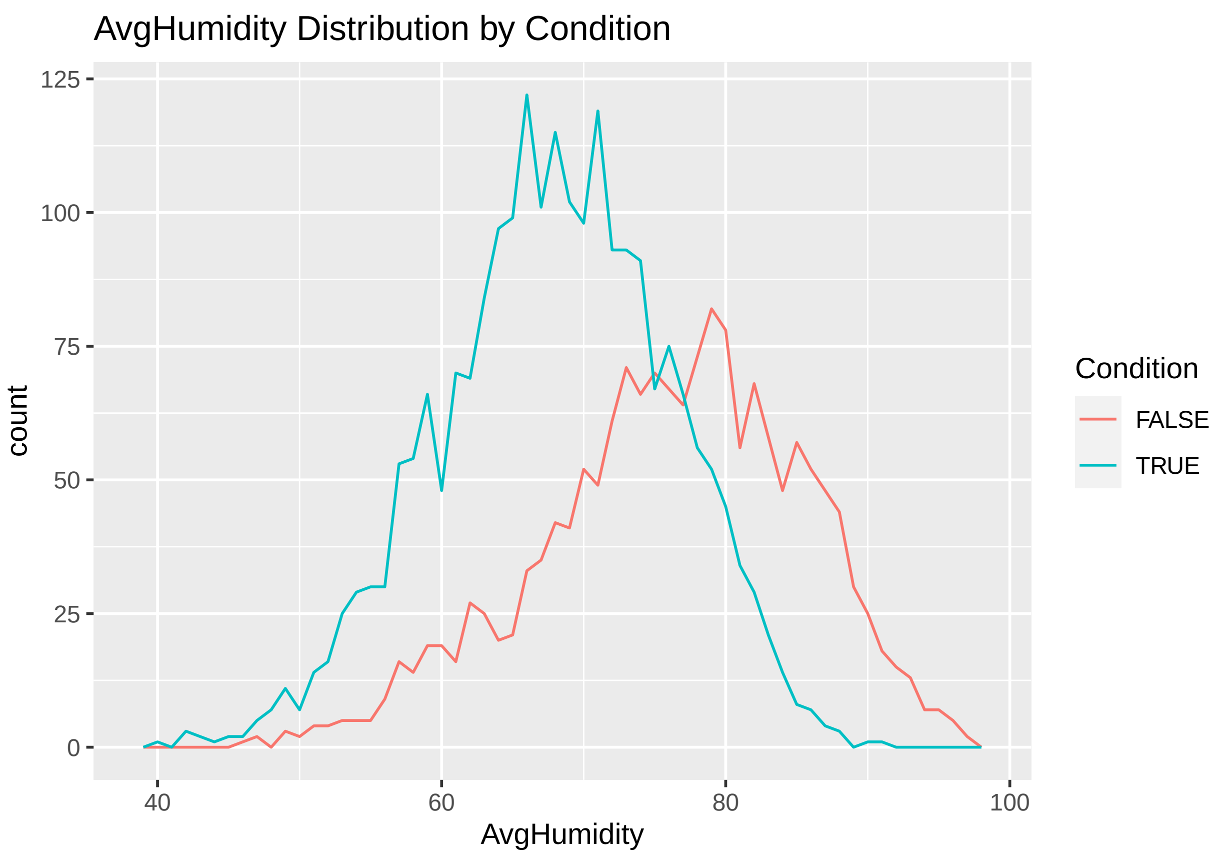
Distribution of maximum temperature by condition:

```
ggplot(dat1, aes(x=MaxTemp, fill=Condition, color=Condition)) +
  geom_histogram(binwidth=1) + labs(title="MaxTemp Distribution by Condition") +
  theme_bw()
```



Distribution of average humidity by condition:

```
ggplot(dat1, aes(AvgHumidity, colour=Condition)) +
  geom_freqpoly(binwidth=1) + labs(title="AvgHumidity Distribution by Condition")
```



Calculate the accuracy of the model's prediction

```
# Split the data into train and test datasets
indextrain <- createDataPartition(y=dat1$Condition, p=0.8, list=F)
train1 <- dat1[indextrain,]
test1 <- dat1[-indextrain,]

# Check dimensions of the split
prop.table(table(dat1$Condition)) * 100
```

```
##
##   FALSE    TRUE
## 42.4538 57.5462
```

```
prop.table(table(train1$Condition)) * 100
```

```
##
##    FALSE     TRUE
## 42.46312 57.53688
```

```
prop.table(table(test1$Condition)) * 100
```

```
##
##    FALSE     TRUE
## 42.41645 57.58355
```

```r
# Create objects x and y holding the predictor and the response variables respectively
x = train1[,-1]
y = train1$Condition

# Apply Naive Bayes
nb_model <- naiveBayes(Condition ~ ., data=train1)
summary(nb_model)
```

```
##           Length Class  Mode
## apriori   2      table  numeric
## tables    6      -none- list
## levels    2      -none- character
## isnumeric 6      -none- logical
## call      4      -none- call
```

```r
# Predict test set
predict <- predict(nb_model, newdata=test1[-1])

# Get the confusion matrix to see the accuracy value and other parameter values
new1 <- data.frame("Rain"=44,"MaxTemp"=29,"MinTemp"=23,"AvgWind"=19,"AvgHumidity"=57,"AvgPressure"=1017
c1 <- predict(nb_model, new1)

if (c1==TRUE) {
  print('Sunny')
} else {
  print('Rainy')
}
```

```
## [1] "Rainy"
```

```r
# Calculate the accuracy
a1 <- mean(test1[,1]==predict)
a1
```

```
## [1] 0.7133676
```

A data frame 'acc' is created to store the accuracy scores.

```r
acc <- data.frame(Method="Naive Bayes Model", Accuracy=a1)
acc
```

```
##              Method  Accuracy
## 1 Naive Bayes Model 0.7133676
```

Second Model: KNN Model

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. ### Data Exploration

```r
# Create a copy of the original data frame to work with in this model
dat2 <- df

# Data cleaning
dat2 <- dat2[,c(-1,-6,-7,-8,-9)]
dat2$AvgWind <- as.numeric(dat2$AvgWind)
dat2$Rain <- as.numeric(dat2$Rain)
dat2$MinTemp <- as.numeric(dat2$MinTemp)
dat2$AvgHumidity <- as.numeric(dat2$AvgHumidity)
dat2$MaxTemp <- as.numeric(dat2$MaxTemp)
dat2$AvgPressure<-as.numeric(dat2$AvgPressure)

head(dat2)
```

```
##              Condition Rain MaxTemp MinTemp AvgWind AvgHumidity AvgPressure
## 1        Partly cloudy  0.0      27      22      23          66        1012
## 2        Partly cloudy  0.0      27      22      21          66        1011
## 3 Patchy rain possible  0.5      26      22      22          63        1015
## 4        Partly cloudy  0.0      27      22      20          64        1016
## 5        Partly cloudy  0.0      27      23      24          61        1015
## 6                Sunny  0.0      28      24      27          58        1016
```

Calculate the accuracy of the model's prediction

```r
r <- sample(1:nrow(dat2), 0.9*nrow(dat2))

# Create normalization function
norm <- function(x){
  (x-min(x))/(max(x)-min(x))}

# Run normalization on first 10 columns of the dataset as they are the predictors
dat2_norm <- as.data.frame(lapply(dat2[,c(2,3,4,5,6,7)], norm))
summary(dat2_norm)
```

```
##       Rain               MaxTemp          MinTemp          AvgWind         AvgHumidity
##  Min.   :0.0000000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000000   1st Qu.:0.3750   1st Qu.:0.4194   1st Qu.:0.1667   1st Qu.:0.4386
##  Median :0.0002381   Median :0.5250   Median :0.6129   Median :0.2593   Median :0.5439
##  Mean   :0.0225427   Mean   :0.5271   Mean   :0.6056   Mean   :0.2776   Mean   :0.5511
##  3rd Qu.:0.0171429   3rd Qu.:0.7000   3rd Qu.:0.8065   3rd Qu.:0.3704   3rd Qu.:0.6667
##  Max.   :1.0000000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##   AvgPressure
##  Min.   :0.0000
##  1st Qu.:0.4130
##  Median :0.5000
##  Mean   :0.5061
##  3rd Qu.:0.5870
##  Max.   :1.0000
```

```r
train2 <- dat2_norm[r,]

# Extract test set
test2 <- dat2_norm[-r,]

# Extract 5th column of train dataset as it'll be used as 'cl' argument in knn function.
t1 <- dat2[r,1]

# Extract 5th column of test dataset to measure the accuracy
t2 <- dat2[-r,1]

# Run knn function
p <- knn(train2, test2, cl=t1, k=6)

# Calculate the accuracy
a2 <- mean(p==t2)
a2
```

```
## [1] 0.6102564
```

```r
# Save accuracy results in the data frame 'acc'
acc <- bind_rows(acc, data_frame(Method="KNN Model", Accuracy=a2))
acc
```

```
##              Method  Accuracy
## 1 Naive Bayes Model 0.7133676
## 2         KNN Model 0.6102564
```

Accuracy decreased significantly and thus another model is implemented for prediction. ## Third Model: Random Forest Model Random forests, otherwise known as the random forest model, is a method for classification and other tasks. It operates from decision trees and outputs classification of the individual trees. Random forests correct for the habit of decision trees to overfit to their training set. ### Data Exploration

```r
set.seed(100, sample.kind="Rounding")

# Create a copy of the original data frame to work with in this model
dat3 <- df

# Data cleaning
dat3 <- dat3[,c(-1,-6,-7,-8,-9)]
dat3$Condition <- ifelse(dat3$Condition =="Sunny", 1, 0)
dat3$Condition <- factor(dat3$Condition, levels = c(0, 1))
dat3$AvgWind <- as.integer(dat3$AvgWind)
dat3$Rain <- as.integer(dat3$Rain)
dat3$MinTemp <- as.integer(dat3$MinTemp)
dat3$AvgHumidity <- as.integer(dat3$AvgHumidity)
dat3$MaxTemp <- as.integer(dat3$MaxTemp)
dat3$AvgPressure <- as.integer(dat3$AvgPressure)

head(dat3)
```

```
##   Condition Rain MaxTemp MinTemp AvgWind AvgHumidity AvgPressure
## 1         0    0      27      22      23          66        1012
## 2         0    0      27      22      21          66        1011
## 3         0    0      26      22      22          63        1015
## 4         0    0      27      22      20          64        1016
## 5         0    0      27      23      24          61        1015
## 6         1    0      28      24      27          58        1016
```

Calculate the accuracy of the model's prediction

```
# Split the dataset into train and test
train3<-dat3[1:3000,]
test3<-dat3[3001:3854,]

# Apply Random Forest
rf_model <- randomForest(Condition ~., data = train3)
rf_model
```

```
##
## Call:
##  randomForest(formula = Condition ~ ., data = train3)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 24.07%
## Confusion matrix:
##     0    1 class.error
## 0 842  428   0.3370079
## 1 294 1436   0.1699422
```

```
importance(rf_model)
```

```
##             MeanDecreaseGini
## Rain                117.2890
## MaxTemp             274.3223
## MinTemp             219.9752
## AvgWind             261.5101
## AvgHumidity         319.4827
## AvgPressure         223.0896
```

```
# Predict test set
pred <- predict(rf_model, newdata=test3[,-1], type ='class')

# Get the confusion matrix to see the accuracy value and other parameter values
new2 <- data.frame("Rain"= 0,"MaxTemp"= 29,"MinTemp"= 23,"AvgWind"= 19,"AvgHumidity"= 57,"AvgPressure"=
c2 <- predict(rf_model, new2)

if (c2==1) {
  print('Sunny')
} else  {
```

```r
  print('Rainy')
}
```

```
## [1] "Rainy"
```

```r
# Calculate the accuracy
a3 <- auc(pred,test3$Condition)
a3
```

```
## [1] 0.8079227
```

```r
# Save accuracy results in the data frame 'acc'
acc <- bind_rows(acc, data_frame(Method="Random Forest Model", Accuracy=a3))
acc
```

```
##                Method  Accuracy
## 1   Naive Bayes Model 0.7133676
## 2           KNN Model 0.6102564
## 3 Random Forest Model 0.8079227
```

Results

We can check the accuracy scores for the various models trained from Naive Bayes Model, KNN model and Random Forest Model. The resultant accuracy of predictions for all the four models are as follows:

```
##                Method  Accuracy
## 1   Naive Bayes Model 0.7133676
## 2           KNN Model 0.6102564
## 3 Random Forest Model 0.8079227
```

As we see from the results, Random Forest Model is the most accurate model in forecasting the weather with an accuracy score of 0.8079227, which means the predictions of this model has an accuracy of 80.79% or ~81%.

Conclusion

For forecasting the weather dataset, 3 separate machine learning models are used in this project and each of its accuracy scores are calculated and compared to find out the best model for weather forecast here. The first model, Naive Bayes Model, scored an accuracy of 0.7133676 which is a decent score but yet needs to be improved. The second model, KNN Model, scored an accuracy of 0.6102564 which decreases the accuracy score from the previous model significantly and thus should not be taken into consideration. The third and final model, Random Forest Model, scored an accuracy of 0.8079227, which has improved by a lot compared to the previous model and is the best out of all the models used here. Thus, the Random Forest Model, with an accuracy of 80.79% or 81%, should be considered for forecasting the weather dataset most correctly. Other different machine learning models could also improve the results further, but hardware limitations, such as the RAM, are a constraint.