University of Padua
Departments of Mathematics

Master's Degree in Data Science

Venice's Air Quality Prediction

Laura Legrottaglie

ID: 2073222

Academic Year 2022/2023

Introduction

The quality of the air that we breath is a crucial aspect of everyday life. Just think that air pollution causes approximately 400000 deaths in Europe every year and over 50000 in Italy due to diseases on the respiratory and cardiovascular systems. One of the most polluted areas in Italy is Po Valley where several cities frequently exceed the regulatory limits for fine dust particles (PM10 and PM2.5), sometimes doubling the times allowed. One of these cities is Venice that in 2022 recorded 70 days with the PM10 values above $50\,\mu g/m^3$ against the 35 days a year allowed by the European normative.

This project focuses on analyzing the meteorological data of the city of Venice and, based on them, trying to predict whether the next day's air quality will be below the normative limit of ($50\,\mu g/m^3$) or above it.

Predicting air quality can be beneficial in several ways. Firstly, knowing in advance the air quality might help people with health conditions and respiratory diseases to detect critical days avoiding outdoor activities and taking protective measures such as wearing masks. Secondly, the detection of particulate matter such as PM10 necessitates specialized equipment, which is not only costly to set up and maintain but also sparsely distributed. Conversely, meteorological conditions can be easily measured with affordable instrumentation providing a more accessible and cost-effective approach.

The study's interpretative purpose is, thus, determining if the meteorological data can be used to predict the air quality and identifying what are the significant weather predictors. Besides these goals, the other objective is to evaluate and compare the performances of different models, focusing on their efficiency in accurately predicting air quality.

1 Dataset construction

1.1 Weather datasets

The dataset for this analysis has been created using weather data from the well-known meteorological site ilMeteo.it. It contains data of a four-year span, specifically the years 2018, 2019, 2021, and 2022, and focuses on the city of Venice, specifically drawing data from the *Venice Tessera* meteorological station.

```
# Import the libraries
library(dplyr)
library(ggplot2)
library(glmnet)
library(readxl)
library(corrplot)
library(knitr)
library(pander)
library(caret)
library(gridExtra)
library(igraph)
library(glasso)
library(correlation)
library(ROSE)
library(car)
library(stats)
library(kableExtra)
library(MASS)
library(e1071)
library(pROC)
library(devtools)
library(dprep)
```

The data of each month are uploaded and then combined in a complete dataset related to the specific year.

```
#Upload and combine the weather data of 2018's months
January2018 <- read.csv("Venezia-2018-Gennaio.csv",sep=";")
February2018 <- read.csv("Venezia-2018-Febbraio.csv",
                         sep=";")
March2018 <- read.csv("Venezia-2018-Marzo.csv",
                      sep=";")
April2018 <- read.csv("Venezia-2018-Aprile.csv",
                      sep=";")
May2018 <- read.csv("Venezia-2018-Maggio.csv",
                    sep=";")
June2018 <- read.csv("Venezia-2018-Giugno.csv",
                     sep=";")
July2018 <- read.csv("Venezia-2018-Luglio.csv",
                     sep=";")
August2018 <- read.csv("Venezia-2018-Agosto.csv",
                       sep=";")
September2018 <- read.csv("Venezia-2018-Settembre.csv",
                          sep=";")
October2018 <- read.csv("Venezia-2018-Ottobre.csv",
                        sep=";")
November2018 <- read.csv("Venezia-2018-Novembre.csv",
                         sep=";")
December2018 <- read.csv("Venezia-2018-Dicembre.csv",
                         sep=";")
weather2018<- rbind(January2018, February2018, March2018, April2018, May2018,
                    June2018, July2018, August2018, September2018, October2018,
                    November2018, December2018)
#This process is repeated for the years 2019,2021 and 2022
```

For each year there are 365 observations and 15 variables. To solve some spelling issues and translate the names in English the attributes have been renamed to make them more readable.

```
#Rename variables of weather datasets
rename_variables <- function(df){
  colnames(df) <- c("Location","Date","T_med","T_max","T_min","Dew_point",
                    "Humidity","Visibility","Wind_speed_med",
                    "Wind_speed_max","Gust","Pressure","Pressure_med","Rain","Phenomena")
  return(df)
}
weather2018 <- rename_variables(weather2018)
weather2019 <- rename_variables(weather2019)
weather2021 <- rename_variables(weather2021)
weather2022 <- rename_variables(weather2022)
```

Let's now visualize the first 5 rows of 2018's dataset.

```
#5 rows of 2018 weather dataset
head(weather2018,5)
```

```
##   Location     Date T_med T_max T_min Dew_point Humidity Visibility
## 1  Venezia 1/1/2018     5     4     7         4       92          9
```

```
## 2  Venezia 2/1/2018     4     1     9         3       94         10
## 3  Venezia 3/1/2018     3     0     6         2       94          9
## 4  Venezia 4/1/2018     4     0     8         3       89         19
## 5  Venezia 5/1/2018     5     1    10         3       84         20
##   Wind_speed_med Wind_speed_max Gust Pressure Pressure_med Rain Phenomena
## 1             14             22    0     1011            0    0  pioggia
## 2              9             17    0     1011            0    0   nebbia
## 3             11             21    0     1010            0    0   nebbia
## 4             11             17    0     1007            0    0
## 5              6             13    0     1009            0    0
```

To enhance the comprehension of the dataset, the subsequent table, enumerates the variables and provides concise explanations for each of them.

Variable	Description
Location	Name of the city where the data have been recorded
Date	Day of the year when the data have been recorded
T_med	The average temperature of the day (°C)
T_max	The highest temperature recorded during the day (°C)
T_min	The lowest temperature recorded during the day(°C)
Dew_point	The temperature at which dew can form (°C) in the day
Humidity	Percentage of the amount of water vapor present in the air
Visibility	The distance at which objects can be clearly seen (km)
Wind_speed_med	The average wind speed for the day (km/h)
Wind_speed_max	The highest wind speed recorded in the day (km/h)
Gust	The increase in the strength of the wind (in km/h)
Pressure	The atmospheric pressure at the time of observation (mb)
Pressure_med	The average atmospheric pressure for the day (mb)
Rain	The amount of precipitation that fell during the day (mm)
Phenomena	Any significant weather events observed, such as thunderstorm, fog, snow, etc.

1.2 PM10 datasets

PM10 data are sourced from Arpa Veneto and contain the daily values for each year in question. The datasets of 2019 and 2021 are presented as an XLSX file and are imported through the function *read_excel*.

```
#Upload PM10 datasets
PM10_2018 <- read.csv("PM10_2018.txt",sep=" ",na.strings = "NA")
PM10_2019 <- read_excel("PM10_2019.xlsx")
PM10_2021 <- read_excel("PM10_2021.xlsx")
PM10_2022 <- read.csv("PM10_2022.txt",sep=" ",na.strings = "NA")
```

2018 and 2022 datasets are related to several stations (*Murano*, *Sacca Fisola*, *Rio Novo*, *Parco Bissuola* and *Via Tagliamento*), while 2019 and 2021 datasets contains only the data of *Parco Bissuola* station that is also the nearest to the meteorological station of *Venice Tessera* already considered for the weather.

```
#Let's take a look at the first 5 rows of 2018 and 2019 PM10 datasets
head(PM10_2018,5)
```

```
##         Date Murano SaccaFisola RioNovo ParcoBissuola viaTagliamento
## 1 01/01/2018     44          44      44            61             62
```

```
## 2 02/01/2018     46          40      43          42          55
## 3 03/01/2018     48          50      43          53          58
## 4 04/01/2018     39          39      35          44          35
## 5 05/01/2018     49          48      43          47          59
```

```
head(PM10_2019,5)
```

```
## # A tibble: 5 x 2
##   Date                PM10
##   <dttm>              <chr>
## 1 2019-01-01 00:00:00 51
## 2 2019-01-02 00:00:00 47
## 3 2019-01-03 00:00:00 32
## 4 2019-01-04 00:00:00 55
## 5 2019-01-05 00:00:00 74
```

Between all the stations of 2018 and 2022 dataset, only the stations of Parco Bissuola and Via Tagliamento have been considered. The values of the last station, in fact, will be useful in the upcoming analysis to replace some missing values.

```
#Select only the stations of interest for 2018 and 2022 datasets
PM10_2018 <- subset(PM10_2018, select= c(Date,ParcoBissuola,viaTagliamento))
PM10_2022 <- subset(PM10_2022, select= c(Date,ParcoBissuola,viaTagliamento))
```

2 Data Preprocessing

2.1 Handling missing values

The first step of data preprocessing is checking for the presence of missing values in each dataset.

```
#Checking for the presence of NA values in weather datasets
sum(is.na(weather2018))
```

```
## [1] 0
```

```
sum(is.na(weather2019))
```

```
## [1] 12
```

```
sum(is.na(weather2021))
```

```
## [1] 12
```

```
sum(is.na(weather2022))
```

```
## [1] 0
```

Specifically, it results that 2019 and 2021 datasets have respectively 12 NA values. Let's now take a look at how these missing values are spread out in the features.

```r
#Checking where are the missing values
sapply(weather2019, function(x) sum(is.na(x)))
```

```
##       Location           Date          T_med          T_max          T_min
##              0              0              1              1              1
##      Dew_point       Humidity     Visibility Wind_speed_med Wind_speed_max
##              1              1              1              1              1
##           Gust       Pressure   Pressure_med           Rain      Phenomena
##              1              1              1              1              0
```

```r
sapply(weather2021, function(x) sum(is.na(x)))
```

```
##       Location           Date          T_med          T_max          T_min
##              0              0              1              1              1
##      Dew_point       Humidity     Visibility Wind_speed_med Wind_speed_max
##              1              1              1              1              1
##           Gust       Pressure   Pressure_med           Rain      Phenomena
##              1              1              1              1              0
```

There are only two days where the data are missing. It's possible to identify the specific dates and then proceed to replace the missing values using the data of the related month and year for imputation. Since some of the features contain outliers, as the futher analysis will show, the best approach is employing the median imputing, which is a more robust measure of centrality than the mean in such cases. Hence, the missing values are substituted with the median of the pertinent month and year for each variable.

```r
#Replace NA values with the median of the specific month and year
na_row_2019 <- apply(weather2019,1,function(x) any(is.na(x)))
na_row_2021 <- apply(weather2021,1,function(x) any(is.na(x)))
#The row with missing values for 2019 dataset is
weather2019[na_row_2019,"Date"]
```

```
## [1] "5/2/2019"
```

```r
#The row with missing values for 2021 dataset is
weather2021[na_row_2021,"Date"]
```

```
## [1] "14/1/2021"
```

```r
#Median imputation
numeric_v <- colnames(weather2019[sapply(weather2019,is.numeric)])
weather2019[na_row_2019,numeric_v] <- sapply(February2019[,numeric_v],median,na.rm=T)
weather2021[na_row_2021,numeric_v] <- sapply(January2021[,numeric_v],median,na.rm=T)


#Let's now check if the handling of missing values has been perfomed correctly
sum(is.na(weather2019))
```

```
## [1] 0
```

```r
sum(is.na(weather2021))
```

```
## [1] 0
```

Regarding the PM10, the datasets containing missing values are the ones related to 2018 and 2022.

```r
#Checking for NA values in PM10 datasets. The reference station is Parco Bissuola
#when several stations are present
sum(is.na(PM10_2018$ParcoBissuola))
```

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

```r
sum(is.na(PM10_2019))
```

```
## [1] 0
```

```r
sum(is.na(PM10_2021))
```

```
## [1] 0
```

```r
sum(is.na(PM10_2022$ParcoBissuola))
```

```
## [1] 12
```

Given that *Parco Bissuola* is the reference pollution station for the analysis, the methodology for handling missing values deviates from the conventional mean or median substitution. Instead, to provide a more accurate imputation, the data from the nearby *Via Tagliamento* station are utilized, assuming similar environmental conditions due to its proximity.

```r
#Define a function for handle missing values and make the PM10 datasets uniform
#in the structure
preprocess_PM10 <- function(df){
  #Substitute the NA of Parco Bissuola with the values in Via Tagliamento
  df$ParcoBissuola[is.na(df$ParcoBissuola)] <-
    df$viaTagliamento[is.na(df$ParcoBissuola)]
  #Removing the unuseful variable Via Tagliamento
  df$viaTagliamento <- NULL
  #Rename ParcoBissuola column
  colnames(df)[colnames(df)=="ParcoBissuola"] <- "PM10"
  return(df)
  }

PM10_2018 <- preprocess_PM10(PM10_2018)
PM10_2022 <- preprocess_PM10(PM10_2022)

#Checking if we correctly replace the missing values
sum(is.na(PM10_2018))
```

```
## [1] 0
```

```r
sum(is.na(PM10_2022))
```

```
## [1] 0
```

2.2 Data cleaning and preparation

The weather datasets from various years can be combined into a final dataset.

```r
#Combining weather data into a final dataset
weather <- rbind(weather2018, weather2019, weather2021, weather2022)
```

To start the data cleaning and encoding process, let's take a preliminary review of the weather dataset.

```r
summary(weather)
```

```
##    Location             Date               T_med           T_max
##  Length:1460        Length:1460        Min.   :-3.00   Min.   :-4.0
##  Class :character   Class :character   1st Qu.: 7.00   1st Qu.: 4.0
##  Mode  :character   Mode  :character   Median :15.00   Median :11.0
##                                        Mean   :14.71   Mean   :10.8
##                                        3rd Qu.:22.00   3rd Qu.:18.0
##                                        Max.   :30.00   Max.   :27.0
##      T_min         Dew_point        Humidity       Visibility
##  Min.   :-2.00   Min.   : 0.00   Min.   :35.00   Min.   : 0.00
##  1st Qu.:11.00   1st Qu.: 4.00   1st Qu.:65.00   1st Qu.:15.00
##  Median :18.00   Median :10.00   Median :74.00   Median :19.00
##  Mean   :18.46   Mean   :10.16   Mean   :73.94   Mean   :16.75
##  3rd Qu.:26.00   3rd Qu.:16.00   3rd Qu.:84.00   3rd Qu.:20.00
##  Max.   :35.00   Max.   :24.00   Max.   :99.00   Max.   :24.00
##  Wind_speed_med  Wind_speed_max       Gust      Pressure     Pressure_med
##  Min.   : 4.00   Min.   :  5.00   Min.   :0   Min.   : 989   Min.   :0
##  1st Qu.: 8.00   1st Qu.: 15.00   1st Qu.:0   1st Qu.:1011   1st Qu.:0
##  Median :10.00   Median : 17.00   Median :0   Median :1015   Median :0
##  Mean   :10.76   Mean   : 19.36   Mean   :0   Mean   :1016   Mean   :0
##  3rd Qu.:12.00   3rd Qu.: 22.00   3rd Qu.:0   3rd Qu.:1020   3rd Qu.:0
##  Max.   :39.00   Max.   :100.00   Max.   :0   Max.   :1037   Max.   :0
##       Rain    Phenomena
##  Min.   :0   Length:1460
##  1st Qu.:0   Class :character
##  Median :0   Mode  :character
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
```

The summary of the dataset reveals several issues that need to be addressed. First of all, some covariates, like *Gust*,*Pressure_med* and *Rain* remain unchanged for all the occurrences, as well as *Location*, and can be removed since they don't bring any additional information to the analysis.

```r
#Remove unuseful variables
weather <- subset(weather, select = -c(Location,Gust,Pressure_med,Rain))
```

Another issue highlighted by the dataset summary is the incorrect encoding of the *Phenomena* and *Date* variables. The *Phenomena* covariate, currently formatted as character type, is converted to categorical factor. More specifically, *Phenomena* has 8 different levels with similarities among some. To ensure more robust and interpretable models and to avoid overfitting, the most significant categories are retained and the similar levels are merged.

```
#Table of the initial levels of Phenomena
table(weather$Phenomena)
```

```
##
##                                        nebbia                         neve
##                         821                           156                            3
##                    pioggia              pioggia nebbia                pioggia neve
##                         293                            57                            2
##          pioggia temporale  pioggia temporale nebbia
##                         126                             2
```

```
#Merging levels
weather$Phenomena <- case_when(
  grepl("temporale", weather$Phenomena) ~ "Storm",
  grepl("pioggia", weather$Phenomena) ~ "Rain",
  grepl("nebbia", weather$Phenomena) ~ "Fog",
  TRUE ~ "No event"
)
#Encoding Phenomena as a factor
weather$Phenomena <- as.factor(weather$Phenomena)
#The reference level is "No event"
weather$Phenomena <- relevel(weather$Phenomena, ref="No event")

#Table of Phenomena with the new levels
table(weather$Phenomena)
```

```
##
## No event      Fog     Rain    Storm
##      824      156      352      128
```

Initially recorded as a character type, the *Date* variable is firstly converted into a Date format. Subsequently, essential components such as the year, month, and day of the week are extracted and encoded as factors based on the rationale that these temporal aspects could potentially influence PM10 levels.

```
#Encoding the Date variable in the correct type for all the datasets
weather$Date <- as.Date(weather$Date,format = "%d/%m/%Y")

PM10_2018$Date <- as.Date(PM10_2018$Date,format = "%d/%m/%Y")
PM10_2019$Date <- as.Date(PM10_2019$Date,format = "%d/%m/%Y")
PM10_2021$Date <- as.Date(PM10_2021$Date,format = "%d/%m/%Y")
PM10_2022$Date <- as.Date(PM10_2022$Date, format ="%d/%m/%Y")

#Extracting the year and encoding as factor
weather$Year <- as.integer(format(weather$Date,"%y"))
weather$Year <- as.factor(weather$Year)
#Extracting the month and encoding as factor
weather$Month <- as.integer(format(weather$Date, "%m"))
```

```r
weather$Month <- as.factor(weather$Month)

#Extracting the day of the week
#Getting the local time system
old_locale <- Sys.getlocale("LC_TIME")
#Setting the time system to USA
Sys.setlocale("LC_TIME","en_US")
```

```
## [1] "en_US"
```

```r
weather$Day_of_week <- weekdays(weather$Date)
#Setting back to the local time system
Sys.setlocale("LC_TIME",old_locale)
```

```
## [1] "Italian_Italy.utf8"
```

```r
#Encoding the day of the week as factor
weather$Day_of_week <- as.factor(weather$Day_of_week)
```

2.3 Final dataset

After all these preliminar operations, the PM10 datasets can be merged into a single one. The binary response variable, named *BadAirQuality*, is then is created. This variable is determined by the PM10 values, where it is assigned a value of 0 if the PM10 level of the following day is below ($50\,\mu g/m^3$), and 1 if it exceeds this threshold. The analysis spans four non-consecutive years: 2018, 2019, 2021, and 2022. Due to this non-sequential timeframe, certain dates, such as December 31st of 2019 and 2022, lack corresponding PM10 data. Consequently, the BadAirQuality value for these dates is marked as NA and these instances are excluded from the analysis.

```r
#Merging all the PM10 datasets
PM10_data <- rbind(PM10_2018,PM10_2019,PM10_2021,PM10_2022)
#Encoding the PM10 variable in the correct type (for some datasets it is a character)
PM10_data$PM10 <- as.integer(PM10_data$PM10)
#Finding occurrences where the PM10 value are greater than 50
mask <- as.integer(PM10_data$PM10>=50)
#Shifting by one occurrence:
#this causes a lack of information for the last day
sup <- c(mask[-1],NA)
#Creating the response variable
PM10_data$BadAirQuality <- sup
#Encoding it as factor
PM10_data$BadAirQuality <- as.factor(PM10_data$BadAirQuality)
#Day with missing information
PM10_data$BadAirQuality[PM10_data$Date=='2019-12-31'] <- NA
#Removing occurrences with missing values
PM10_data <- na.omit(PM10_data)
```

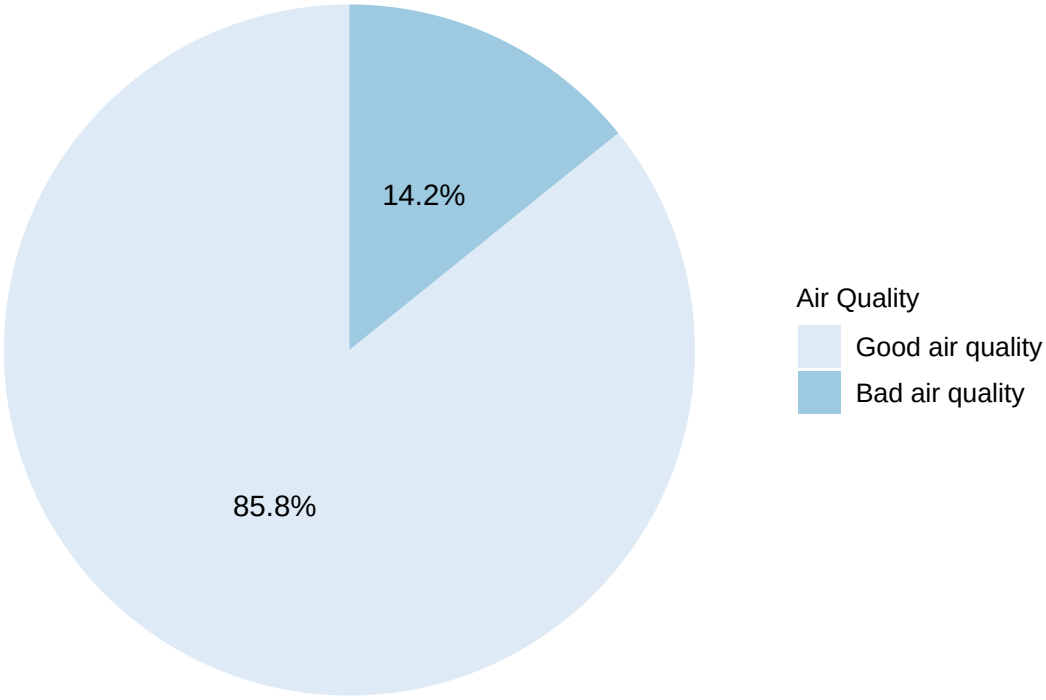
Finally, the weather and PM10 datasets can be merged through an equi-join which is based on the common variable *Date*. After that, since the extraction of all the related information to *Date* has been already performed, the variable is removed.

```r
#Constructing the final dataset
data <- merge(weather,PM10_data)
#Removing Date
data$Date <- NULL
```

2.4 Class balance check

It's important to highlight that the response variable *BadAirQuality* is heavily unbalanced towards the good quality air value 0. This imbalance could potentially bias the predictive models towards the more prevalent class. To mitigate this issue, class balancing techniques, such as undersampling, will be employed in further analysis.

```r
#Pie chart: balance class check
data_pie <- data %>%
  group_by(BadAirQuality) %>%
  summarize(Frequency = n()) %>%
  mutate(Percent = paste0(round(Frequency / sum(Frequency) * 100, 2), "%"))

# Plotting the pie chart
ggplot(data_pie, aes(x = "", y = Frequency, fill = BadAirQuality)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  theme_void() +
  geom_text(aes(label = Percent), position = position_stack(vjust = 0.5))+
  theme(legend.title = element_text(size = 10), legend.text = element_text(size = 10)) +
  scale_fill_brewer(palette = "Blues", labels = c("Good air quality", "Bad air quality")) +
  labs(fill = "Air Quality")
```



2.5 Train and test split

To evaluate the performances of the models analysed to unseen data, the dataset is divided into training (80%) and test sets (20%). The division is performed in a stratified manner to guarantee that the proportion of each class in the original dataset is maintained in both training and test.

```
#Train-test split: 80% for training and 20% for test
set.seed(123)
train_index <- createDataPartition(data$BadAirQuality, p=0.8,list=FALSE)
train_data <- data[train_index,]
test_data <- data[-train_index,]
```

The proportion of the classes is the same of the original dataset.

```
#Checking if the proportions of classes in training and test datasets are the same of the original one
prop.table(table(train_data$BadAirQuality))
```

```
##
##         0         1
## 0.8577549 0.1422451
```

```
prop.table(table(test_data$BadAirQuality))
```

```
##
##         0         1
## 0.8591065 0.1408935
```

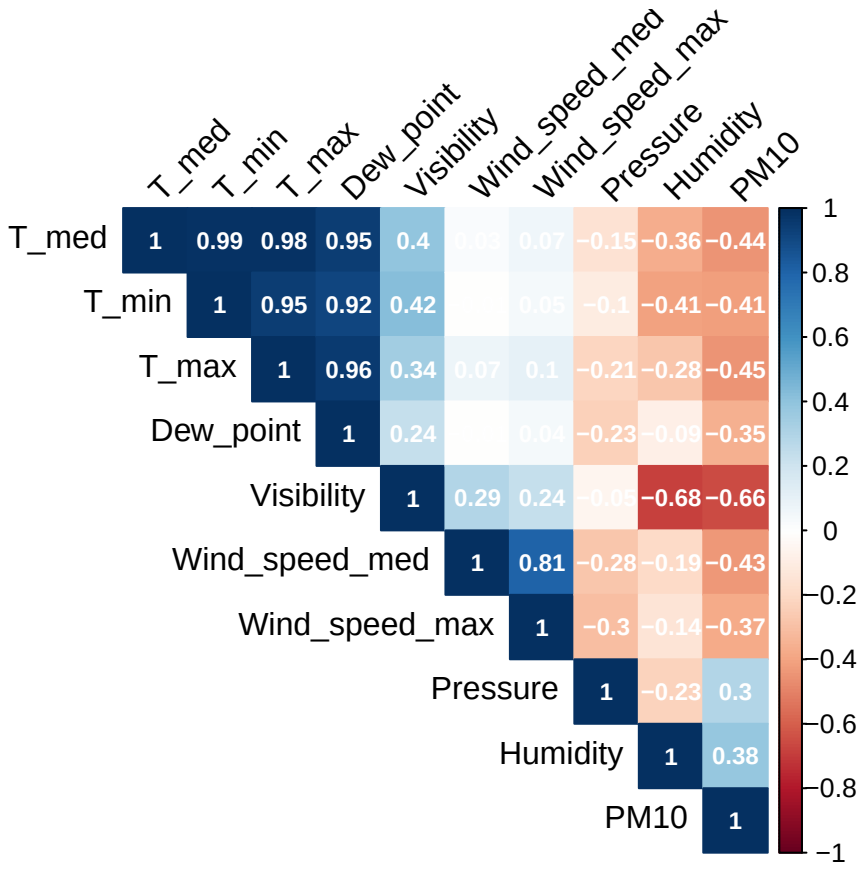
3 Explanatory Data Analysis

Explanatory Data Analysis is employed to initially investigate the behavior of the covariates, understand their distributions and examine the relationships between them.

3.1 Correlation

The initial phase of this analysis involves examining the correlations among the variables. Correlation is a statistical measure that quantifies the linear association between two covariates. It essentially indicates how much one variable tends to change in response to a change in another. Let's take a look at the correlation matrix to find out what are the most correlated variables.

```
#Extracting the numerical variables
num_features <- subset(train_data, select=-c(Phenomena, Month, Year, Day_of_week, BadAirQuality))
#Correlation matrix
cor_matrix <- cor(num_features)
#Plot the correlation matrix
corrplot(cor_matrix,method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45, addCoef.col = "white",
         number.cex=0.75)
```



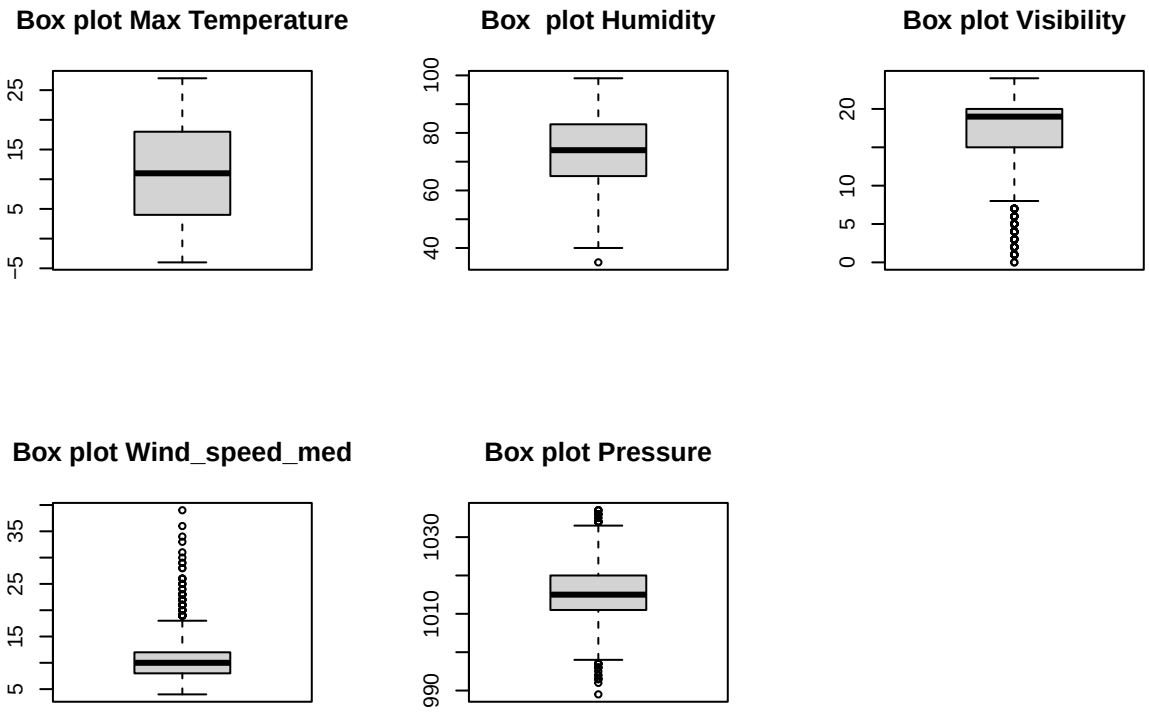
The correlation matrix reveals a high degree of linear association among several variables. *T_med*, *T_min*, *T_max* and *Dew_point* all exhibit Pearson correlation coefficients exceeding 0.9, indicating a strong positive relationship. This is an expected outcome, as these variables are all related to the temperature conditions. Similarly, also *Wind_speed_max* and *Wind_speed_med* are heavily linearly associated. In addition, some considerations about the *Visibility* variable can be made. It's negative correlated with the *Humidity*, which aligns with meteorological principles. In fact, since the percentage of humidity indicates the amount of water vapor contained in the air at a specific temperature, as the humidity level rises, the air approaches its saturation point. After reaching saturation,any additional concentration of water vapor in the air or a small reduction in temperature can cause the condensation of the water vapor in fine dropplets of water, forming fog. These small water particles scatter light in all directions, reducing the visibility. Since the presence of highly correlated variables can become a problem during modeling making also more complicated the interpretability without add any meaningful information, it is prudent to retain only one variable from each group of closely correlated predictors. To determine which variables to keep, the correlation with the numeric response variable PM10 comes in handy: within each group of correlated predictors, the variable that exhibits the highest correlation with the response is selected for retention. The variables chosen to be maintained are, thus, *T_max* and *Wind_speed_med*. Even if selecting the variables between the most correlated has been done by looking at the response variable in the training set and may thus generate too optimistic results during the cross-validation process for the hyperparameter tuning, we'll see that the results obtained during cross-validation will be close to the true error rate.

```
#Removing highly correlated variables
train_data$T_med <- NULL
test_data$T_med <- NULL
train_data$T_min <- NULL
test_data$T_min <- NULL
train_data$Dew_point <- NULL
test_data$Dew_point <- NULL
train_data$Wind_speed_max <- NULL
test_data$Wind_speed_max <- NULL

#Removing the numerical response variable used to select the features to retain
train_data$PM10 <- NULL
test_data$PM10 <- NULL
```

3.2 Outliers

During this phase of the analysis, the attention is directed towards the identification of outliers and the possible strategies to handle them.

```
par(mfrow=c(2,3))
boxplot(train_data$T_max, main = "Box plot Max Temperature")
boxplot(train_data$Humidity, main="Box  plot Humidity")
boxplot(train_data$Visibility, main ="Box plot Visibility")
boxplot(train_data$Wind_speed_med, main= "Box plot Wind_speed_med")
boxplot(train_data$Pressure, main="Box plot Pressure")
par(mfrow=c(1,1))
```



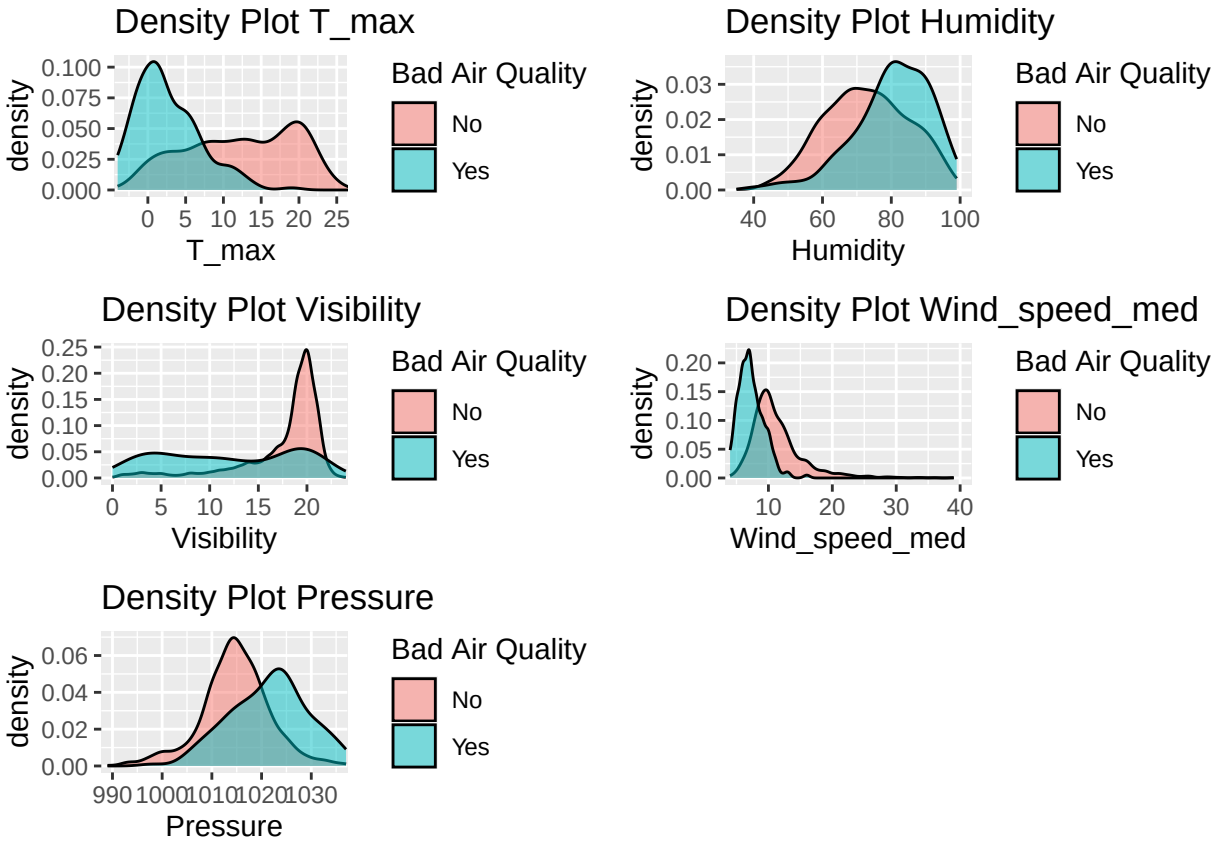
Some of the covariates such as *Pressure*, *Wind_speed_med* and *Visibility* contain outliers. However, these outliers appear to be plausible values and are not indicative of data errors or anomalies. Therefore, there is no need to address or remove these values from the analysis since they can provide useful information for the predictive task.

3.3 Bivariate Density Analysis

In order to identify better which are the most discriminant features that can play a significant role in distinguishing between good and bad air quality, the distribution of each continuous predictor for both cases has been plotted.

```
#Density-plot: Maximum temperature
plot1 <- ggplot(train_data, aes(x = T_max, fill = BadAirQuality)) +
                    geom_density(alpha = 0.5) +
                    coord_cartesian(xlim = c(-3,25)) +
                    labs(title = "Density Plot T_max",
                         x ="T_max") +
                    scale_fill_discrete(name = "Bad Air Quality",labels=c("No","Yes"))
#Density-plot: Humidity
plot2 <-  ggplot(train_data, aes(x = Humidity, fill = BadAirQuality)) +
                    geom_density(alpha = 0.5) +
                    coord_cartesian(xlim = c(35,100)) +
                    labs(title = "Density Plot Humidity", x = "Humidity") +
                    scale_fill_discrete(name = "Bad Air Quality",labels=c("No","Yes"))
#Density-plot: Visibility
plot3 <-  ggplot(train_data, aes(x = Visibility, fill = BadAirQuality)) +
```

```
                    geom_density(alpha = 0.5) +
                    coord_cartesian(xlim = c(0,23)) +
                    labs(title = "Density Plot Visibility", x = "Visibility")+
                    scale_fill_discrete(name = "Bad Air Quality",
                    labels=c("No","Yes"))
#Density-plot: Wind_speed_med
plot4 <-  ggplot(train_data, aes(x = Wind_speed_med, fill = BadAirQuality)) +
                    geom_density(alpha = 0.5) +
                    coord_cartesian(xlim = c(5,40)) +
                    labs(title = "Density Plot Wind_speed_med",
                         x = "Wind_speed_med") +
                    scale_fill_discrete(name = "Bad Air Quality",labels=c("No","Yes"))
#Density-plot: Pressure
plot5 <-  ggplot(train_data, aes(x = Pressure, fill = BadAirQuality)) +
                    geom_density(alpha = 0.5) +
                    coord_cartesian(xlim = c(990,1035)) +
                    labs(title = "Density Plot Pressure", x = "Pressure") +
                    scale_fill_discrete(name = "Bad Air Quality",
                                        labels=c("No","Yes"))
grid.arrange(plot1,plot2, plot3, plot4, plot5, ncol=2, nrow=3)
```



Some variables like *T_max*,*Visibility* and *Wind_speed_med* have a clearly different distribution between cases of good and bad air quality. Specifically, it's possible to notice that warmer days are associated with a good air quality. Furthermore, these days are also characterized by higher visibility values, indicating that objects can be easily discerned at longer distances, and higher wind speeds, which might help in dispersing fine dust particles. Regarding the other variables, it seems that days with high values of PM10 have high

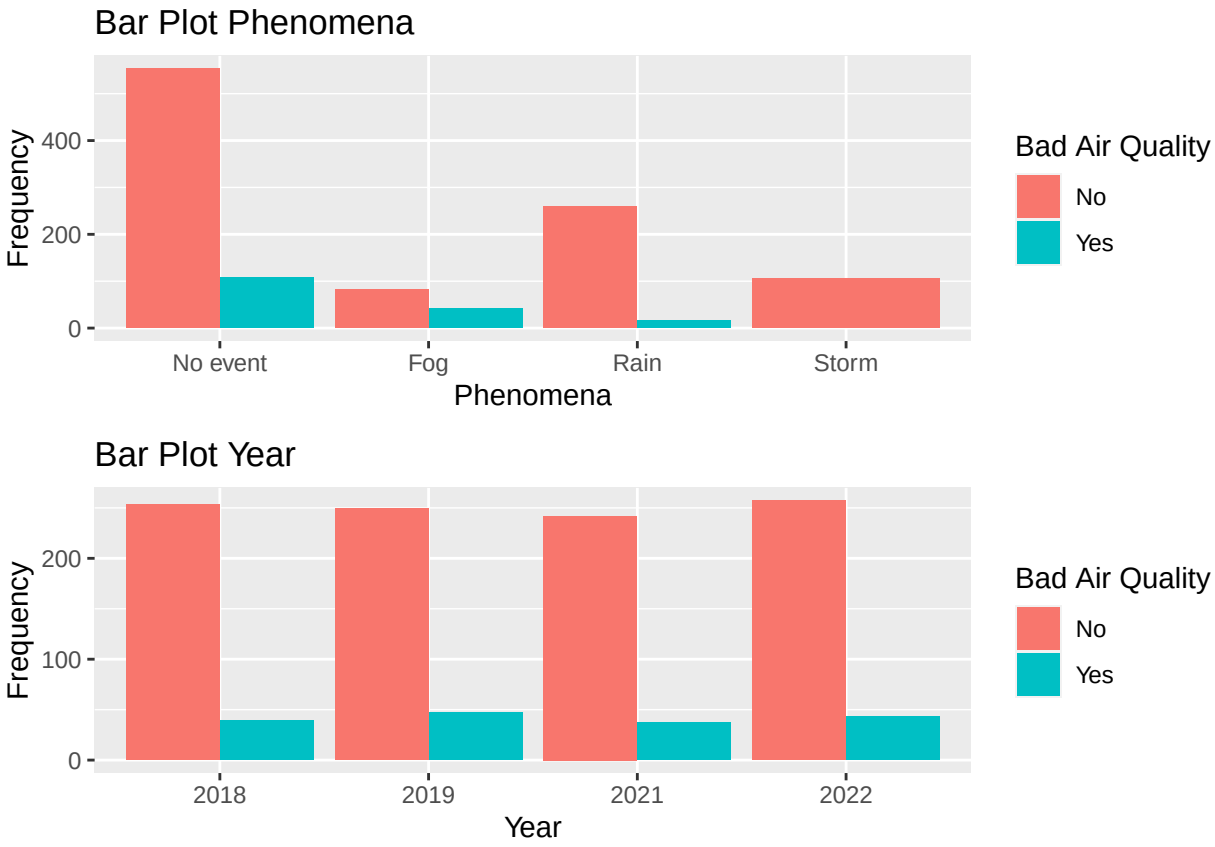
humidity levels. It might be due to the fact that PM10 particles can adhere to water vapor ones, making them heavier, closer to the ground and less prone to be disperse in the atmosphere. Lastly, days with higher atmospheric pressure tend to have a bad air quality. In fact, high-pressure systems are typically associated with stable atmospheric conditions. In these conditions, vertical motion of the air is suppressed and temperature inversions are common: a layer of warm air traps cooler air near the surface and with it also pollutants, leading to the accumulation of PM10. It is important to remark that these observations are interpretations and possible explanations for understanding real-world phenomena. They do not assert any causal relationships between the covariates and air quality, whose causal links should be investigated properly.

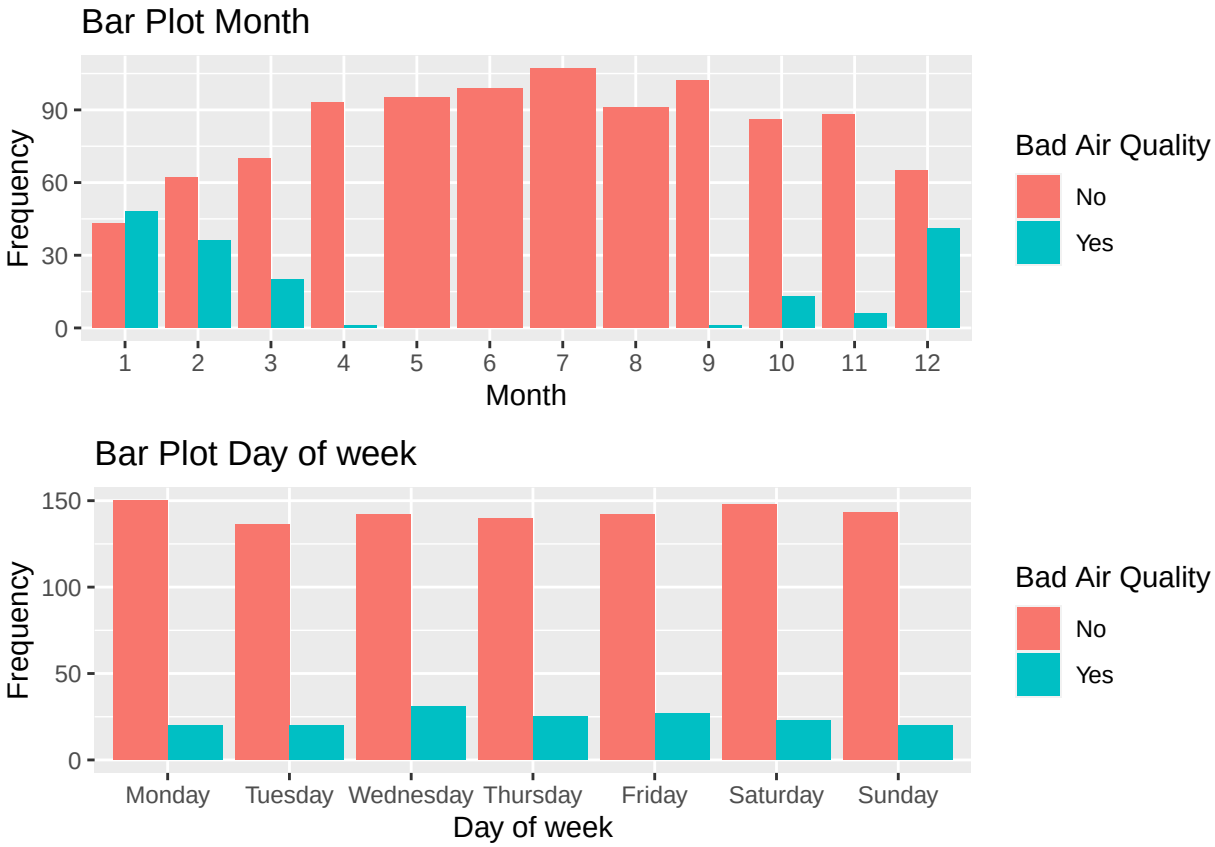
3.4 Bivariate Categorical Analysis

The analysis now continues by exploring the relationships between the categorical variables involved and the response through the examination of the barplots.

```
#Bar plot Phenomena
counts1 <- train_data %>%
  group_by(Phenomena, BadAirQuality) %>%
  summarise(Frequency = n(), .groups="drop")

plot1 <- ggplot(counts1, aes(x = Phenomena,
                             y = Frequency,
                             fill = BadAirQuality)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Bar Plot Phenomena",
       x = "Phenomena",
       y = "Frequency",
       fill = "BadAirQuality") + coord_cartesian(xlim = c(1,4))+
  scale_fill_discrete(name = "Bad Air Quality", labels= c("No","Yes"))
#Bar plot Year
counts2 <- train_data %>%
  group_by(Year, BadAirQuality) %>%
  summarise(Frequency = n(), .groups="drop")

plot2 <- ggplot(counts2, aes(x = Year,
                             y = Frequency,
                             fill = BadAirQuality)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Bar Plot Year",
       x = "Year",
       y = "Frequency",
       fill = "BadAirQuality") + coord_cartesian(xlim = c(1,4))+
  scale_fill_discrete(name = "Bad Air Quality", labels=c("No","Yes")) +
  scale_x_discrete(labels=c("2018","2019","2021","2022"))
#Bar plot Month
counts3 <- train_data %>%
  group_by(Month, BadAirQuality) %>%
  summarise(Frequency = n(), .groups="drop")

plot3 <- ggplot(counts3, aes(x = Month,
                             y = Frequency,
                             fill = BadAirQuality)) +
  geom_bar(stat = "identity", position = "dodge") +
```

```
  labs(title = "Bar Plot Month",
       x = "Month",
       y = "Frequency",
       fill = "BadAirQuality") + coord_cartesian(xlim = c(1,12))+
  scale_fill_discrete(name = "Bad Air Quality", labels=c("No","Yes"))
#Bar plot Day of week
counts4 <- train_data %>%
  group_by(Day_of_week, BadAirQuality) %>%
  summarise(Frequency = n(), .groups="drop")

plot4 <- ggplot(counts4, aes(x = Day_of_week,
                             y = Frequency,
                             fill = BadAirQuality)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Bar Plot Day of week",
       x = "Day of week",
       y = "Frequency",
       fill = "BadAirQuality") + coord_cartesian(xlim = c(1,7))+
  scale_fill_discrete(name = "Bad Air Quality", labels=c("No","Yes")) +
  scale_x_discrete(limits=c("Monday","Tuesday","Wednesday","Thursday",
                            "Friday","Saturday","Sunday"),
                   labels=c("Monday","Tuesday","Wednesday","Thursday",
                            "Friday","Saturday","Sunday"))
```





Let's break down the observations for each covariate individually.

- Regarding the *Phenomena* variable, the data suggest that rainy days are associated with a good air quality. Specifically, the presence of a storm is never linked with high PM10 values. This could be indicative of the effect of the rain that clears the atmosphere of PM10 particles, leading to improved air quality.
- Every year considered tends to have approximately the same number of days with poor air quality.
- The hottest months (May, June, July and August) consistently show low PM10 values. However, it's important to note that this period is associated with another air quality concern: high ozone levels. While PM10 may not be a significant issue during the warmest months, elevated ozone levels can still lead to poor air quality.
- The days with bad air quality are spread out approximately uniformly across the days of the week with a slight peak on Wednesday and lower values on Sunday, Monday and Tuesday. It's essential to remember that the analysis is focused on predicting the air quality for the next day, therefore the frequencies plotted should be interpreted as counts for the following day of the week. The mentioned behavior may be attributed to the fact that weekdays typically have higher industrial and vehicular activities compared to weekends, resulting in smaller emissions of PM10 for the days that follow the weekend (Monday and Tuesday) and higher emissions during the middle of the week (Wednesday).

Since some levels of the factors lead to perfect prediction of the occurrences, the instances of the months May, June, July and August are removed, as well as the ones whose *Phenomena* variable indicates the presence of a storm.

```r
#Removing occurrences of the following months: May,June, July and August
train_data <- train_data[train_data$Month %in% c(1,2,3,4,9,10,11,12),]
test_data <- test_data[test_data$Month %in% c(1,2,3,4,9,10,11,12),]

#Removing Storm occurrences
train_data <- train_data[train_data$Phenomena != "Storm",]
test_data <- test_data[test_data$Phenomena != "Storm",]

#Checking if the proportion of the classes is still respected between training and test set
prop.table(table(train_data$BadAirQuality))
```

```
##
##         0         1
## 0.7765814 0.2234186
```

```r
prop.table(table(test_data$BadAirQuality))
```

```
##
##        0        1
## 0.787234 0.212766
```

```r
#Removal of unuseful levels
train_data$Phenomena <- droplevels(train_data$Phenomena)
train_data$Month <- droplevels(train_data$Month)
test_data$Phenomena <- droplevels(test_data$Phenomena)
test_data$Month <- droplevels(test_data$Month)
```

4 Models

The final training dataset is composed of 743 observations and 10 variables.

```r
dim(train_data)
```

```
## [1] 743  10
```

As previously mentioned, the dataset is quite unbalanced, so to address this issue, the undersampling technique is applied. In the first phase of modeling, the performances of the models trained on the original unbalanced dataset and the balanced one will be compared. This comparative analysis will help determine whether balancing the classes improves the models' ability to predict air quality.

```r
#Dimension of the minority class
min_class_dim <- dim(train_data[train_data$BadAirQuality==1,])[1]
#Train balanced dataset: undersampling
train_balanced<- ovun.sample(BadAirQuality~.,
                             data = train_data,method = "under",
                             N=2*min_class_dim, seed = 123)$data
#Dimension of the undersampled dataset
dim(train_balanced)
```

```
## [1] 332  10
```

```r
#Proportion of the two classes after undersampling
prop.table(table(train_balanced$BadAirQuality))
```

```
##
##   0   1
## 0.5 0.5
```

```r
#Removal of unuseful levels
train_balanced$Phenomena <- droplevels(train_balanced$Phenomena)
train_balanced$Month <- droplevels(train_balanced$Month)
```

In order to have a robust framework for comparing the predictive performances of the models also in presence of the unbalanced dataset in question, several metrics will be employed.

- Accuracy, used to measure the overall correctness of the model.
- Precision, which evaluates the proportion of true positives among all positive predictions.
- Sensitivity, to determine the model's ability to correctly identify all the days with poor air quality. This metrics is particularly important because it shows the extent to which the model is able to detect the days with high PM10 values, thus avoiding health risks associated with exposure of air pollutants to vulnerable people and protecting public health by providing warnings in advance.
- Additionally, the F1-Score will provide a harmonic mean of Precision and Sensitivity, offering a balance between the two in cases where one may be more favorable than the other.

4.1 Logistic Regression: original dataset

The first model analysed is a simple Logistic Regression model, that outputs the probability of poor air quality for the following day. After fitting the model using all the predictors, the presence of multicollinearity between features has been checked using the Variance Inflaction Factor, VIF. While the initial phase of the analysis examined the pairwise collinearity between individual features, VIF is useful for detecting scenarios where a variable might be a linear combination of the other covariates. A value of VIF equal to 1 indicates absence of multicollinearity. Conversely, high VIF values are a signal of strong multicollinearity which can compromise the model's validity increasing the standard errors of the estimated coefficients. Since the dataset contains also categorical predictors with more than two levels, a variation of the Variance Inflaction Factor is considered, that is called Generalized Variance Inflaction Factor (GVIF). While VIF is typically used for models with continuous predictors, GVIF is an extention of VIF to categorical variables. Specifically, for continuous variables the value of GVIF is the same as VIF, while for categorical variables GVIF takes into account the group of dummy variables together rather than individually, resulting in having a single value of GVIF for each categorical predictor. Since GVIF value can be high simply due to the high number of levels a categorical variable have and not because of collinearity, the GVIF should be adjusted by the number of degrees of freedom involved. As a result, for categorical variables, the measure to check for multicollinearity is (GVIF^(1/(2*Df)), where Df is the number of degrees of freedom.

```r
#Fitting a logistic regression model
glm.fit <- glm(BadAirQuality~., data=train_data,
               family="binomial")
summary(glm.fit)
```

```
##
## Call:
```

```
## glm(formula = BadAirQuality ~ ., family = "binomial", data = train_data)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -54.926273  19.182080  -2.863 0.004191 **
## T_max                 -0.110581   0.048534  -2.278 0.022701 *
## Humidity              -0.003712   0.018366  -0.202 0.839816
## Visibility            -0.118340   0.033058  -3.580 0.000344 ***
## Wind_speed_med        -0.435059   0.068592  -6.343 2.26e-10 ***
## Pressure               0.061544   0.018290   3.365 0.000766 ***
## PhenomenaFog          -1.490783   0.415699  -3.586 0.000336 ***
## PhenomenaRain         -1.537547   0.469977  -3.272 0.001070 **
## Year19                -0.041464   0.356181  -0.116 0.907327
## Year21                -0.784544   0.380214  -2.063 0.039072 *
## Year22                -0.986382   0.381714  -2.584 0.009764 **
## Month2                -1.135136   0.444772  -2.552 0.010705 *
## Month3                -1.057284   0.457035  -2.313 0.020703 *
## Month4                -3.251543   1.155153  -2.815 0.004881 **
## Month9                -3.295459   1.297510  -2.540 0.011091 *
## Month10               -1.718629   0.686644  -2.503 0.012317 *
## Month11               -2.404697   0.661473  -3.635 0.000278 ***
## Month12               -1.258106   0.416290  -3.022 0.002510 **
## Day_of_weekMonday     -0.513213   0.472132  -1.087 0.277032
## Day_of_weekSaturday   -0.410256   0.468269  -0.876 0.380969
## Day_of_weekSunday     -0.731669   0.472091  -1.550 0.121178
## Day_of_weekThursday   -0.244982   0.448161  -0.547 0.584627
## Day_of_weekTuesday    -0.598415   0.468875  -1.276 0.201858
## Day_of_weekWednesday  -0.122717   0.438696  -0.280 0.779684
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 789.36  on 742  degrees of freedom
## Residual deviance: 422.93  on 719  degrees of freedom
## AIC: 470.93
##
## Number of Fisher Scoring iterations: 7
```

```
#Checking the VIF for all the predictors
vif(glm.fit)
```

```
##                    GVIF Df GVIF^(1/(2*Df))
## T_max          3.290769  1        1.814048
## Humidity       3.579425  1        1.891937
## Visibility     3.266605  1        1.807375
## Wind_speed_med 1.555637  1        1.247252
## Pressure       1.472572  1        1.213496
## Phenomena      2.623416  2        1.272673
## Year           1.364436  3        1.053155
## Month          5.168010  7        1.124480
## Day_of_week    1.200242  6        1.015327
```

From the GVIF values, it's evident that the numerical covariates, especially *T_max*, *Humidity* and *Visibility*

present a moderate multicollinearity. Even if multicollinearity is present, for the moment the problem will not be addressed, since their VIF values are above 1 but below the common threshold of 5. Regarding the categorical variables, on the other hand, some GVIF values are high, but the adjusted values which take into account the number of degrees of freedom are all close to 1, indicating lack of collinearity. Thus, no further adjustment is required.

4.1.1 Logistic Regression on original dataset: feature selection

To select the best model, backward stepwise selection will be applied employing the Bayesian Information Criterion (BIC) as a criterion to minimize. The BIC allows the comparison of models with a different number of predictors since it includes a penalty term on the complexity of the model proportional to the count of predictors utilized. This penalty is weighted by the logarithm of the sample size (log(n)), which tends to favour simpler models as the sample size grows, compared to the Akaike Information Criterion (AIC) which imposes a less stringent penalty. The choice of opting for BIC over AIC in the analysis is double: above improving the model's predictive capability by mitigating overfitting, having fewer coefficients simplifies interpretability.

```
#Training set dimension
n <- dim(train_data)[1]
#Backward stepwise selection with BIC as criterion
mod.R <- step(glm.fit,trace=1,k=log(n), direction="backward")
```

```
## Start:  AIC=581.59
## BadAirQuality ~ T_max + Humidity + Visibility + Wind_speed_med +
##     Pressure + Phenomena + Year + Month + Day_of_week
##
##                  Df Deviance    AIC
## - Day_of_week     6   426.85 545.84
## - Month           7   445.64 558.02
## - Year            3   434.15 572.97
## - Humidity        1   422.97 575.02
## - T_max           1   428.15 580.19
## <none>                422.93 581.59
## - Phenomena       2   441.07 586.50
## - Pressure        1   434.94 586.99
## - Visibility      1   436.57 588.61
## - Wind_speed_med  1   482.63 634.67
##
## Step:  AIC=545.84
## BadAirQuality ~ T_max + Humidity + Visibility + Wind_speed_med +
##     Pressure + Phenomena + Year + Month
##
##                  Df Deviance    AIC
## - Month           7   449.19 521.91
## - Year            3   437.47 536.63
## - Humidity        1   426.87 539.25
## - T_max           1   431.73 544.11
## <none>                426.85 545.84
## - Pressure        1   438.46 550.84
## - Phenomena       2   445.33 551.10
## - Visibility      1   440.55 552.94
## - Wind_speed_med  1   485.95 598.33
##
```

```
## Step:  AIC=521.91
## BadAirQuality ~ T_max + Humidity + Visibility + Wind_speed_med +
##     Pressure + Phenomena + Year
##
##                  Df Deviance    AIC
## - Year            3   460.52 513.41
## - Humidity        1   449.21 515.32
## <none>                449.19 521.91
## - Pressure        1   460.35 526.46
## - Phenomena       2   470.42 529.92
## - Visibility      1   469.22 535.33
## - Wind_speed_med  1   504.47 570.57
## - T_max           1   541.27 607.38
##
## Step:  AIC=513.41
## BadAirQuality ~ T_max + Humidity + Visibility + Wind_speed_med +
##     Pressure + Phenomena
##
##                  Df Deviance    AIC
## - Humidity        1   460.67 506.94
## <none>                460.52 513.41
## - Pressure        1   469.63 515.90
## - Phenomena       2   482.05 521.72
## - Visibility      1   480.68 526.96
## - Wind_speed_med  1   509.99 556.27
## - T_max           1   548.77 595.04
##
## Step:  AIC=506.94
## BadAirQuality ~ T_max + Visibility + Wind_speed_med + Pressure +
##     Phenomena
##
##                  Df Deviance    AIC
## <none>                460.67 506.94
## - Pressure        1   469.99 509.66
## - Phenomena       2   482.73 515.79
## - Visibility      1   491.85 531.51
## - Wind_speed_med  1   519.68 559.35
## - T_max           1   549.06 588.73
```

```
#Summary of the selected model
summary(mod.R)
```

```
##
## Call:
## glm(formula = BadAirQuality ~ T_max + Visibility + Wind_speed_med +
##     Pressure + Phenomena, family = "binomial", data = train_data)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -41.93796   15.88063  -2.641 0.008270 **
## T_max           -0.21235    0.02662  -7.977 1.50e-15 ***
## Visibility      -0.13698    0.02598  -5.273 1.34e-07 ***
## Wind_speed_med  -0.36897    0.05822  -6.337 2.34e-10 ***
## Pressure         0.04662    0.01551   3.005 0.002654 **
```

```
## PhenomenaFog    -1.43224    0.38665  -3.704 0.000212 ***
## PhenomenaRain   -1.66530    0.44178  -3.770 0.000164 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 789.36  on 742  degrees of freedom
## Residual deviance: 460.67  on 736  degrees of freedom
## AIC: 474.67
##
## Number of Fisher Scoring iterations: 6
```

The selected predictors by the stepwise process are: *T_max*, *Visibility*, *Wind_speed_med*, *Pressure* and *Phenomena*. From the estimates of the coeffiecients, it is shown clearly the role of each variable in predicting air quality. As already suggested by their distributions, given that all the other variables are fixed:

- an increase of one unit in *T_max* decreases the odds ratio of having poor air quality the following day of $e^{-0.212} = 0.809$ times;
- an increase of one unit in *Wind_speed_med* affects negatively the probability of having bad air quality the next day, reducing the odds ratio of $e^{-0.369} = 0.691$ times;
- an increase of one unit in *Visibility* is negative related with high concentration of PM10 values in the air, decreasing the odds ratio of $e^{-0.137} = 0.872$ times;
- conversely, an increase of one unit in *Pressure* raises the odds ratio of having poor air quality in the next day of $e^{0.047} = 1.047$ times;

Concerning the categorical variable *Phenomena*, holding all the other variables constant:

- the odds of having poor air quality the next day is about 76% lower in presence of fog compared with the reference level *No event*, given an odds ratio of $e^{-1.432} = 0.238$
- the odds of having bad air quality next day is 81% lower if it rains compared with no event happening given an odds ratio of $e^{-1.665} = 0.189$.

At first sight, it seems controintuitive that the presence of fog decreases the probability of having bad air quality the following day, but a possible explanation for this phenomenon is that when fog is present the PM10 particles can adhere to water vapor droplets, becoming heavier, falling to the ground purifying the air from pollutants.

```
#Extracting the response variable of the test dataset
y.test <- test_data$BadAirQuality
test_data$BadAirQuality <- NULL
#Predicted probabilities
glm.pred <- predict(mod.R,newdata=test_data,
                    type="response")
#Construting the predictions based on the output probabilities
logistic.pred <- rep(0,length(y.test))
logistic.pred[glm.pred>0.5] <- 1
#Confusion matrix
conf.matrix <- table(logistic.pred, y.test)
#Dataframe of the confusion matrix for visualization
```

```r
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")
#Filling the dataframe with the confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of Logistic Regression on the original dataset. Threshold:0.5") %>%
  kable_styling(latex_options = "hold_position") %>% column_spec(c(1,3), border_right = TRUE)
```

Table 2: Confusion Matrix of Logistic Regression on the original dataset. Threshold:0.5

	True Good Air	True Bad Air	Total
Pred. Good Air	137	17	154
Pred. Bad Air	11	23	34
Total	148	40	188

```r
#Accuracy
acc_lr <- mean(logistic.pred==y.test)
cat("Accuracy: ", acc_lr, "\n")
```

```
## Accuracy:  0.8510638
```

```r
#Precision
prec_lr <- conf.matrix[2,2]/sum(conf.matrix[2,])
cat("Precision: ", prec_lr, "\n")
```

```
## Precision:  0.6764706
```

```r
#Recall
recall_lr <- conf.matrix[2,2]/sum(conf.matrix[,2])
cat("Recall: ", recall_lr, "\n")
```

```
## Recall:  0.575
```

```r
#F1-Score
F1_score <- 2*(prec_lr*recall_lr)/(prec_lr + recall_lr)
cat("F1-score: ", F1_score, "\n")
```

```
## F1-score:  0.6216216
```

The presented results were derived without the optimization of the decision threshold. The observed low recall suggests a need to experiment with lower threshold values, specifically, 0.3 and 0.4. The objective of this step is to enhance the model's sensitivity to detect days with poor air quality, without compromising the general predictive accuracy of the model and its precision.

As the following tables show, the best overall threshold that balances the need to identify as many days with poor air quality as possible without a proportional increase in false positives is 0.3.

Table 3: Confusion Matrix of Logistic Regression on the original dataset. Threshold:0.4

	True Good Air	True Bad Air	Total
Pred. Good Air	134	12	146
Pred. Bad Air	14	28	42
Total	148	40	188

Table 4: Confusion Matrix of Logistic Regression on the original dataset. Threshold:0.3

	True Good Air	True Bad Air	Total
Pred. Good Air	129	8	137
Pred. Bad Air	19	32	51
Total	148	40	188

Table 5: Metrics for different thresholds: original dataset

	0.3	0.4	0.5
Accuracy	0.8563830	0.8617021	0.8510638
Precision	0.6274510	0.6666667	0.6764706
Recall	0.8000000	0.7000000	0.5750000
F1-score	0.7032967	0.6829268	0.6216216

4.2 Logistic Regression: undersampled dataset and stepwise selection

Let's now fit the logistic regression model on the balanced dataset and apply again the backward stepwise selection.

```
#Full logistic regression model on the undersampled dataset
glm.fit_under <- glm(BadAirQuality~., data=train_balanced, family="binomial")
summary(glm.fit_under)
```

```
##
## Call:
## glm(formula = BadAirQuality ~ ., family = "binomial", data = train_balanced)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -93.022919  29.686922  -3.133  0.00173 **
## T_max                -0.096675   0.063036  -1.534  0.12512
## Humidity              0.008016   0.025818   0.310  0.75620
## Visibility           -0.135575   0.045334  -2.991  0.00278 **
## Wind_speed_med       -0.351847   0.084836  -4.147 3.36e-05 ***
## Pressure              0.099405   0.028377   3.503  0.00046 ***
## PhenomenaFog         -1.809909   0.565903  -3.198  0.00138 **
## PhenomenaRain        -1.516309   0.596743  -2.541  0.01105 *
## Year19               -0.016071   0.485246  -0.033  0.97358
## Year21               -0.764980   0.523808  -1.460  0.14417
## Year22               -1.039600   0.530500  -1.960  0.05004 .
## Month2               -1.965364   0.675681  -2.909  0.00363 **
```

```
## Month3                -1.808857   0.676686  -2.673  0.00752 **
## Month4                -3.972410   1.312846  -3.026  0.00248 **
## Month9                -4.437371   1.519474  -2.920  0.00350 **
## Month10               -2.233402   0.942512  -2.370  0.01781 *
## Month11               -2.760531   0.906379  -3.046  0.00232 **
## Month12               -1.715673   0.653709  -2.625  0.00868 **
## Day_of_weekMonday     -0.862317   0.654522  -1.317  0.18768
## Day_of_weekSaturday   -0.824592   0.631488  -1.306  0.19162
## Day_of_weekSunday     -0.650368   0.640732  -1.015  0.31009
## Day_of_weekThursday   -0.499025   0.588769  -0.848  0.39668
## Day_of_weekTuesday    -0.392014   0.685952  -0.571  0.56767
## Day_of_weekWednesday  -0.055691   0.617079  -0.090  0.92809
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 460.25  on 331  degrees of freedom
## Residual deviance: 234.10  on 308  degrees of freedom
## AIC: 282.1
##
## Number of Fisher Scoring iterations: 6
```

```r
#Backward stepwise selection on the undersampled dataset
#dimension of the undersampled set
n_under <- dim(train_balanced)[1]
#Backward stepwise selection with BIC as criterion
mod.R_under <- step(glm.fit_under,trace=0,k=log(n_under), direction="backward")
#Summary of the selected model
summary(mod.R_under)
```

```
##
## Call:
## glm(formula = BadAirQuality ~ T_max + Visibility + Wind_speed_med +
##     Pressure + Phenomena, family = "binomial", data = train_balanced)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -57.46647   21.54480  -2.667  0.00765 **
## T_max           -0.21705    0.03326  -6.526 6.75e-11 ***
## Visibility      -0.14756    0.03438  -4.292 1.77e-05 ***
## Wind_speed_med  -0.31584    0.07060  -4.474 7.68e-06 ***
## Pressure         0.06273    0.02112   2.970  0.00298 **
## PhenomenaFog    -1.48948    0.50685  -2.939  0.00330 **
## PhenomenaRain   -1.52011    0.54933  -2.767  0.00565 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 460.25  on 331  degrees of freedom
## Residual deviance: 263.00  on 325  degrees of freedom
## AIC: 277
##
```

```
## Number of Fisher Scoring iterations: 6
```

The backward stepwise selection for the logistic regression model on the balanced dataset has selected the same covariates obtained without the balacing of the classes, namely *T_max*, *Visibility*, *Wind_speed_med*, *Pressure* and *Phenomena*.

```
#Construting the predictions based on the output probabilities
pred_under <- predict(mod.R_under,newdata=test_data,
                      type="response")
logistic.pred_u <- rep(0,length(y.test))
logistic.pred_u[pred_under>0.6] <- 1
#Confusion matrix
conf.matrix_u <- table(logistic.pred_u, y.test)
#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")
#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_u
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of Logistic Regression on the undersampled dataset. Threshold:0.6") %>%
  kable_styling(latex_options = "hold_position") %>% column_spec(c(1,3), border_right = TRUE)
```

Table 6: Confusion Matrix of Logistic Regression on the undersampled dataset. Threshold:0.6

	True Good Air	True Bad Air	Total
Pred. Good Air	130	8	138
Pred. Bad Air	18	32	50
Total	148	40	188

```
#Accuracy
acc_lr_u <- mean(logistic.pred_u==y.test)
cat("Accuracy: ", acc_lr_u, "\n")
```

```
## Accuracy:  0.8617021
```

```
#Precision
prec_lr_u <- conf.matrix_u[2,2]/sum(conf.matrix_u[2,])
cat("Precision: ", prec_lr_u, "\n")
```

```
## Precision:  0.64
```

```
#Recall
recall_lr_u <- conf.matrix_u[2,2]/sum(conf.matrix_u[,2])
cat("Recall: ", recall_lr_u, "\n")
```

```
## Recall:  0.8
```

```
#F1-Score
F1_score_u <- 2*(prec_lr_u*recall_lr_u)/(prec_lr_u + recall_lr_u)
cat("F1-score: ", F1_score_u, "\n")
```

```
## F1-score:  0.7111111
```

The models trained on the balanced dataset successfully achieves the objective of decreasing the rate of false negatives. However, this improvement comes at a cost of increasing the incidence of false positives. Also in this case different thresholds have been experimented and it results that the best overall model is the one with the threshold equal to 0.6.

```
#Constructing the predictions based on the output probabilities
logistic.pred_u4 <- rep(0,length(y.test))
logistic.pred_u4[pred_under>0.4] <- 1
#Confusion matrix
conf.matrix_u4 <- table(logistic.pred_u4, y.test)


#Constructing the predictions based on the output probabilities
logistic.pred_u5 <- rep(0,length(y.test))
logistic.pred_u5[pred_under>0.5] <- 1
#Confusion matrix
conf.matrix_u5 <- table(logistic.pred_u5, y.test)

#Accuracy
acc_lr_u4 <- mean(logistic.pred_u4==y.test)

#Precision
prec_lr_u4 <- conf.matrix_u4[2,2]/sum(conf.matrix_u4[2,])

#Recall
recall_lr_u4 <- conf.matrix_u4[2,2]/sum(conf.matrix_u4[,2])

#F1-Score
F1_score_u4 <- 2*(prec_lr_u4*recall_lr_u4)/(prec_lr_u4 + recall_lr_u4)


#Accuracy
acc_lr_u5 <- mean(logistic.pred_u5==y.test)

#Precision
prec_lr_u5 <- conf.matrix_u5[2,2]/sum(conf.matrix_u5[2,])

#Recall
recall_lr_u5 <- conf.matrix_u5[2,2]/sum(conf.matrix_u5[,2])

#F1-Score
F1_score_u5 <- 2*(prec_lr_u5*recall_lr_u5)/(prec_lr_u5 + recall_lr_u5)


df.metrics <- as.data.frame(matrix(0,ncol=3,nrow=4))
```

```
colnames(df.metrics) <- c("0.4","0.5","0.6")
rownames(df.metrics) <- c("Accuracy","Precision","Recall","F1-score")
df.metrics[,"0.4"] <- c(acc_lr_u4,prec_lr_u4,recall_lr_u4,F1_score_u4)
df.metrics[,"0.5"] <- c(acc_lr_u5,prec_lr_u5,recall_lr_u5,F1_score_u5)
df.metrics[,"0.6"] <- c(acc_lr_u,prec_lr_u,recall_lr_u,F1_score_u)

kable(df.metrics, format = "latex", booktabs = TRUE,
caption = "Metrics for different thresholds: undersampled dataset") %>%
  kable_styling(latex_options = "hold_position") %>% column_spec(c(1,3), border_right = TRUE)
```

Table 7: Metrics for different thresholds: undersampled dataset

	0.4	0.5	0.6
Accuracy	0.8085106	0.8244681	0.8617021
Precision	0.5303030	0.5593220	0.6400000
Recall	0.8750000	0.8250000	0.8000000
F1-score	0.6603774	0.6666667	0.7111111

4.3 Interaction effects

To have a deeper understanding of how the relationships between covariates influence the predictions, interaction terms are included in the Logistic Regression model. The aim is to capture the combined effects of the variables that are not evident when considering the predictors independently. All the variables selected by the backwise stepwise procedure are considered to fit the model as primary predictors and all the possible second order interaction terms between them. Regarding the continuous variables, also third and fourth order interactions are included. It's important to highlight that the interaction terms between the categorical variable *Phenomena* and the continuous variables help to determine whether the relationship between each continuous variable and the response changes across the different levels of the categorical variable (*No event*, *Fog* and *Rain*). After fitting the full model on the original training dataset, backward stepwise selection is again performed to find the most significant predictors. The metric chosen to minimize this time is the Akaike Information Criterion (AIC).

```
#Logistic regression with interaction effects
glm.fit.int <- glm(BadAirQuality~Visibility+Pressure+Wind_speed_med+T_max+Phenomena+
                     Visibility:Pressure+Visibility:Wind_speed_med+
                     Pressure:Wind_speed_med+Visibility:T_max+
                     Pressure:T_max+Wind_speed_med:T_max+
                     Visibility:Pressure:Wind_speed_med+
                     Visibility:Pressure:T_max+Visibility:Wind_speed_med:T_max+
                     Pressure:Wind_speed_med:T_max+
                     Visibility:Pressure:Wind_speed_med:T_max+
                     Phenomena:T_max+Phenomena:Wind_speed_med+
                     Phenomena:Pressure+ Phenomena:Visibility, data=train_data,
                   family="binomial")
summary(glm.fit.int)
```

```
##
## Call:
## glm(formula = BadAirQuality ~ Visibility + Pressure + Wind_speed_med +
##     T_max + Phenomena + Visibility:Pressure + Visibility:Wind_speed_med +
```

```
##     Pressure:Wind_speed_med + Visibility:T_max + Pressure:T_max +
##     Wind_speed_med:T_max + Visibility:Pressure:Wind_speed_med +
##     Visibility:Pressure:T_max + Visibility:Wind_speed_med:T_max +
##     Pressure:Wind_speed_med:T_max + Visibility:Pressure:Wind_speed_med:T_max +
##     Phenomena:T_max + Phenomena:Wind_speed_med + Phenomena:Pressure +
##     Phenomena:Visibility, family = "binomial", data = train_data)
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                             3.556e+02  2.320e+02   1.532   0.1254
## Visibility                             -2.934e+01  1.376e+01  -2.132   0.0330
## Pressure                               -3.459e-01  2.280e-01  -1.517   0.1293
## Wind_speed_med                         -3.855e+01  2.745e+01  -1.405   0.1602
## T_max                                  -5.967e+01  3.724e+01  -1.602   0.1091
## PhenomenaFog                           -1.604e+01  6.068e+01  -0.264   0.7915
## PhenomenaRain                          -1.254e+01  6.288e+01  -0.199   0.8419
## Visibility:Pressure                     2.881e-02  1.352e-02   2.131   0.0331
## Visibility:Wind_speed_med               2.910e+00  1.596e+00   1.824   0.0682
## Pressure:Wind_speed_med                 3.783e-02  2.703e-02   1.400   0.1616
## Visibility:T_max                        2.835e+00  2.644e+00   1.072   0.2836
## Pressure:T_max                          5.858e-02  3.658e-02   1.601   0.1093
## Wind_speed_med:T_max                    5.880e+00  4.477e+00   1.313   0.1891
## T_max:PhenomenaFog                     -6.560e-02  9.768e-02  -0.672   0.5019
## T_max:PhenomenaRain                     5.088e-02  1.069e-01   0.476   0.6340
## Wind_speed_med:PhenomenaFog            -3.337e-01  2.474e-01  -1.349   0.1773
## Wind_speed_med:PhenomenaRain            3.314e-02  1.770e-01   0.187   0.8515
## Pressure:PhenomenaFog                   1.586e-02  5.917e-02   0.268   0.7887
## Pressure:PhenomenaRain                  9.394e-03  6.157e-02   0.153   0.8787
## Visibility:PhenomenaFog                 1.200e-01  6.994e-02   1.716   0.0862
## Visibility:PhenomenaRain                6.768e-02  7.221e-02   0.937   0.3486
## Visibility:Pressure:Wind_speed_med      -2.879e-03  1.571e-03  -1.832   0.0669
## Visibility:Pressure:T_max               -2.802e-03  2.595e-03  -1.080   0.2801
## Visibility:Wind_speed_med:T_max         -3.385e-01  3.029e-01  -1.117   0.2638
## Pressure:Wind_speed_med:T_max           -5.790e-03  4.405e-03  -1.314   0.1888
## Visibility:Pressure:Wind_speed_med:T_max  3.337e-04  2.977e-04   1.121   0.2624
##
## (Intercept)
## Visibility                             *
## Pressure
## Wind_speed_med
## T_max
## PhenomenaFog
## PhenomenaRain
## Visibility:Pressure                    *
## Visibility:Wind_speed_med              .
## Pressure:Wind_speed_med
## Visibility:T_max
## Pressure:T_max
## Wind_speed_med:T_max
## T_max:PhenomenaFog
## T_max:PhenomenaRain
## Wind_speed_med:PhenomenaFog
## Wind_speed_med:PhenomenaRain
## Pressure:PhenomenaFog
```

```
## Pressure:PhenomenaRain
## Visibility:PhenomenaFog                  .
## Visibility:PhenomenaRain
## Visibility:Pressure:Wind_speed_med       .
## Visibility:Pressure:T_max
## Visibility:Wind_speed_med:T_max
## Pressure:Wind_speed_med:T_max
## Visibility:Pressure:Wind_speed_med:T_max
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 789.36  on 742  degrees of freedom
## Residual deviance: 435.22  on 717  degrees of freedom
## AIC: 487.22
##
## Number of Fisher Scoring iterations: 8
```

```
#Stepwise Backward selection based on AIC
mod.R_int <- step(glm.fit.int,direction="backward",trace=0)
summary(mod.R_int)
```

```
##
## Call:
## glm(formula = BadAirQuality ~ Visibility + Pressure + Wind_speed_med +
##     T_max + Phenomena + Visibility:Pressure + Pressure:Wind_speed_med +
##     Visibility:T_max + Pressure:T_max, family = "binomial", data = train_data)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -56.968082  61.074488  -0.933 0.350942
## Visibility               -5.382306   2.395798  -2.247 0.024668 *
## Pressure                  0.060897   0.059924   1.016 0.309516
## Wind_speed_med           12.029382   6.301239   1.909 0.056256 .
## T_max                    -9.420771   4.466200  -2.109 0.034915 *
## PhenomenaFog             -1.328282   0.390655  -3.400 0.000674 ***
## PhenomenaRain            -1.667566   0.460543  -3.621 0.000294 ***
## Visibility:Pressure       0.005184   0.002351   2.205 0.027442 *
## Pressure:Wind_speed_med  -0.012184   0.006198  -1.966 0.049343 *
## Visibility:T_max         -0.008811   0.004097  -2.151 0.031497 *
## Pressure:T_max            0.009145   0.004381   2.088 0.036840 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 789.36  on 742  degrees of freedom
## Residual deviance: 445.90  on 732  degrees of freedom
## AIC: 467.9
##
## Number of Fisher Scoring iterations: 7
```

With interaction terms, the coefficients of the main effects are no longer interpreted as the effects on the

outcome for a one-unit change in the predictor when all other variables are held constant. Instead, they represent the effect when the interacting variables are held at zero, which may not be a meaningful condition, especially for some variables like *Pressure* whose range does not include the zero value. When considering interactions, for the main effects, the impact on the probability of having bad air quality depends on the values of the other variables with they interact with that can have a synergetic or antagonist effect. The significant terms kept by the backward stepwise selection are *Visibility*, *T_max*, *Wind_speed_med*, *PhenomenaRain*, *PhenomenaFog* and the interaction terms *Visibility:Pressure*, *Visibility:T_max*, *Pressure:Wind_speed_med* and *Pressure:T_max*. From the summary of the model, it's evident that for the hierarchical principle since the interaction terms are included in the model, also the main effects are included even if their p-value are not significant. This means that *Pressure* does not have a significant independent effect on the response, but its effect becomes significant when *Visibility*, *T_max* and *Wind_speed_med* variables are taken into account. Let's analyze the significant interaction terms:

- Visibility:T_max. The negative coefficient shows that the effect of *T_max* on the log-odds of having bad air quality decreases as *Visibility* increases. In other words, the presence of higher visibility levels moderates the impact that *T_max* has on the likelihood of bad air quality.
- Visibility:Pressure. From the positive coefficient is clear that higher visibility values increase the effect of pressure on the likelihood of bad air quality.
- Pressure:Wind_speed_med. This interaction term has a negative coefficient, indicating that the effect of pressure on the log-odds of bad air quality decreases as wind speed is higher.
- Pressure:T_max. The positive coefficient shows that *T_max* has a synergetic effect for the *Pressure* on the log-odds of bad air quality.

```
#Predictions
glm.pred.int <- predict(mod.R_int, newdata=test_data, type="response")
logistic.pred_int <- rep(0,length(y.test))
#Different thresholds tested:0.3,0.4,0.5
logistic.pred_int[glm.pred.int > 0.3] <- 1

#Confusion matrix
conf.matrix_int <- table(as.factor(logistic.pred_int), y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_int
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption =
  "Confusion Matrix of Logistic Regression model with interaction terms
on the original training dataset. Threshold: 0.3") %>%
  kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 8: Confusion Matrix of Logistic Regression model with interaction terms on the original training dataset. Threshold: 0.3

	True Good Air	True Bad Air	Total
Pred. Good Air	128	8	136
Pred. Bad Air	20	32	52
Total	148	40	188

```
#Accuracy
acc_int <- mean(logistic.pred_int==y.test)
cat("Accuracy: ", acc_int, "\n")
```

```
## Accuracy:  0.8510638
```

```
#Precision
prec_int <- conf.matrix_int[2,2]/sum(conf.matrix_int[2,])
cat("Precision: ", prec_int, "\n")
```

```
## Precision:  0.6153846
```

```
#Recall
recall_int <- conf.matrix_int[2,2]/sum(conf.matrix_int[,2])
cat("Recall: ", recall_int, "\n")
```

```
## Recall:  0.8
```

```
#F1-Score
F1_score_int <- 2*(prec_int*recall_int)/(prec_int + recall_int)
cat("F1-score: ", F1_score_int, "\n")
```

```
## F1-score:  0.6956522
```

Interaction effects on the undersampled dataset

```
#Interaction effects on the balanced training set
glm.fit.int_u <- glm(BadAirQuality~Visibility+Pressure+Wind_speed_med+T_max+Phenomena+
                     Visibility:Pressure+Visibility:Wind_speed_med+
                     Pressure:Wind_speed_med+Visibility:T_max+
                     Pressure:T_max+Wind_speed_med:T_max+
                     Visibility:Pressure:Wind_speed_med+
                     Visibility:Pressure:T_max+Visibility:Wind_speed_med:T_max+
                     Pressure:Wind_speed_med:T_max+
                     Visibility:Pressure:Wind_speed_med:T_max+
                     Phenomena:T_max+Phenomena:Wind_speed_med+
                     Phenomena:Pressure+ Phenomena:Visibility,
                     data=train_balanced, family="binomial")

#Stepwise Backward selection based on AIC
mod.R_int_u <- step(glm.fit.int_u,direction="backward", trace=0)
summary(mod.R_int_u)
```

```
##
## Call:
## glm(formula = BadAirQuality ~ Visibility + Pressure + Wind_speed_med +
##     T_max + Phenomena + Visibility:Pressure + Visibility:Wind_speed_med +
##     Pressure:Wind_speed_med + Visibility:T_max + Pressure:T_max +
##     Visibility:Pressure:Wind_speed_med, family = "binomial",
##     data = train_balanced)
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     236.852331 176.762778   1.340  0.18026
## Visibility                      -25.793239  11.344209  -2.274  0.02298 *
## Pressure                         -0.228198   0.173591  -1.315  0.18865
## Wind_speed_med                  -27.334825  21.607728  -1.265  0.20585
## T_max                            -9.369122   6.136833  -1.527  0.12683
## PhenomenaFog                     -1.352649   0.525970  -2.572  0.01012 *
## PhenomenaRain                    -1.559590   0.563537  -2.768  0.00565 **
## Visibility:Pressure               0.025306   0.011144   2.271  0.02316 *
## Visibility:Wind_speed_med         2.482656   1.307560   1.899  0.05760 .
## Pressure:Wind_speed_med           0.026679   0.021275   1.254  0.20984
## Visibility:T_max                 -0.016235   0.006100  -2.661  0.00778 **
## Pressure:T_max                    0.009204   0.006030   1.526  0.12696
## Visibility:Pressure:Wind_speed_med -0.002447   0.001287  -1.901  0.05729 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 460.25  on 331  degrees of freedom
## Residual deviance: 247.85  on 319  degrees of freedom
## AIC: 273.85
##
## Number of Fisher Scoring iterations: 6
```

On the undersampled dataset, the significant covariates retained by the backward stepwise selection are *Visibility*, *PhenomenaRain*, *PhenomenaFog* and the interaction terms *Visibility:Pressure*, *Visibility:Wind_speed_med*, *Visibility:T_max* and *Visibility:Pressure:Wind_speed_med*. In this case, also a third-order interaction term is included. It shows a more complex relationship between these three variables: as *Wind_speed_med* increases, it changes the combined effect of *Visibility* and *Pressure* on the log-odds of bad air quality. The negative sign indicates that the combined effect of visibility and pressure on bad air quality is reduced with higher wind speed. In fact, visibility and pressure's impact may be reduced on air quality since higher wind speed disperse pollutants.

```r
#Predictions
glm.pred.int_u <- predict(mod.R_int_u, newdata=test_data, type="response")
logistic.pred_int_u <- rep(0,length(y.test))
#Different thresholds tested:0.3,0.4,0.5
logistic.pred_int_u[glm.pred.int_u > 0.4] <- 1

#Confusion matrix
conf.matrix_int_u <- table(as.factor(logistic.pred_int_u), y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
```

```r
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_int_u
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of Logistic Regression model with
interaction terms on the undersampled training dataset. Threshold: 0.4") %>%
  kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 9: Confusion Matrix of Logistic Regression model with interaction terms on the undersampled training dataset. Threshold: 0.4

	True Good Air	True Bad Air	Total
Pred. Good Air	118	4	122
Pred. Bad Air	30	36	66
Total	148	40	188

```r
#Accuracy
acc_int_u <- mean(logistic.pred_int_u==y.test)
cat("Accuracy: ", acc_int_u, "\n")
```

```
## Accuracy:  0.8191489
```

```r
#Precision
prec_int_u <- conf.matrix_int_u[2,2]/sum(conf.matrix_int_u[2,])
cat("Precision: ", prec_int_u, "\n")
```

```
## Precision:  0.5454545
```

```r
#Recall
recall_int_u <- conf.matrix_int_u[2,2]/sum(conf.matrix_int_u[,2])
cat("Recall: ", recall_int_u, "\n")
```

```
## Recall:  0.9
```

```r
#F1-Score
F1_score_int_u <- 2*(prec_int_u*recall_int_u)/(prec_int_u + recall_int_u)
cat("F1-score: ", F1_score_int_u, "\n")
```

```
## F1-score:  0.6792453
```

4.4 Discriminant Analysis

Two classification algorithms commonly used for solving classification problems are Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). These algorithms rely on assumptions about the normality of the covariates' distributions in the two classes, "Bad Air Quality" and "Good Air Quality". To validate this assumption the Shapiro-Wilk test has been employed. The Shapiro-Wilk test is a hypothesis test for verifying the normality. By setting a significance level, typically 0.05, if the p-value is less than or equal to 0.05, the alternative hypothesis, which states that the distribution of the data is not normal, is accepted. In the context of this analysis, the Shapiro-Wilk test will be exclusively applied to the original unbalanced training dataset.

```
#Shapiro-Wilk test: T_max
shapiro.test(train_data$T_max[train_data$BadAirQuality==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$T_max[train_data$BadAirQuality == 1]
## W = 0.94033, p-value = 1.967e-06
```

```
shapiro.test(train_data$T_max[train_data$BadAirQuality==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$T_max[train_data$BadAirQuality == 0]
## W = 0.97659, p-value = 5.631e-08
```

```
#Shapiro-Wilk test: Wind_speed_med
shapiro.test(train_data$Wind_speed_med[train_data$BadAirQuality==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Wind_speed_med[train_data$BadAirQuality == 0]
## W = 0.84464, p-value < 2.2e-16
```

```
shapiro.test(train_data$Wind_speed_med[train_data$BadAirQuality==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Wind_speed_med[train_data$BadAirQuality == 1]
## W = 0.93402, p-value = 6.37e-07
```

```
#Shapiro-Wilk test: Pressure
shapiro.test(train_data$Pressure[train_data$BadAirQuality==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Pressure[train_data$BadAirQuality == 0]
## W = 0.98615, p-value = 2.716e-05
```

```r
shapiro.test(train_data$Pressure[train_data$BadAirQuality==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Pressure[train_data$BadAirQuality == 1]
## W = 0.98849, p-value = 0.1939
```

```r
#Shapiro-Wilk test: Humidity
shapiro.test(train_data$Humidity[train_data$BadAirQuality==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Humidity[train_data$BadAirQuality == 0]
## W = 0.97344, p-value = 1.009e-08
```

```r
shapiro.test(train_data$Humidity[train_data$BadAirQuality==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Humidity[train_data$BadAirQuality == 1]
## W = 0.95294, p-value = 2.312e-05
```

```r
#Shapiro-Wilk test: Visibility
shapiro.test(train_data$Visibility[train_data$BadAirQuality==1])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Visibility[train_data$BadAirQuality == 1]
## W = 0.91462, p-value = 2.8e-08
```

```r
shapiro.test(train_data$Visibility[train_data$BadAirQuality==0])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  train_data$Visibility[train_data$BadAirQuality == 0]
## W = 0.81532, p-value < 2.2e-16
```

While all the variables, except for the *Pressure* distribution in the "Bad Air Quality" class, do not meet the normality assumption, both Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) will be performed, despite the initial assumptions. In fact, although real-world data often deviates from strict normality assumptions, LDA and QDA can still perform well in classification tasks.

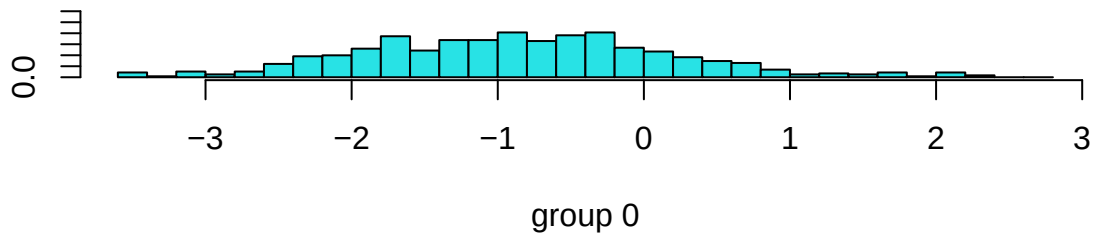
4.4.1 Linear Discriminant Analysis

LDA generates a linear decision boundary and one of its key assumption is that all classes share the same covariance matrix. It will be performed both on the original dataset and on the balanced one, considering different thresholds. The features considered to fit the model are the one selected by the stepwise backward procedure, since it results in best performances as the removal of irrelevant and redundant features avoid overfitting.

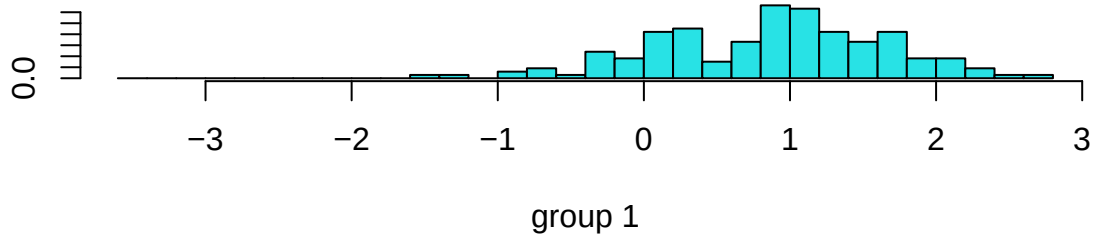
```
#LDA on the original unbalanced dataset
lda.fit <- lda(BadAirQuality~.-Humidity-Month-Year-Day_of_week,data=train_data)
lda.fit
```

```
## Call:
## lda(BadAirQuality ~ . - Humidity - Month - Year - Day_of_week,
##     data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.7765814 0.2234186
##
## Group means:
##      T_max Visibility Wind_speed_med Pressure PhenomenaFog PhenomenaRain
## 0 7.818024   16.38995      11.175043 1015.705    0.1299827    0.31542461
## 1 2.614458   11.79518       7.301205 1021.416    0.2530120    0.09638554
##
## Coefficients of linear discriminants:
##                         LD1
## T_max          -0.11307230
## Visibility     -0.10724902
## Wind_speed_med -0.09742866
## Pressure        0.04168477
## PhenomenaFog   -0.77065351
## PhenomenaRain  -0.58206813
```

```
#Histograms of the discriminant scores of two classes
plot(lda.fit, type="histogram")
```



group 0



group 1

```r
#Prediction
lda.pred <- predict(lda.fit, newdata=test_data, type="response")

#Posterior probabilities
post_lda<- lda.pred$posterior

train_index <- createDataPartition(train_data$BadAirQuality, p=0.8,list=FALSE)
final_train <- train_data[train_index]
val_data <- train_data[-train_index]

#Different thresholds tested (0.3,0.4,0.5)
pred_lda<- as.factor(ifelse(post_lda[,2] > 0.3, 1, 0))

#Confusion matrix with the best threshold: 0.3
conf.matrix_lda <- table(pred_lda, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_lda
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)
```

```
#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of LDA on the original dataset.
Best threshold: 0.3") %>% kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 10: Confusion Matrix of LDA on the original dataset. Best threshold: 0.3

	True Good Air	True Bad Air	Total
Pred. Good Air	127	11	138
Pred. Bad Air	21	29	50
Total	148	40	188

```
#Accuracy
acc_lda <- mean(pred_lda==y.test)
cat("Accuracy: ", acc_lda, "\n")
```

```
## Accuracy:  0.8297872
```

```
#Precision
prec_lda <- conf.matrix_lda[2,2]/sum(conf.matrix_lda[2,])
cat("Precision: ", prec_lda, "\n")
```

```
## Precision:  0.58
```

```
#Recall
recall_lda <- conf.matrix_lda[2,2]/sum(conf.matrix_lda[,2])
cat("Recall: ", recall_lda, "\n")
```

```
## Recall:  0.725
```

```
#F1-Score
F1_score_lda <- 2*(prec_lda*recall_lda)/(prec_lda + recall_lda)
cat("F1-score: ", F1_score_lda, "\n")
```

```
## F1-score:  0.6444444
```

LDA on the undersampled dataset

```
#LDA on the balanced dataset
lda.under <- lda(BadAirQuality~.-Humidity-Month-Year-Day_of_week,
                 data=train_balanced)
lda.under
```

```
## Call:
## lda(BadAirQuality ~ . - Humidity - Month - Year - Day_of_week,
##     data = train_balanced)
##
```

```
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##      T_max Visibility Wind_speed_med Pressure PhenomenaFog PhenomenaRain
## 0 7.975904   16.01807      10.590361 1015.151     0.126506    0.32530120
## 1 2.614458   11.79518       7.301205 1021.416     0.253012    0.09638554
##
## Coefficients of linear discriminants:
##                         LD1
## T_max          -0.12790232
## Visibility     -0.08852542
## Wind_speed_med -0.13529245
## Pressure        0.04181623
## PhenomenaFog   -0.79282221
## PhenomenaRain  -0.66937724
```

```r
#Prediction
lda.pred_u <- predict(lda.under, newdata=test_data, type="response")

#Posterior probabilities
post_lda_u<- lda.pred_u$posterior
#Setting the threshold: different thresholds tested (0.4,0.5,0.6)
pred_lda_u<- as.factor(ifelse(post_lda_u[,2] > 0.5, 1, 0))

#Confusion matrix with the best threshold: 0.3
conf.matrix_lda_u <- table(pred_lda_u, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_lda_u
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of LDA on the undersampled dataset.
Best threshold: 0.5") %>% kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 11: Confusion Matrix of LDA on the undersampled dataset. Best threshold: 0.5

	True Good Air	True Bad Air	Total
Pred. Good Air	121	5	126
Pred. Bad Air	27	35	62
Total	148	40	188

```r
#Accuracy
acc_lda_u <- mean(pred_lda_u==y.test)
cat("Accuracy: ", acc_lda_u, "\n")
```

```
## Accuracy:  0.8297872
```

```r
#Precision
prec_lda_u <- conf.matrix_lda_u[2,2]/sum(conf.matrix_lda_u[2,])
cat("Precision: ", prec_lda_u, "\n")
```

```
## Precision:  0.5645161
```

```r
#Recall
recall_lda_u <- conf.matrix_lda_u[2,2]/sum(conf.matrix_lda_u[,2])
cat("Recall: ", recall_lda_u, "\n")
```

```
## Recall:  0.875
```

```r
#F1-Score
F1_score_lda_u <- 2*(prec_lda_u*recall_lda_u)/(prec_lda_u + recall_lda_u)
cat("F1-score: ", F1_score_lda_u, "\n")
```

```
## F1-score:  0.6862745
```

Similarly to what happens with the Logistic Regression model, the model trained over the balanced set obtained through undersampling outperforms the one trained over the original set. Even if the accuracy is the same, the sensitivity is higher (0.875 vs 0.725), and thus, the F1-score. Looking at the histograms of the discriminant scores for each class, it's evident that the two groups are not very well separated and there's a lot of overlap. This shows that the model has difficulty in clearly distinguishing between the poor and good air classes and for this reason it is less effective compared to the Logistic Regression.

4.4.2 Quadratic Discriminant Analysis (QDA)

The LDA assumption that all the classes share the same covariance matrix can be quite restrictive. QDA is more flexible in its approach and allows each class to have its own covariance matrix. Moreover, its the decision boundary is not linear, as the one of LDA, but quadratic.

```r
#QDA on the original unbalanced dataset
qda.fit <- qda(BadAirQuality~.-Humidity-Month-Year-Day_of_week,data=train_data)
qda.fit
```

```
## Call:
## qda(BadAirQuality ~ . - Humidity - Month - Year - Day_of_week,
##     data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.7765814 0.2234186
##
## Group means:
##      T_max Visibility Wind_speed_med Pressure PhenomenaFog PhenomenaRain
## 0 7.818024   16.38995      11.175043 1015.705    0.1299827    0.31542461
## 1 2.614458   11.79518       7.301205 1021.416    0.2530120    0.09638554
```

```r
#Prediction
qda.pred <- predict(qda.fit, newdata=test_data, type="response")

#Posterior probabilities
post_qda<- qda.pred$posterior
#Setting the threshold: different thresholds tested (0.3,0.4,0.5)
pred_qda<- as.factor(ifelse(post_qda[,2] > 0.3, 1, 0))

#Confusion matrix with the best threshold: 0.3
conf.matrix_qda <- table(pred_qda, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_qda
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of QDA on the original dataset.
Best threshold: 0.3") %>% kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 12: Confusion Matrix of QDA on the original dataset. Best threshold: 0.3

	True Good Air	True Bad Air	Total
Pred. Good Air	124	7	131
Pred. Bad Air	24	33	57
Total	148	40	188

```r
#Accuracy
acc_qda <- mean(pred_qda==y.test)
cat("Accuracy: ", acc_qda, "\n")
```

```
## Accuracy:  0.8351064
```

```r
#Precision
prec_qda <- conf.matrix_qda[2,2]/sum(conf.matrix_qda[2,])
cat("Precision: ", prec_qda, "\n")
```

```
## Precision:  0.5789474
```

```r
#Recall
recall_qda <- conf.matrix_qda[2,2]/sum(conf.matrix_qda[,2])
cat("Recall: ", recall_qda, "\n")
```

```
## Recall:  0.825
```

```
#F1-Score
F1_score_qda <- 2*(prec_qda*recall_qda)/(prec_qda + recall_qda)
cat("F1-score: ", F1_score_qda, "\n")
```

```
## F1-score:  0.6804124
```

QDA on the undersampled dataset

```
#QDA on the balanced dataset
qda.under <- qda(BadAirQuality~.-Humidity-Month-Year-Day_of_week,
                 data=train_balanced)
qda.under
```

```
## Call:
## qda(BadAirQuality ~ . - Humidity - Month - Year - Day_of_week,
##     data = train_balanced)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##      T_max Visibility Wind_speed_med Pressure PhenomenaFog PhenomenaRain
## 0 7.975904   16.01807      10.590361 1015.151     0.126506    0.32530120
## 1 2.614458   11.79518       7.301205 1021.416     0.253012    0.09638554
```

```
#Prediction
qda.pred_u <- predict(qda.under, newdata=test_data, type="response")

#Posterior probabilities
post_qda_u<- qda.pred_u$posterior
#Setting the threshold: different thresholds tested (0.4,0.5,0.6)
pred_qda_u<- as.factor(ifelse(post_qda_u[,2] > 0.5, 1, 0))

#Confusion matrix with the best threshold: 0.3
conf.matrix_qda_u <- table(pred_qda_u, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_qda_u
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of QDA on the undersampled dataset.
Best threshold: 0.5") %>% kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 13: Confusion Matrix of QDA on the undersampled dataset. Best threshold: 0.5

	True Good Air	True Bad Air	Total
Pred. Good Air	123	6	129
Pred. Bad Air	25	34	59
Total	148	40	188

```
#Accuracy
acc_qda_u <- mean(pred_qda_u==y.test)
cat("Accuracy: ", acc_qda_u, "\n")
```

```
## Accuracy:  0.8351064
```

```
#Precision
prec_qda_u <- conf.matrix_qda_u[2,2]/sum(conf.matrix_qda_u[2,])
cat("Precision: ", prec_qda_u, "\n")
```

```
## Precision:  0.5762712
```

```
#Recall
recall_qda_u <- conf.matrix_qda_u[2,2]/sum(conf.matrix_qda_u[,2])
cat("Recall: ", recall_qda_u, "\n")
```

```
## Recall:  0.85
```

```
#F1-Score
F1_score_qda_u <- 2*(prec_qda_u*recall_qda_u)/(prec_qda_u + recall_qda_u)
cat("F1-score: ", F1_score_qda_u, "\n")
```

```
## F1-score:  0.6868687
```

The performances on the original training set are the same in term of accuracy (0.835) compared to the ones obtained with the undersampled dataset even if the latter has a slightly lower false negative rate (recall values are 0.825 and 0.85 respectively). However, the results are quite similar to the ones obtained using Linear Discriminant Analysis.

Models comparison: Original vs Undersampled Dataset

From an initial comparison of the models, fitted utilizing the optimal threshold for each, it's possible to see that the balanced dataset achieves better performances both in terms of accuracy and in terms of sensitivity for all the models, except the Logistic Regression with the interaction terms. The lower accuracy of this model on the balanced dataset, compared to the original one, can be attributed to the removal of important occurrences, due to the undersampling procedure, that were relevant in capturing some interactions between the covariates. However, the model with interactions terms fitted on the undersampled dataset has a higher sensitivity. Therefore, based on the outcomes achieved so far, the analysis will proceed with fitting the models exclusively on the balanced dataset.

```r
#Model metrics for the original training set
table_data <- data.frame(
Model = c("Logistic Regression", "Logistic with interactions", "LDA", "QDA"),
Accuracy = round(c(acc_lr_3,acc_int, acc_lda, acc_qda), digits=3),
Sensitivity = round(c(recall_lr_3,recall_int, recall_lda, recall_qda), digits=3),
F1_score = round(c(F1_score_3,F1_score_int, F1_score_lda, F1_score_qda), digits=3)
)
kable(table_data, format = "latex", booktabs = T,
caption = "Model metrics for Original Train Set") %>%
kable_styling(latex_options = c("striped", "hold_position"))
```

Table 14: Model metrics for Original Train Set

Model	Accuracy	Sensitivity	F1_score
Logistic Regression	0.856	0.800	0.703
Logistic with interactions	0.851	0.800	0.696
LDA	0.830	0.725	0.644
QDA	0.835	0.825	0.680

```r
#Model metrics for the undersampled dataset
table_data <- data.frame(
Model = c("Logistic Regression","Logistic with interactions", "LDA", "QDA"),
Accuracy = round(c(acc_lr_u,acc_int_u, acc_lda_u, acc_qda_u), digits=3),
Sensitivity = round(c(recall_lr_u,recall_int_u, recall_lda_u, recall_qda_u), digits=3),
F1_score = round(c(F1_score_u,F1_score_int_u, F1_score_lda_u, F1_score_qda_u), digits=3)
)
kable(table_data, format = "latex", booktabs = T,
caption = "Model metrics for the Undersampled Train Set") %>%
kable_styling(latex_options = c("striped", "hold_position"))
```

Table 15: Model metrics for the Undersampled Train Set

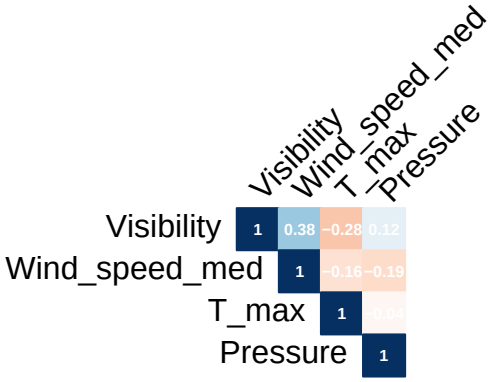
Model	Accuracy	Sensitivity	F1_score
Logistic Regression	0.862	0.800	0.711
Logistic with interactions	0.819	0.900	0.679
LDA	0.830	0.875	0.686
QDA	0.835	0.850	0.687

4.5 Naive Bayes

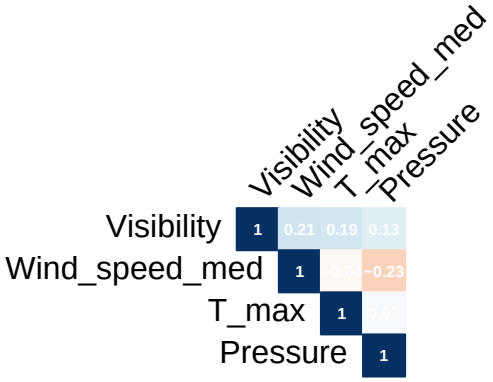
Another model that can be used in classification tasks is Naive Bayes. It relies on the assumption that features within each class are independent of each other. This independence assumption simplifies the computation of joint probability distributions in Bayes' formula, allowing for the use of marginal distributions. As it's evident from the two correlation matrices below, some predictors are slightly correlated, and thus dependent. However, the Naive Bayes model is known for its robustness and often produces good results, even when the assumption of independence is violated. Therefore, Naive Bayes model remains a possible approach. The model is fitted only on the variables selected by the backward stepwise process and its decision boundary has been adjusted varying thresholds on the posterior probabilities. The best threshold results to be 0.6.

```r
#Numerical features extraction for the two classes
num_features_poor <- subset(train_balanced[train_balanced$BadAirQuality==1,],
                            select=-c(Phenomena, Month, Year, Day_of_week, BadAirQuality,Humidity))

num_features_good <- subset(train_balanced[train_balanced$BadAirQuality==0,],
                            select=-c(Phenomena, Month, Year, Day_of_week, BadAirQuality,Humidity))

#Correlation matrix of the numerical variables
cor_matrix_poor <- cor(num_features_poor)
cor_matrix_good <- cor(num_features_good)


# Plot the correlation matrices
par(mfrow=c(1,2))
corrplot(cor_matrix_poor,method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45, addCoef.col = "white",
         number.cex=0.5, cl.pos="n",main="\n\nBadAirQuality=1")
corrplot(cor_matrix_good,method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45, addCoef.col = "white",
         number.cex=0.5, cl.pos="n", main="\n\nBadAirQuality=0")
```

BadAirQuality=1



BadAirQuality=0




```r
#Naive Bayes model
nb.fit <- naiveBayes(BadAirQuality ~.-Year-Month-Day_of_week-Humidity,
                     data = train_balanced)
nb.fit
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   0   1
## 0.5 0.5
##
## Conditional probabilities:
##    T_max
## Y       [,1]     [,2]
##   0 7.975904 5.759789
##   1 2.614458 4.237871
##
##    Visibility
## Y       [,1]     [,2]
##   0 16.01807 5.461128
##   1 11.79518 6.839740
##
##    Wind_speed_med
## Y        [,1]     [,2]
##   0 10.590361 3.889492
##   1  7.301205 2.007471
##
##    Pressure
## Y       [,1]     [,2]
##   0 1015.151 8.366323
##   1 1021.416 7.750785
##
##    Phenomena
## Y     No event        Fog       Rain
##   0 0.54819277 0.12650602 0.32530120
##   1 0.65060241 0.25301205 0.09638554
```

```r
#Predictions
pred_nb <- predict(nb.fit, test_data, type = "raw")

#Setting the threshold: different thresholds tested (0.4,0.5,0.6)
nb.class <- ifelse(pred_nb[, 2] > 0.6, 1,0)

#Confusion matrix
conf.matrix_nb <- table(nb.class, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_nb
conf.df[,"Total"] <- rowSums(conf.df)
```

```r
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of Naive Bayes on the undersampled dataset.
Best threshold: 0.6") %>% kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 16: Confusion Matrix of Naive Bayes on the undersampled dataset. Best threshold: 0.6

	True Good Air	True Bad Air	Total
Pred. Good Air	124	7	131
Pred. Bad Air	24	33	57
Total	148	40	188

```r
#Accuracy
acc_nb <- mean(nb.class==y.test)
cat("Accuracy: ", acc_nb, "\n")
```

```
## Accuracy:  0.8351064
```

```r
#Precision
prec_nb <- conf.matrix_nb[2,2]/sum(conf.matrix_nb[2,])
cat("Precision: ", prec_nb, "\n")
```

```
## Precision:  0.5789474
```

```r
#Recall
recall_nb <- conf.matrix_nb[2,2]/sum(conf.matrix_nb[,2])
cat("Recall: ", recall_nb, "\n")
```

```
## Recall:  0.825
```

```r
#F1-Score
F1_score_nb <- 2*(prec_nb*recall_nb)/(prec_nb + recall_nb)
cat("F1-score: ", F1_score_nb, "\n")
```

```
## F1-score:  0.6804124
```

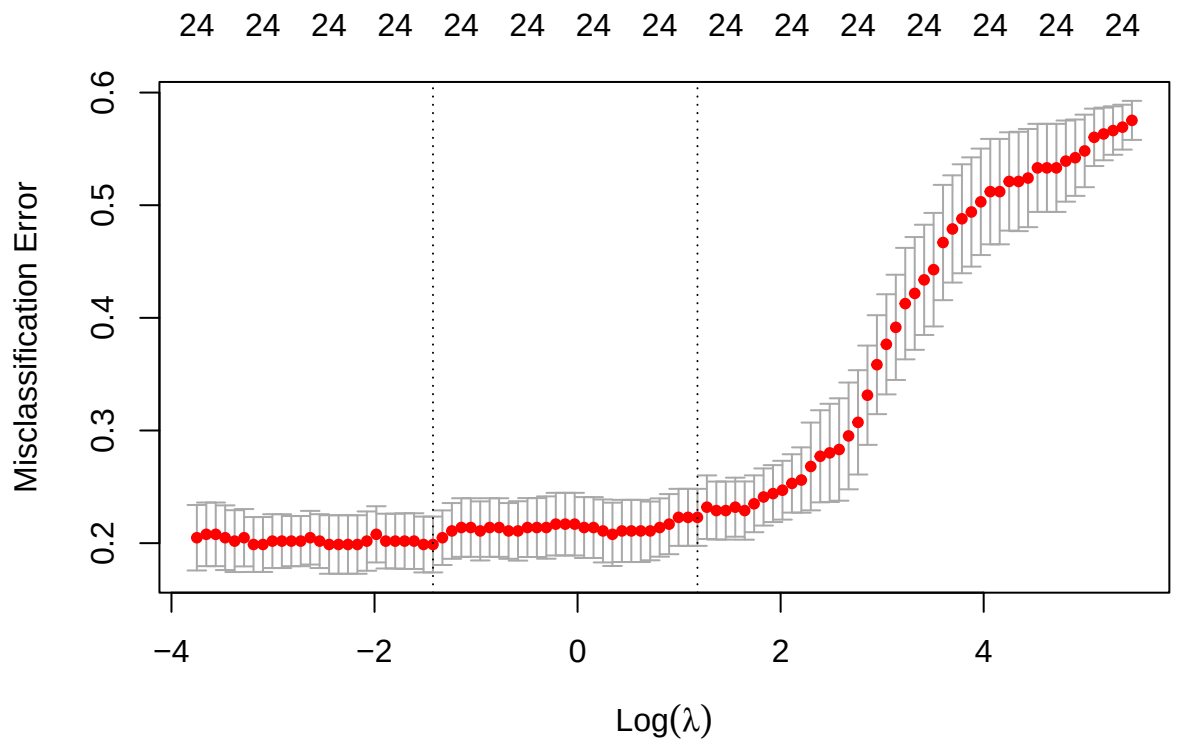
4.6 Shrinkage methods

The analysis continues with the regularized methods: Ridge and Lasso Regression. To control the complexity of the models they introduce a penalization term which shrinks the values of the coefficients towards zero. In Ridge Regression all the variables are kept, while in Lasso Regression some of the coefficients are set exactly equal to zero. The regularization parameter λ has been selected by the means of a 10-fold cross validation as the one that results in having the minimum classification error.

4.6.1 Ridge Regression

Ridge Regression applies an L2 penalty, which imposes quadratic shrinkage on the coefficients of the model. It is specifically suited in collinearity problems (where Lasso fails) since makes the coefficients' estimates more stable.

```
#Model matrix: training set
x_train<-model.matrix(~.-1, data=train_balanced[,-10])
#Response values of the training set
y.train <- train_balanced$BadAirQuality

#Model matrix: test set
x_test<-model.matrix(~.-1, data=test_data)

#Ridge regression: cross validation to select the best lambda based
#on the classification error
cvfit.ridge <- cv.glmnet(x_train, y.train, alpha=0,family="binomial",
                         type.measure="class",set.seed=123)
plot(cvfit.ridge)
```




```
#Best lambda
best_lambda <- cvfit.ridge$lambda.min
best_lambda
```

```
## [1] 0.2407043
```

```r
pred.ridge <- predict(cvfit.ridge, x_test, type="class", s=best_lambda)

#Confusion matrix
conf.matrix_ridge <- table(pred.ridge, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_ridge
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of Ridge Regression on the undersampled dataset") %>%
  kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 17: Confusion Matrix of Ridge Regression on the undersampled dataset

	True Good Air	True Bad Air	Total
Pred. Good Air	123	6	129
Pred. Bad Air	25	34	59
Total	148	40	188

```r
#Accuracy
acc_ridge <- mean(pred.ridge==y.test)
cat("Accuracy: ", acc_ridge, "\n")
```

```
## Accuracy:  0.8351064
```

```r
#Precision
prec_ridge <- conf.matrix_ridge[2,2]/sum(conf.matrix_ridge[2,])
cat("Precision: ", prec_ridge, "\n")
```

```
## Precision:  0.5762712
```

```r
#Recall
recall_ridge <- conf.matrix_ridge[2,2]/sum(conf.matrix_ridge[,2])
cat("Recall: ", recall_ridge, "\n")
```

```
## Recall:  0.85
```

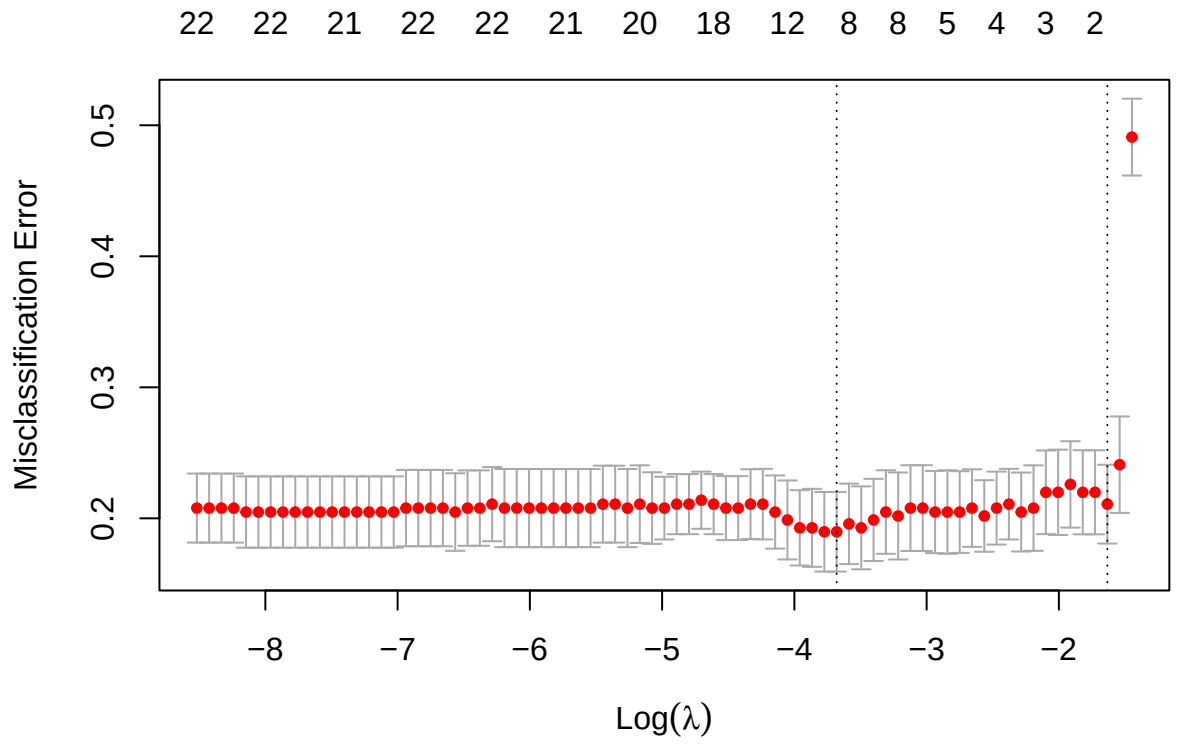
```r
#F1-Score
F1_score_ridge <- 2*(prec_ridge*recall_ridge)/(prec_ridge + recall_ridge)
cat("F1-score: ", F1_score_ridge, "\n")
```

```
## F1-score:  0.6868687
```

4.6.2 Lasso Regression

Lasso Regression incorporates an L1 penalty into the model, logistic regression in the present context. This approach significantly influences the model's coefficients: the L1 penalty has the effect of pushing the coefficients of the least relevant covariates exactly equal to zero, effectively performing variable selection.

```
#Lasso regression: cross validation to select the best lambda based on
#the classification error
set.seed(123)
cvfit.lasso <- cv.glmnet(x_train, y.train, alpha=1,family="binomial",
                         type.measure="class",
                         set.seed=123)
plot(cvfit.lasso)
```




```
#Best lambda
best_lambda <- cvfit.lasso$lambda.min
best_lambda
```

```
## [1] 0.02521656
```

```
#Coefficients different from zero
coefs <- coef(cvfit.lasso, s=cvfit.lasso$lambda.min)
coefs
```

```
## 25 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                                s1
## (Intercept)          -41.57689791
## T_max                 -0.14707986
## Humidity               .
## Visibility            -0.06692432
## Wind_speed_med        -0.23337379
## Pressure               0.04427037
## PhenomenaNo event      0.46313581
## PhenomenaFog           .
## PhenomenaRain         -0.09750476
## Year19                 .
## Year21                 .
## Year22                 .
## Month2                 .
## Month3                 .
## Month4                -0.75594965
## Month9                -0.38739212
## Month10                .
## Month11               -0.03225330
## Month12                .
## Day_of_weekMonday      .
## Day_of_weekSaturday    .
## Day_of_weekSunday      .
## Day_of_weekThursday    .
## Day_of_weekTuesday     .
## Day_of_weekWednesday   .
```

The variables selected from retention by the Lasso are the same as the ones kept by the backward stepwise selection, with the difference that, unlike the stepwise process, Lasso keeps some dummy variable related to *Month*, specifically *Month4*, *Month9* and *Month11*.

```
#Predictions
pred.lasso <- predict(cvfit.lasso, x_test, type="class", s=best_lambda)

#Confusion matrix
conf.matrix_lasso <- table(pred.lasso, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_ridge
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of Lasso Regression on the undersampled dataset") %>%
  kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 18: Confusion Matrix of Lasso Regression on the undersampled dataset

	True Good Air	True Bad Air	Total
Pred. Good Air	123	6	129
Pred. Bad Air	25	34	59
Total	148	40	188

```
#Accuracy
acc_lasso <- mean(pred.lasso==y.test)
cat("Accuracy: ", acc_lasso, "\n")
```

```
## Accuracy:  0.8138298
```

```
#Precision
prec_lasso <- conf.matrix_lasso[2,2]/sum(conf.matrix_lasso[2,])
cat("Precision: ", prec_lasso, "\n")
```

```
## Precision:  0.5409836
```

```
#Recall
recall_lasso <- conf.matrix_lasso[2,2]/sum(conf.matrix_lasso[,2])
cat("Recall: ", recall_lasso, "\n")
```

```
## Recall:  0.825
```

```
#F1-Score
F1_score_lasso <- 2*(prec_lasso*recall_lasso)/(prec_lasso + recall_lasso)
cat("F1-score: ", F1_score_lasso, "\n")
```

```
## F1-score:  0.6534653
```

At the initial step of the analysis, to solve collinearity issues, the predictors were chosen based on their correlation with the response variable within the entire training set. Although this procedure could have resulted in too optimistic cross-validation errors, the observed errors are actually good estimates of the true errors computed on the test set, suggesting that the initial concern may be passed over for a simpler implementation approach. The results obtained by the Ridge and Lasso regressions are quite similar in terms of all metrics: Ridge performs well because it keeps all the variables but with smaller-sized coefficients, while Lasso effectively ignores the less important variables, such as *Humidity*, *Year* and almost all the *Month* dummy variables, leading to a simpler model less prone to overfit the data.

4.7 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric algorithm widely used for classification tasks. It is based on the idea that similar data points tend to belong to the same category, thus an unlabeled example is classified by assigning it to the most common class among its k nearest neighbors. These neighbors are the k closest data points in the feature space, where the distance between points determines their closeness. Since the dataset contain both numerical and categorical features, it is not appropriate to use traditional distance metrics such as Euclidean distance. For this reason, a distance that can handle both numerical and categorical features has been chosen, that is Gower distance. The Gower distance between two data points X and Y with n features is computed as follows:

$$\text{Gower}(X, Y) = \frac{\sum_{i=1}^{n} w_i \cdot d(x_i, y_i)}{\sum_{i=1}^{n} w_i}$$

where $d(x_i, y_i)$ is the normalized distance between the two points for each attribute and w_i is the weight given to each attribute, which can be useful if some variables are considered more important than others. For numeric variables, Gower distance uses a normalized version of the L1-norm to ensure that attributes with a large range does not dominate the distance calculation. For categorical covariates, it uses a simple matching coefficient (0/1): if two observations have the same level, the distance for that variable is 0, otherwise 1. K-Nearest Neighbors with Gower distance is implemented using *dpred* library and *knngow()* function. The number of nearest neighbors that might be optimized by cross-validation is set equal to 3.

```
#Features retained by backward stepwise selection
features_ret <- c("T_max","Pressure","Wind_speed_med","Visibility","Phenomena")
#Predictions
pred.knn <- knngow(train_balanced[,c(features_ret, "BadAirQuality")],
                   test_data[,features_ret], k=3)
#Confusion matrix
conf.matrix_knn <- table(pred.knn, y.test)

#Dataframe of the confusion matrix for visualization
conf.df <- as.data.frame(matrix(0, nrow = 3, ncol = 3))
colnames(conf.df) <- c("True Good Air","True Bad Air","Total")
rownames(conf.df) <- c("Pred. Good Air ","Pred. Bad Air", "Total")

#Filling the dataframe with confusion matrix's values
conf.df[1:2,1:2] <- conf.matrix_knn
conf.df[,"Total"] <- rowSums(conf.df)
conf.df["Total",] <- colSums(conf.df)

#Confusion matrix: table
kable(conf.df, format = "latex", booktabs = TRUE,
caption = "Confusion Matrix of K-NN on the undersampled dataset") %>%
  kable_styling(latex_options = "hold_position") %>%
  column_spec(c(1,3), border_right = TRUE)
```

Table 19: Confusion Matrix of K-NN on the undersampled dataset

	True Good Air	True Bad Air	Total
Pred. Good Air	120	8	128
Pred. Bad Air	28	32	60
Total	148	40	188

```
#Accuracy
acc_knn <- mean(pred.knn==y.test)
cat("Accuracy: ", acc_knn, "\n")
```

```
## Accuracy:  0.8085106
```

```
#Precision
prec_knn <- conf.matrix_knn[2,2]/sum(conf.matrix_knn[2,])
cat("Precision: ", prec_knn, "\n")
```

```
## Precision:  0.5333333
```

```
#Recall
recall_knn <- conf.matrix_knn[2,2]/sum(conf.matrix_knn[,2])
cat("Recall: ", recall_knn, "\n")
```

```
## Recall:  0.8
```

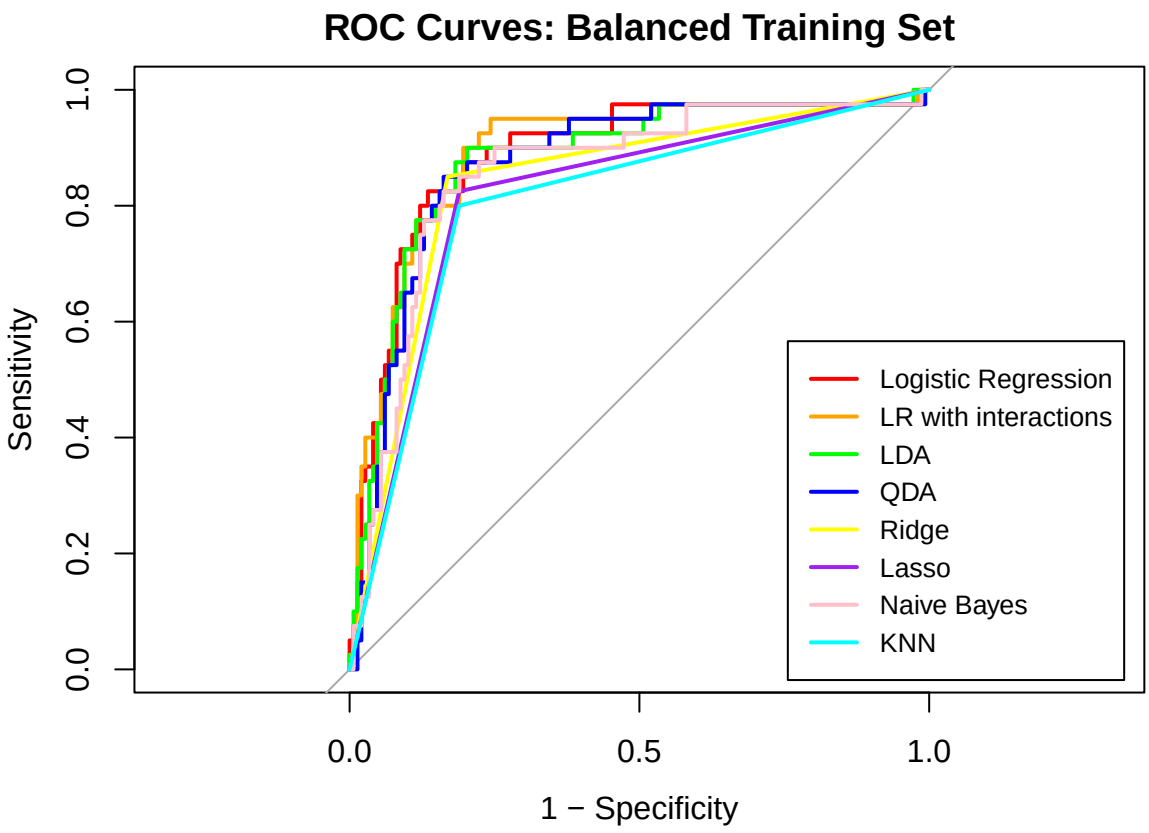
```
#F1-Score
F1_score_knn <- 2*(prec_knn*recall_knn)/(prec_knn + recall_knn)
cat("F1-score: ", F1_score_knn, "\n")
```

```
## F1-score:  0.64
```

5 Models comparison

When comparing different models, the ROC curve (Receiver Operating Characteristic) and AUC (Area Under the Curve) are valuable tools for evaluating the performances of the classification models. The ROC curve is a graphical representation that displays the trade-offs between the True Positive Rate (TPR), represented on the y-axis, and the False Positive Rate (FPR), represented on the x-axis, across different thresholds. The AUC is the area under a ROC curve: if it's equal to 0.5, it suggests no discrimination (equivalent to random guessing), while an AUC of 1.0 indicates perfect prediction. Therefore, higher is the AUC value, better is the model's predictive performance.

```
#ROC curves
roc_lr <- roc(y.test, pred_under)
roc_lr_int <- roc(y.test,glm.pred.int_u)
roc_lda <- roc(y.test,post_lda_u[,2])
roc_qda <- roc(y.test,post_qda_u[,2])
roc_ridge <- roc(y.test,as.numeric(pred.ridge))
roc_lasso <- roc(y.test,as.numeric(pred.lasso))
roc_nb <- roc(y.test, pred_nb[,2])
roc_knn <- roc(y.test, as.numeric(pred.knn))

#Plot the ROC curves on the same graph
plot(roc_lr, col = "red", main = "ROC Curves: Balanced Training Set",
print.auc = FALSE, legacy.axes = TRUE)
lines(roc_lr_int, col = "orange")
lines(roc_lda, col = "green")
lines(roc_qda, col = "blue")
lines(roc_ridge, col = "yellow")
lines(roc_lasso, col = "purple")
lines(roc_nb, col = "pink")
lines(roc_knn, col="cyan")

#Add a legend
legend("bottomright", legend = c("Logistic Regression",
                                 "LR with interactions", "LDA", "QDA",
                                 "Ridge","Lasso","Naive Bayes","KNN"),
col = c( "red","orange", "green","blue","yellow", "purple", "pink","cyan"),
lwd = 2, inset = c(0.02, 0.02), cex=0.8)
```



To have a complete overview, let's take a look at a summary table that includes all the models under analysis fitted on the undersampled dataset and the corresponding metrics considered. All the models under examination, with the exception of the Ridge and Lasso regressions, are the ones fitted on the variables selected by the backward stepwise procedure.

```
#Model comparison on the balanced dataset
table_data <- data.frame(
Model = c("Logistic Regression","Logistic with interactions", "LDA", "QDA",
          "Ridge","Lasso","Naive Bayes","KNN"),
Accuracy = round(c(acc_lr_u, acc_int_u, acc_lda_u, acc_qda_u,acc_ridge,acc_lasso,
                   acc_nb, acc_knn), digits=3),
Recall = round(c(recall_lr_u, recall_int_u, recall_lda_u, recall_qda_u,
                 recall_ridge, recall_lasso, recall_nb, recall_knn), digits=3),
Precision = round(c(prec_lr_u, prec_int_u, prec_lda_u, prec_qda_u,
                    prec_ridge,prec_lasso, prec_nb, prec_knn), digits=3),
F1_score = round(c(F1_score_u, F1_score_int_u, F1_score_lda_u, F1_score_qda_u,
                   F1_score_ridge,F1_score_lasso, F1_score_nb, F1_score_knn),
                 digits=3),
AUC = round(c(roc_lr$auc,roc_lr_int$auc,roc_lda$auc,roc_qda$auc,roc_ridge$auc,
              roc_lasso$auc,roc_nb$auc,roc_knn$auc),3)
)

#Comparison table
kable(table_data, format = "latex", booktabs = T,
caption = "Model comparison on the Undersampled Train Set") %>%
kable_styling(latex_options = c("striped", "hold_position"))
```

Table 20: Model comparison on the Undersampled Train Set

Model	Accuracy	Recall	Precision	F1_score	AUC
Logistic Regression	0.862	0.800	0.640	0.711	0.888
Logistic with interactions	0.819	0.900	0.545	0.679	0.892
LDA	0.830	0.875	0.565	0.686	0.879
QDA	0.835	0.850	0.576	0.687	0.872
Ridge	0.835	0.850	0.576	0.687	0.841
Lasso	0.814	0.825	0.541	0.653	0.818
Naive Bayes	0.835	0.825	0.579	0.680	0.856
KNN	0.809	0.800	0.533	0.640	0.805

All the models have achieved good performances which are quite similar. The models that have the best recall are the Logistic Regression with the interaction terms (0.900) and the LDA (0.875) which are also the models with the highest AUC. However, the overall best model which is able to detect effectively days with poor air quality without increasing significantly the number of false positives, thus having a good precision, is the Logistic Regression model with 0.6 as threshold. It reaches the best accuracy (0.862) and the highest F1-score (0.711) among all the models and has a sensitivity of 0.800.

6 Conclusions

Based on the results achieved from the analysis, it seems that meteorological conditions can accurately predict whether the PM10 particulate levels in the air will be high or low in the following day. Some important considerations can be highlighted.

- During warmest months PM10 is not a serious concern.
- Year, Day_of_week and Humidity do not add any useful information to predict the response but increase the model complexity leading to overfitting. These variables are in fact removed both by the stepwise selection process and by the Lasso regression.
- Rainy days and specifically storms reduce significantly the dust particles in the air making it more clean and highly decreasing the probability of having poor air quality the next day.
- The most significant variables to predict air quality are:
  - Phenomena, above rain, also the presence of fog tends to lower the likelihood of high PM10 values the following day.
  - Visibility, T_max and Wind_speed_med. High values exhibit a negative influence on this probability.
  - Pressure plays a significant role as well. High pressure systems where vertical motion of the air is suppressed increases the probability of having poor air quality the next day.