document

Load dataset

```r
library(readr)
database <- read_rds("../data/database.rds")

colnames(database)
```

```
 [1] "Estación"                   "Temperatura (°C)"
 [3] "Humedad (%)"                "Presión (hPa)"
 [5] "Velocidad de viento (m/s)" "CO (mg/m3)"
 [7] "NO (ug/m3)"                 "NO2 (ug/m3)"
 [9] "NOX (ug/m3)"                "O3 (ug/m3)"
[11] "PM10 (ug/m3)"               "time"
```

```r
nrow(database)
```

```
[1] 1105
```

```r
summary(database)
```

```
   Estación          Temperatura (°C)  Humedad (%)     Presión (hPa)
 Length:1105        Min.   : 0.00    Min.   : 3.00   Min.   :902.0
 Class :character   1st Qu.:11.40    1st Qu.:32.00   1st Qu.:923.2
 Mode  :character   Median :18.00    Median :46.00   Median :927.3
                    Mean   :17.32    Mean   :48.04   Mean   :926.9
                    3rd Qu.:23.50    3rd Qu.:62.00   3rd Qu.:930.5
                    Max.   :36.60    Max.   :95.00   Max.   :940.0
```

```
 Velocidad de viento (m/s)   CO (mg/m3)       NO (ug/m3)       NO2 (ug/m3)
 Min.   : 0.000            Min.   :0.000   Min.   :  0.02   Min.   :  1.28
 1st Qu.: 0.310            1st Qu.:0.740   1st Qu.:  3.53   1st Qu.: 30.71
 Median : 1.110            Median :1.120   Median : 26.00   Median : 45.56
 Mean   : 1.484            Mean   :1.172   Mean   : 35.36   Mean   : 44.17
 3rd Qu.: 1.940            3rd Qu.:1.580   3rd Qu.: 55.15   3rd Qu.: 54.11
 Max.   :18.060            Max.   :3.460   Max.   :363.27   Max.   :131.05

  NOX (ug/m3)       O3 (ug/m3)       PM10 (ug/m3)
 Min.   :  4.15   Min.   :  0.00   Min.   :  0.00
 1st Qu.: 40.11   1st Qu.: 13.05   1st Qu.: 15.00
 Median : 93.49   Median : 30.55   Median : 29.00
 Mean   : 98.40   Mean   : 51.02   Mean   : 36.75
 3rd Qu.:137.60   3rd Qu.: 43.86   3rd Qu.: 49.00
 Max.   :559.91   Max.   :487.52   Max.   :318.00
                                   NA's   :11
      time
 Min.   :2022-02-11 17:00:00.00
 1st Qu.:2022-05-02 10:00:00.00
 Median :2023-01-13 02:00:00.00
 Mean   :2022-11-29 21:23:43.71
 3rd Qu.:2023-03-27 15:00:00.00
 Max.   :2023-10-17 07:00:00.00
```

Cleaning

- Tener en cuenta Temperatura, Humedad relativa, Presión atmosférica, Velocidad de viento, CO, NO, NO2, O3 como variables predictoras de PM10.
- Remuevo NA

```
library(dplyr)
```

```
Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
data <- database %>% select(-one_of(c("Estación","time")))
data <- data[complete.cases(data),]
```

```
nrow(data)
```

```
[1] 1094
```

```
summary(data)
```

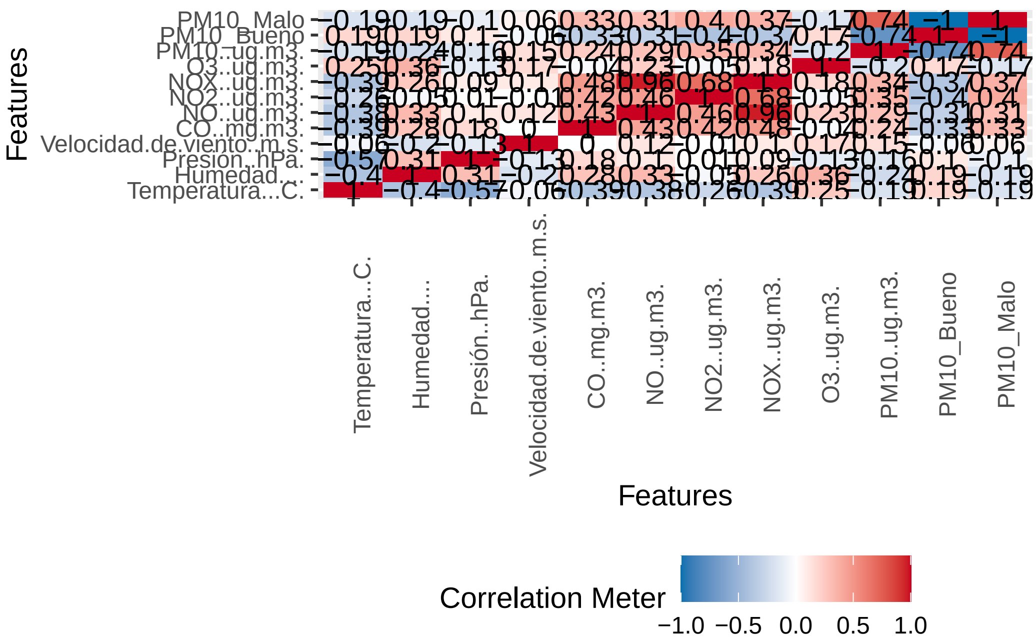
```
 Temperatura (°C)  Humedad (%)   Presión (hPa)   Velocidad de viento (m/s)
 Min.   : 0.00    Min.   : 5.0   Min.   :902.0   Min.   : 0.000
 1st Qu.:11.31    1st Qu.:32.0   1st Qu.:923.3   1st Qu.: 0.310
 Median :17.90    Median :46.0   Median :927.3   Median : 1.110
 Mean   :17.25    Mean   :48.2   Mean   :927.0   Mean   : 1.483
 3rd Qu.:23.45    3rd Qu.:62.0   3rd Qu.:930.5   3rd Qu.: 1.940
 Max.   :36.60    Max.   :95.0   Max.   :940.0   Max.   :18.060
   CO (mg/m3)      NO (ug/m3)       NO2 (ug/m3)      NOX (ug/m3)
 Min.   :0.000   Min.   :  0.020   Min.   :  1.28   Min.   :  4.15
 1st Qu.:0.740   1st Qu.:  3.542   1st Qu.: 30.94   1st Qu.: 40.31
 Median :1.120   Median : 26.045   Median : 45.73   Median : 94.02
 Mean   :1.174   Mean   : 35.203   Mean   : 44.38   Mean   : 98.36
 3rd Qu.:1.577   3rd Qu.: 55.203   3rd Qu.: 54.13   3rd Qu.:137.90
 Max.   :3.460   Max.   :196.860   Max.   :131.05   Max.   :385.19
   O3 (ug/m3)       PM10 (ug/m3)
 Min.   :  0.00   Min.   :  0.00
 1st Qu.: 13.05   1st Qu.: 15.00
 Median : 30.61   Median : 29.00
 Mean   : 51.27   Mean   : 36.75
 3rd Qu.: 43.81   3rd Qu.: 49.00
 Max.   :487.52   Max.   :318.00
```

- Discretizar la variable Material Particulado (PM10) tomando como umbral el valor de 45 µg/m3, por debajo del cual se categorizará como "Bueno". Por encima de 45 µg/m3, se asignará el valor "Malo".

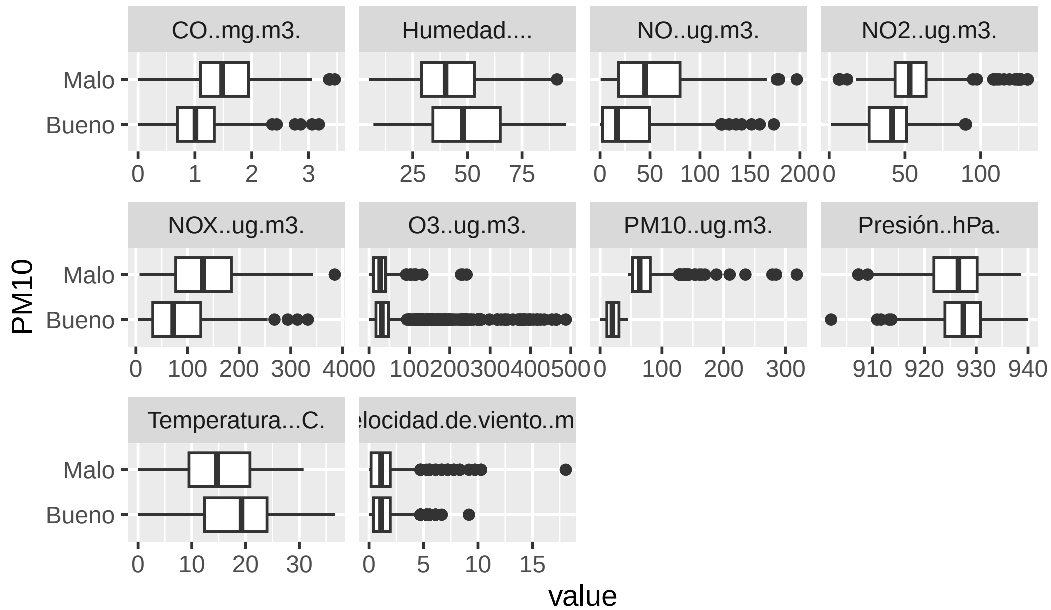
```r
y_col_name <- colnames(data)[10]
y_cut <- cut(data$`PM10 (ug/m3)`,breaks=c(-10,45,400),labels = c("Bueno","Malo"))
data$PM10 <- y_cut
```

Pearson correlation

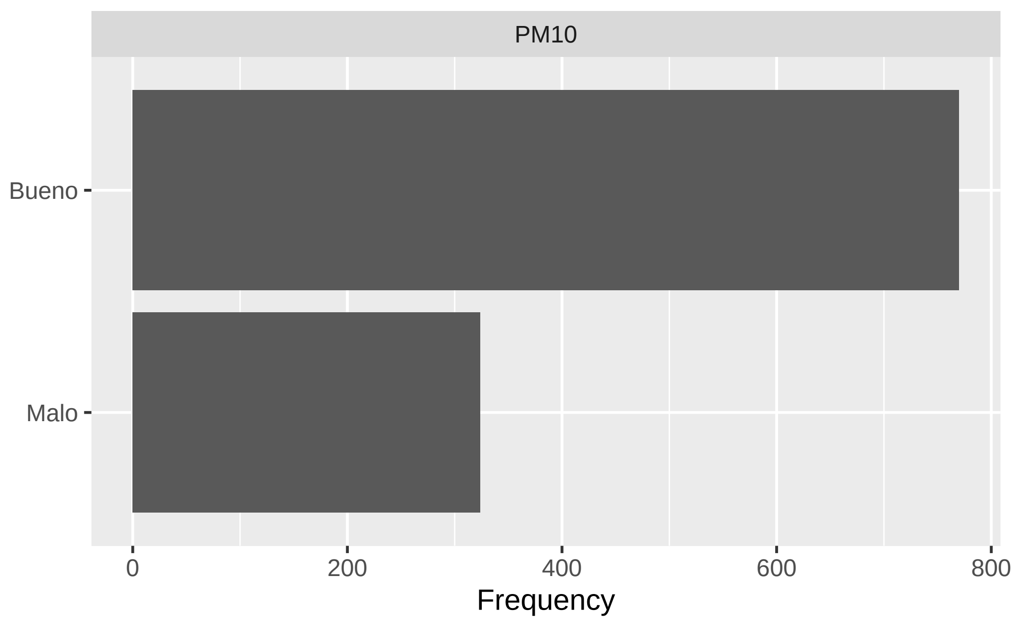
```r
library(DataExplorer)
plot_correlation(data)
```




```r
plot_boxplot(data, by = "PM10")
```




```r
plot_bar(data)
```



```r
data <- data %>% select(-one_of(c("PM10 (ug/m3)")))
```

Modelos

Separación de sets de datos

```r
library(tidymodels)
```

```
-- Attaching packages -------------------------------------- tidymodels 1.2.0 --

v broom        1.0.5      v recipes      1.0.10
v dials        1.2.1      v rsample      1.2.1
v ggplot2      3.5.0      v tibble       3.2.1
v infer        1.0.7      v tune         1.2.1
v modeldata    1.3.0      v workflows    1.1.4
v parsnip      1.2.1      v workflowsets 1.1.0
v purrr        1.0.2      v yardstick    1.3.1

-- Conflicts ----------------------------------------- tidymodels_conflicts() --
x purrr::discard()  masks scales::discard()
x dplyr::filter()   masks stats::filter()
x dplyr::lag()      masks stats::lag()
x yardstick::spec() masks readr::spec()
x recipes::step()   masks stats::step()
* Learn how to get started at https://www.tidymodels.org/start/
```

```r
set.seed(123)
splits      <- initial_split(data, strata = PM10, prop = 3/4)

data_train <- training(splits) # 75 % entrenamiento
data_test  <- testing(splits)  # 25 % en testeo
```

Clasificación binaria

Regresion logistica

```
lr_mod <-
  logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")
```

Receta

```
lr_recipe <-
  recipe(PM10 ~ ., data = data_train) %>%
  step_normalize(all_predictors())
```

Grid tunning

```
lr_workflow <-
  workflow() %>%
  add_model(lr_mod) %>%
  add_recipe(lr_recipe)
```

Since we have only one hyperparameter to tune here, we can set the grid up manually using a one-column tibble with 30 candidate values:

```
lr_reg_grid <- tibble(penalty = 10^seq(-4, -1, length.out = 30))
```

```
lr_reg_grid
```

```
# A tibble: 30 x 1
    penalty
      <dbl>
 1 0.0001
 2 0.000127
 3 0.000161
 4 0.000204
 5 0.000259
 6 0.000329
 7 0.000418
 8 0.000530
 9 0.000672
```

```
10 0.000853
# i 20 more rows
```

Conjunto de validación para usar durante el entrenamiento

```
set.seed(234)
# 20 %
val_set <- validation_split(data_train,
                            strata = PM10,
                            prop = 0.80)
```

```
Warning: `validation_split()` was deprecated in rsample 1.2.0.
i Please use `initial_validation_split()` instead.
```

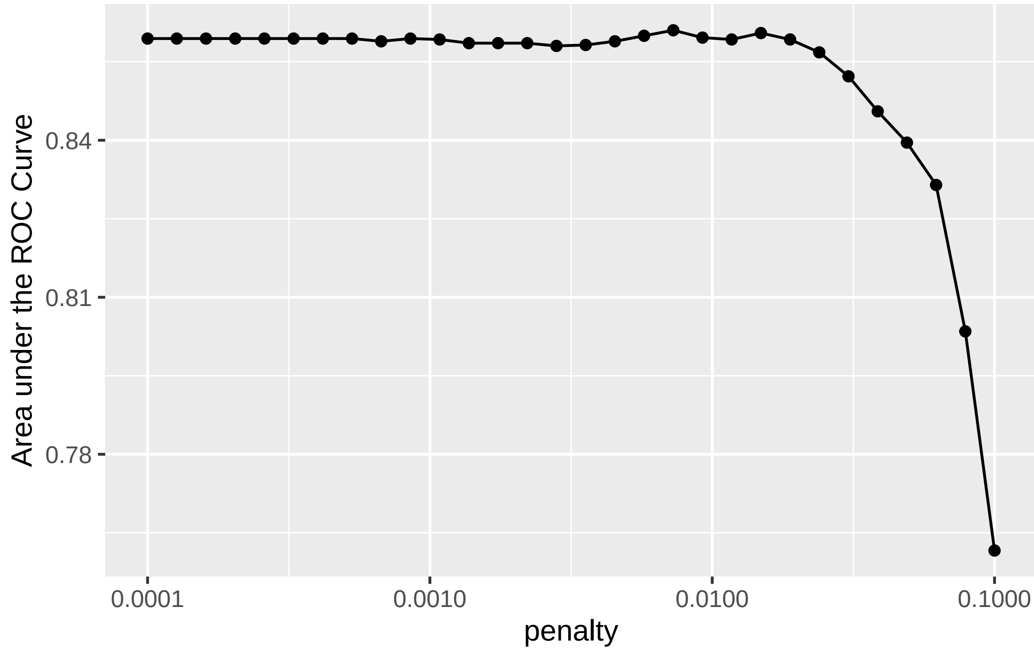
```
lr_res <-
  lr_workflow %>%
  tune_grid(val_set,
            grid = lr_reg_grid,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc))
```

```
lr_res
```

```
# Tuning results
# Validation Set Split (0.8/0.2)  using stratification
# A tibble: 1 x 5
  splits            id         .metrics          .notes           .predictions
  <list>            <chr>      <list>            <list>           <list>
1 <split [655/165]> validation <tibble [30 x 5]> <tibble [0 x 3]> <tibble>
```

```
lr_plot <-
  lr_res %>%
  collect_metrics() %>%
  ggplot(aes(x = penalty, y = mean)) +
  geom_point() +
  geom_line() +
  ylab("Area under the ROC Curve") +
  scale_x_log10(labels = scales::label_number())
```

```r
lr_plot
```



Mejores modelos de Logistic Regression

```r
top_models <-
  lr_res %>%
  show_best(metric = "roc_auc", n = 15) %>%
  arrange(penalty)
top_models
```

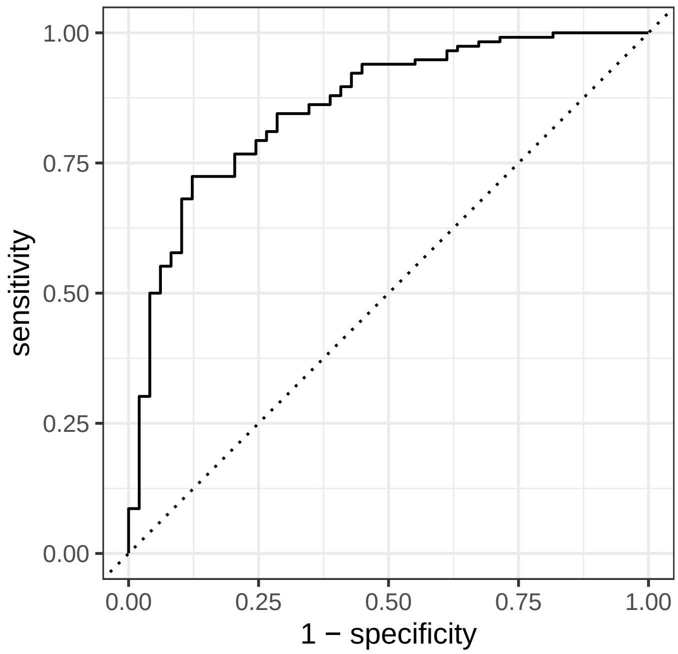
```
# A tibble: 15 x 7
    penalty .metric .estimator  mean     n std_err .config
      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
 1 0.0001   roc_auc binary     0.859     1      NA Preprocessor1_Model01
 2 0.000127 roc_auc binary     0.859     1      NA Preprocessor1_Model02
 3 0.000161 roc_auc binary     0.859     1      NA Preprocessor1_Model03
 4 0.000204 roc_auc binary     0.859     1      NA Preprocessor1_Model04
 5 0.000259 roc_auc binary     0.859     1      NA Preprocessor1_Model05
 6 0.000329 roc_auc binary     0.859     1      NA Preprocessor1_Model06
 7 0.000418 roc_auc binary     0.859     1      NA Preprocessor1_Model07
```

```
 8 0.000530 roc_auc binary     0.859     1      NA Preprocessor1_Model08
 9 0.000853 roc_auc binary     0.859     1      NA Preprocessor1_Model10
10 0.00108  roc_auc binary     0.859     1      NA Preprocessor1_Model11
11 0.00574  roc_auc binary     0.860     1      NA Preprocessor1_Model18
12 0.00728  roc_auc binary     0.861     1      NA Preprocessor1_Model19
13 0.00924  roc_auc binary     0.860     1      NA Preprocessor1_Model20
14 0.0117   roc_auc binary     0.859     1      NA Preprocessor1_Model21
15 0.0149   roc_auc binary     0.860     1      NA Preprocessor1_Model22
```

```r
lr_best <-
  lr_res %>%
  collect_metrics() %>%
  arrange(penalty) %>%
  slice(12) # modelo 12

lr_best
```

```
# A tibble: 1 x 7
  penalty .metric .estimator  mean     n std_err .config
    <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 0.00137 roc_auc binary     0.859     1      NA Preprocessor1_Model12
```

```r
lr_auc <-
  lr_res %>%
  collect_predictions(parameters = lr_best) %>%
  roc_curve(PM10, .pred_Bueno) %>%
  mutate(model = "Logistic Regression")

autoplot(lr_auc)
```



Random Forest

```
cores <- parallel::detectCores()
#cores

rf_mod <-
  rand_forest(mtry = tune(), min_n = tune(), trees = 100) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("classification")
```

```
rf_recipe <-
  recipe(PM10 ~ ., data = data_train)
```

```
rf_workflow <-
  workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(rf_recipe)
```

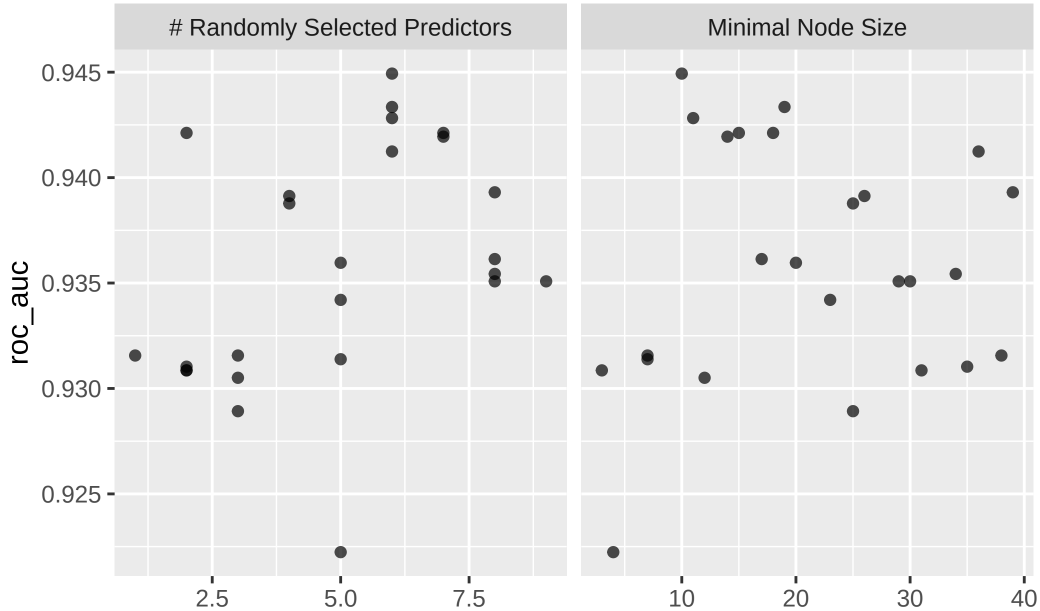
```
set.seed(345)
rf_res <-
  rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc))
```

```
i Creating pre-processing data to finalize unknown parameter: mtry
```

```
rf_res %>%
  show_best(metric = "roc_auc")
```

```
# A tibble: 5 x 8
   mtry min_n .metric .estimator  mean     n std_err .config
  <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1     6    10 roc_auc binary     0.945     1      NA Preprocessor1_Model06
2     6    19 roc_auc binary     0.943     1      NA Preprocessor1_Model10
3     6    11 roc_auc binary     0.943     1      NA Preprocessor1_Model09
4     7    15 roc_auc binary     0.942     1      NA Preprocessor1_Model20
5     2    18 roc_auc binary     0.942     1      NA Preprocessor1_Model24
```

```
autoplot(rf_res)
```




```
rf_best <-
  rf_res %>%
  select_best(metric = "roc_auc")
rf_best
```

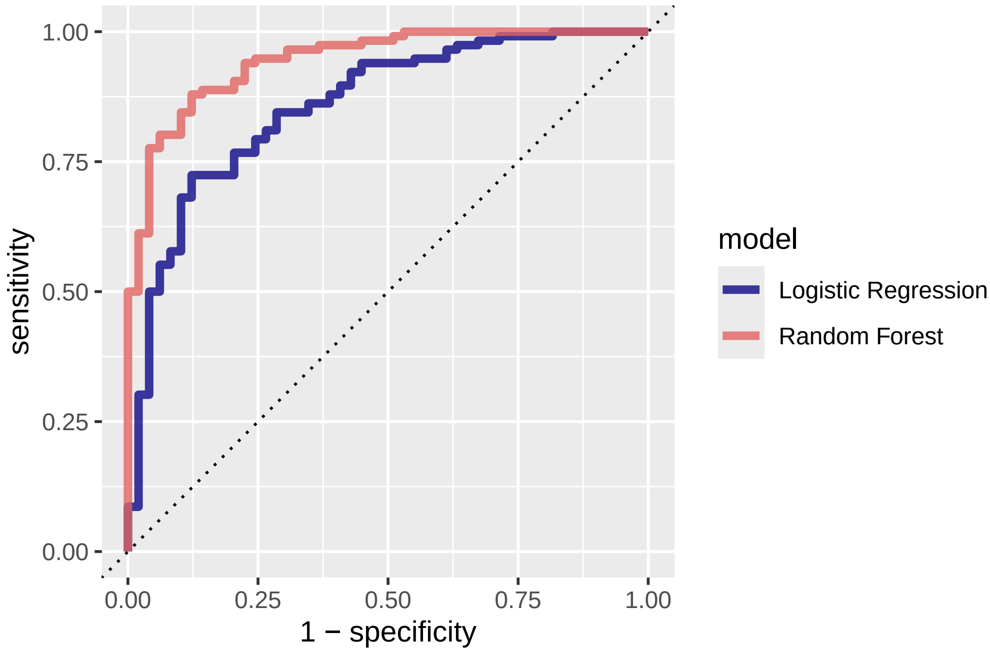
```
# A tibble: 1 x 3
   mtry min_n .config
  <int> <int> <chr>
1     6    10 Preprocessor1_Model06
```

To filter the predictions for only our best random forest model, we can use the parameters argument and pass it our tibble with the best hyperparameter values from tuning, which we called rf_best:

```
rf_auc <-
  rf_res %>%
  collect_predictions(parameters = rf_best) %>%
  roc_curve(PM10, .pred_Bueno) %>%
  mutate(model = "Random Forest")
```

```
bind_rows(rf_auc, lr_auc) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity, col = model)) +
```

```r
  geom_path(lwd = 1.5, alpha = 0.8) +
  geom_abline(lty = 3) +
  coord_equal() +
  scale_color_viridis_d(option = "plasma", end = .6)
```



The random forest is uniformly better across event probability thresholds.

last random forest fit

```r
# the last model
last_rf_mod <-
  rand_forest(mtry = 6, min_n = 10, trees = 100) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("classification")

# the last workflow
last_rf_workflow <-
  rf_workflow %>%
  update_model(last_rf_mod)

# the last fit
```

```r
set.seed(345)
last_rf_fit <-
  last_rf_workflow %>%
  last_fit(splits)

last_rf_fit
```

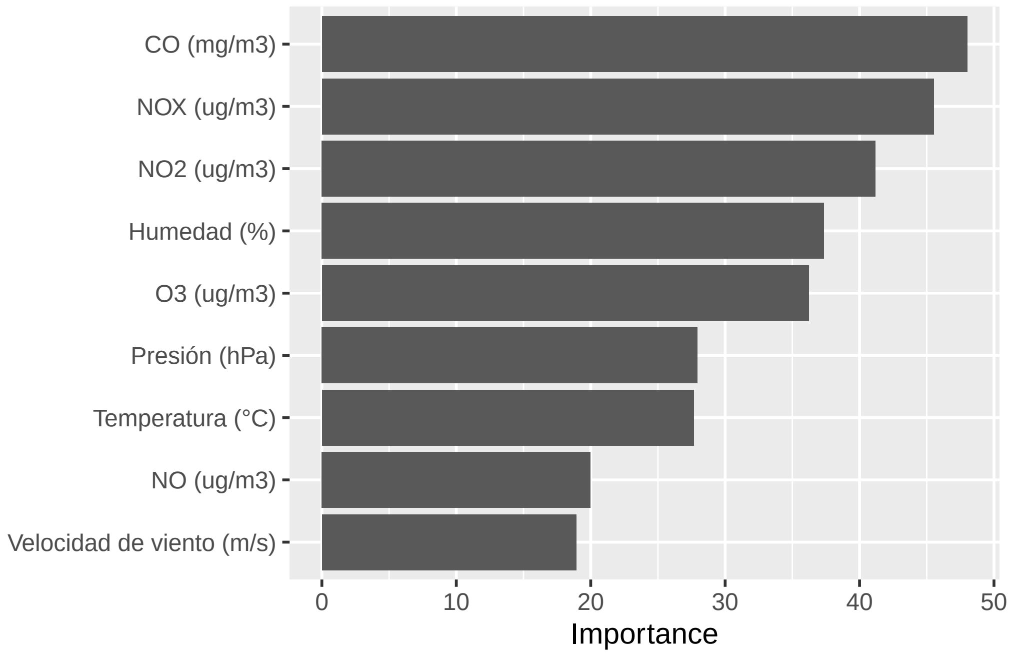
```
# Resampling results
# Manual resampling
# A tibble: 1 x 6
  splits             id               .metrics .notes   .predictions .workflow
  <list>             <chr>            <list>   <list>   <list>       <list>
1 <split [820/274]> train/test split <tibble> <tibble> <tibble>     <workflow>
```

Vip Variable importance

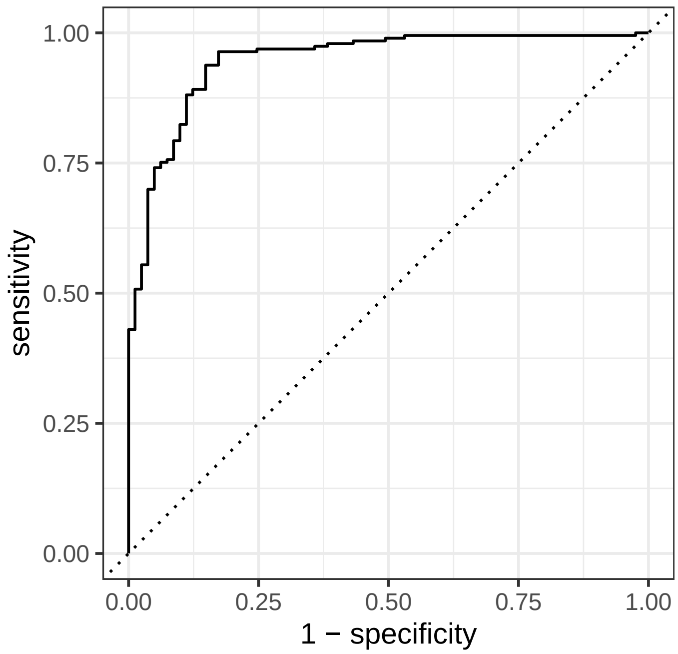
```r
library(vip)
```

```
Attaching package: 'vip'

The following object is masked from 'package:utils':

    vi
```

```r
last_rf_fit %>%
  extract_fit_parsnip() %>%
  vip(num_features = 10)
```




```
last_rf_fit %>%
  collect_predictions() %>%
  roc_curve(PM10, .pred_Bueno) %>%
  autoplot()
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity