Trabajo final

Arturo González Moya

05 febrero, 2021

Contenidos

1 Introducción

En este trabajo realizaremos el estudio de diferentes modelos de predicción sobre los datos de la librería "nycflights13", que contiene la información de los diferentes vuelos que han salido de los aeropuertos de Nueva York en el año 2013.

2 Carga de datos

Lo primero que haremos será cargar las tablas que contienen la información de los vuelos y las que contienen la información del tiempo.

```
#Trata de datos
vuelos = nycflights13::flights
tiempo = nycflights13::weather

tabla= inner_join(vuelos, tiempo)
```

```
## Joining, by = c("year", "month", "day", "origin", "hour", "time_hour")
```

Lo que intentaremos predecir será el retraso que sufren los vuelos con respecto al tiempo que hay en Nueva York. Para ello, las variables que utilizaremos son:

- dep_delay : Esta variable contiene el tiempo que el vuelo retrasó su salida en minutos.
- temp: Es la temperatura (en grados fahrenheit).
- dewp: Es la temperatura a la que debe enfriarse el aire a una presión constante para que se sature de vapor de agua y este se condense (en grados fahrenheit).
- humid: Humedad relativa.
- wind_dir: Dirección del viento (en grados).

- wind_speed: Velocidad del viento (en millas por hora).
- wind_gust: Velocidad de ráfaga (en millas por hora).
- precip: Precipitaciones (en pulgadas).
- pressure: Presión a nivel de mar (en milibares).
- visib: Visibilidad (en millas).

```
head(tabla)
```

```
## # A tibble: 6 x 28
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     1      517            515         2      830
## 2  2013     1     1      533            529         4      850
## 3  2013     1     1      542            540         2      923
## 4  2013     1     1      544            545        -1     1004
## 5  2013     1     1      554            600        -6      812
## 6  2013     1     1      554            558        -4      740
## # ... with 21 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, temp <dbl>, dewp <dbl>, humid <dbl>, wind_dir <dbl>,
## #   wind_speed <dbl>, wind_gust <dbl>, precip <dbl>, pressure <dbl>,
## #   visib <dbl>
```

Como podemos obervar, en la variable "wind_gust" hay una gran cantidad de valores NA, por lo que eliminaremos esta columna de nuestro conjunto de datos para evitar problemas.

```
tabla = subset(tabla, select = -c(wind_gust) )
```

Veamos ahora si hay algún dato perdido en la tabla.

```
#Ver cuantos NA hay y eliminarlos

anyNA(tabla)
```

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

```
table(is.na(tabla))
```

```
##
##   FALSE    TRUE
## 8958900   92040
```

```
mean(is.na(tabla))
```

```
## [1] 0.01016911
```

Como no hay muchos valores NA en comparación con los datos que tenemos, lo que haremos será eliminarlos directamente.

```
tabla=na.omit(tabla)
```

Ahora que ya tenemos los datos limpios, veamos cuantas observaciones poseemos.

```
dim(tabla)
```

```
## [1] 284550     27
```

Tenemos 284550 observaciones con las que hacer nuestra predicción. Como son tantas, lo que haremos será separar en dos conjuntos, uno que sea el de entrenamiento de los modelos y otro que sea el de test de los modelos de predicción.

```
set.seed(12345)
train = sample(1:nrow(tabla), round(nrow(tabla)*0.75,0), replace = FALSE)
entrenamiento = tabla[train, ]
test= tabla[-train, ]
```

3 Métodos de clasificación

3.1 Modelo lineal

Comenzaremos con el modelo más fácil que podemos realizar, el modelo lineal.

```
# modelo lineal
lm.fit=lm(dep_delay ~ temp + dewp + humid + wind_dir + wind_speed + precip + pressure + visib,
            data=entrenamiento)

lm.pred=predict(lm.fit,newdata=test)

rmse_lm= sqrt(MSE(y_pred = lm.pred, y_true = test$dep_delay))
```

Como nuestra variable dependiente (que es "dep_delay") es cuantitativa, lo que utilizaremos para ver la efectividad del modelo será la raiz cuadrada del error cuadrático medio (RMSE). El RMSE de la predicción con el modelo lineal es 36.9489534.

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = dep_delay ~ temp + dewp + humid + wind_dir + wind_speed +
##     precip + pressure + visib, data = entrenamiento)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -71.07  -16.70  -11.15   -0.69 1294.99
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.003e+02  1.316e+01  30.424  < 2e-16 ***
## temp         4.011e-01  4.965e-02   8.079 6.54e-16 ***
## dewp        -3.185e-01  5.348e-02  -5.955 2.60e-09 ***
## humid        2.786e-01  2.735e-02  10.184  < 2e-16 ***
## wind_dir    -5.136e-03  8.978e-04  -5.721 1.06e-08 ***
## wind_speed   3.197e-01  1.679e-02  19.042  < 2e-16 ***
## precip       7.933e+01  6.777e+00  11.705  < 2e-16 ***
## pressure    -4.066e-01  1.234e-02 -32.948  < 2e-16 ***
## visib       -3.892e-01  7.240e-02  -5.376 7.61e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.25 on 213403 degrees of freedom
## Multiple R-squared:  0.02192,    Adjusted R-squared:  0.02189
```

```
## F-statistic: 597.9 on 8 and 213403 DF,  p-value: < 2.2e-16
```

Mirando los coeficientes de la regresión, vemos que todas las variables son muy significativas a la hora de estimar el retraso de los vuelos.

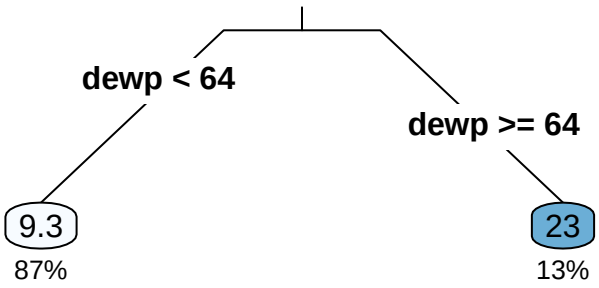
4 Métodos basados en arboles

4.1 Arboles aleatorios

Pasamos ahora a estudiar los metodos basados en arboles y comenzamos con el arbol aleatorio. Lo realizaremos con las mismas variables independientes que hemos escogido anteriormente y ya el arbol eliminará las que no sean necesarias.

```
#Metodo de arboles
tree.vuelos=rpart(dep_delay ~ temp + dewp + humid + wind_dir + wind_speed + precip + pressure + visib,
                  data=entrenamiento)

rpart.plot(tree.vuelos, type = 3, clip.right.labs = FALSE, branch = .3, under = TRUE)
```



Aquí vemos que la única variable que utiliza el arbol para predecir el retraso de los vuelos es la variable "dewp". Veamos ahora el error que comete en la predicción.

```
tree.pred=predict(tree.vuelos,test)
rmse_arbol = with(test,sqrt(MSE(tree.pred,dep_delay)))
```

Vemos que el RMSE del arbol aleatorio es 37.0879455 que es practicamente el mismo que el de la regesión lineal (que era de 36.9489534)

4.2 GBM

Lo primero que haremos será selecciona los parametros de control y los parametros de grid.

```r
# GBM mediante CV

Control <- trainControl(method = 'cv', number = 5, summaryFunction=defaultSummary)

gbm_Grid <-  expand.grid(interaction.depth = c(1,4,6,8),
                         n.trees = c(500,1000,1500),
                         shrinkage = c(.005, .02,.05),
                         n.minobsinnode = 10)
```

Pasamos ahora a realizar boosting con los parametros anteriormente escogidos. Ya que tenemos más de 280000 observaciones, lo que haremos será reducir la muestra para poder realizar los métodos de GBM y RF.

```r
set.seed(12345)
subconjunto = sample(1:nrow(tabla), round(nrow(tabla)*0.05,0), replace = FALSE)
subtabla = tabla[subconjunto, ]
sub_train = sample(1:nrow(subtabla), round(nrow(subtabla)*0.7,0), replace = FALSE)
sub_entrenamiento = tabla[sub_train, ]
sub_test= tabla[-sub_train, ]
```

Se han seleccionado, de manera aleatoria, un 5% de los datos que teniamos inicialmente, que son aproximadamente unas 14000 observaciones.

```r
## GBM
fit.gbm <- train(dep_delay ~ temp + dewp + humid + wind_dir + wind_speed + precip + pressure + visib,
                 data=sub_entrenamiento,
                 method = "gbm",
                 trControl=Control,
                 tuneGrid=gbm_Grid,
                 metric="RMSE",
                 distribution="gaussian")
```

Veamos cual es el mejor modelo obtenido mediante boosting.

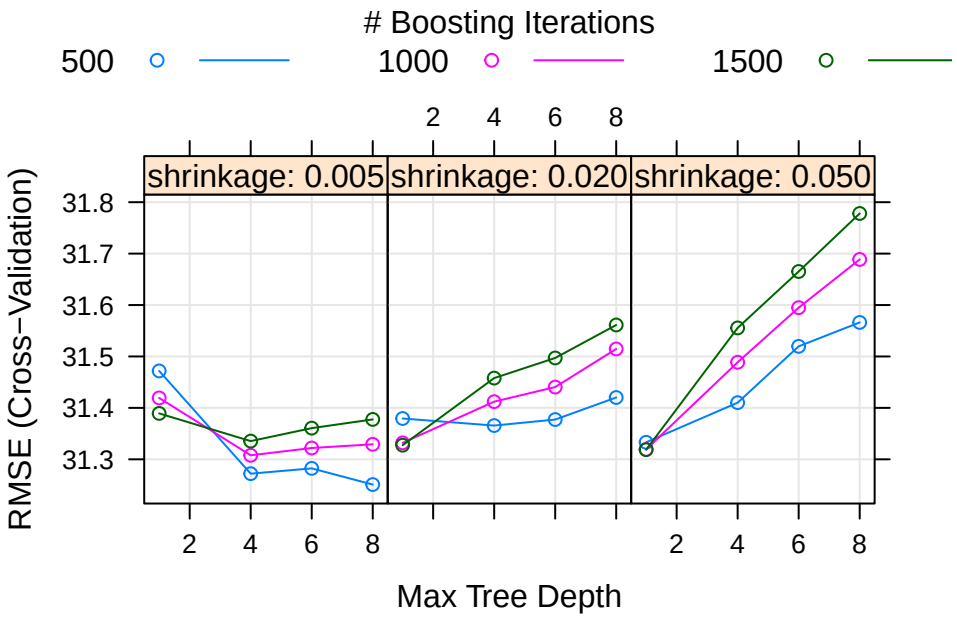
```r
fit.gbm
```

```
## Stochastic Gradient Boosting
##
## 9960 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7967, 7968, 7969, 7968, 7968
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  RMSE      Rsquared    MAE
##   0.005      1                   500     31.47189  0.02423952  15.21247
##   0.005      1                  1000     31.41934  0.02585875  15.12871
##   0.005      1                  1500     31.38925  0.02741319  15.09772
##   0.005      4                   500     31.27210  0.03488800  14.95019
##   0.005      4                  1000     31.30785  0.03467240  14.86998
##   0.005      4                  1500     31.33538  0.03519099  14.84124
##   0.005      6                   500     31.28235  0.03444641  14.90708
##   0.005      6                  1000     31.32200  0.03579464  14.81794
##   0.005      6                  1500     31.36060  0.03627210  14.80291
##   0.005      8                   500     31.25093  0.03687162  14.85703
```

```
##   0.005      8                 1000     31.32919  0.03690841  14.79763
##   0.005      8                 1500     31.37759  0.03718269  14.78003
##   0.020      1                  500     31.37934  0.02795158  15.07424
##   0.020      1                 1000     31.33203  0.03106031  15.02394
##   0.020      1                 1500     31.32743  0.03156162  15.00485
##   0.020      4                  500     31.36565  0.03508265  14.80741
##   0.020      4                 1000     31.41209  0.03649948  14.78672
##   0.020      4                 1500     31.45796  0.03659098  14.80637
##   0.020      6                  500     31.37729  0.03712673  14.79752
##   0.020      6                 1000     31.44052  0.03794692  14.77971
##   0.020      6                 1500     31.49706  0.03752951  14.79305
##   0.020      8                  500     31.42020  0.03715295  14.77835
##   0.020      8                 1000     31.51465  0.03728069  14.79061
##   0.020      8                 1500     31.56120  0.03760939  14.80645
##   0.050      1                  500     31.33314  0.03105297  15.04525
##   0.050      1                 1000     31.31929  0.03249034  14.98709
##   0.050      1                 1500     31.31869  0.03296881  14.97602
##   0.050      4                  500     31.41016  0.03721461  14.80712
##   0.050      4                 1000     31.48871  0.03711723  14.78552
##   0.050      4                 1500     31.55550  0.03715294  14.83081
##   0.050      6                  500     31.51967  0.03665096  14.83671
##   0.050      6                 1000     31.59489  0.03730107  14.85159
##   0.050      6                 1500     31.66520  0.03655403  14.87833
##   0.050      8                  500     31.56619  0.03706648  14.86016
##   0.050      8                 1000     31.68868  0.03668248  14.86192
##   0.050      8                 1500     31.77804  0.03499387  14.91756
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 500,
##  interaction.depth = 8, shrinkage = 0.005 and n.minobsinnode = 10.
```

El modelo que seleccionamos es el que tiene 500 arboles, una profundidad de interacción de 6 y un parametro de contracción de 0.005. Si dibujamos lo obtenido al hacer el boosting tenemos lo siguiente.

```
plot(fit.gbm)
```



Calculamos ahora cual es el RMSE del modelo con los datos de entrenamiento.

```
res_gbm <- fit.gbm$results
rmse_gbm <- subset(res_gbm[5])
# CV con mejor "tune"
rmse_train_gmb = min(rmse_gbm)
# 33.47961
```

Vemos que es de 31.2509328. Pasamos a realizar la predicción y ver su RMSE.

```
boost_pred <- predict(fit.gbm,sub_test)
rmse_test_gbm = sqrt(MSE(y_pred = boost_pred, y_true = sub_test$dep_delay))
#37.74465
```

El RMSE es 38.1352299 que es mayor que el obtenido con la regresión lineal y con el arbol aleatorio. Esto se debe a que hemos reducido el número de observaciones considerablemente.

4.3 Random Forest

El siguiente método que vamos a utilizar es el bosque aleatorio. El parametro "mtry" más grande es 8 ya que es el número máximo de variables que utilizamos para predecir el retraso en los vuelos.

```
#mtry. max = 8 ya que son 8 variables
rf_Grid <-  expand.grid(mtry = c(1,2,3,4,5,6,8))
```

Vamos a realizar el bosque y a comentar los resultados. Para este caso también utilizaremos una muestra reducida del total de los datos.

```
## RANDOM FOREST ##
fit.rf_total <- train(dep_delay ~ temp + dewp + humid + wind_dir + wind_speed + precip + pressure + visi
                      data=sub_entrenamiento,
                      method = "rf",
                      trControl=Control,
                      tuneGrid=rf_Grid,
```

```
                      metric="RMSE",
                      distribution="gaussian")
fit.rf_total
```

```
## Random Forest
##
## 9960 samples
##    8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 7969, 7968, 7967, 7968, 7968
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared    MAE
##   1     30.79892  0.04495236  14.76513
##   2     31.42551  0.03849216  14.97986
##   3     31.78114  0.03523788  15.21063
##   4     31.83390  0.03497807  15.23546
##   5     31.82179  0.03518141  15.24752
##   6     31.85802  0.03463293  15.27097
##   8     31.84646  0.03502246  15.26477
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 1.
```

```
res_rf_total <- fit.rf_total$results
rmse_rf_total <- subset(res_rf_total[2])

# CV con mejor "tune"
rmse_train_rf= min(rmse_rf_total)
#33.28521
```

El mejor bosque es con el parámetro "mtry" igual a 1. El RMSE con los datos de entrenamiento es 30.798917. Vemos que este error es algo menor que el obtenido con los datos de entrenamiento en boosting.

```
rf_pred <- predict(fit.rf_total,sub_test)
rmse_test_rf = sqrt(MSE(y_pred = rf_pred, y_true = sub_test$dep_delay))
#37.78956
```

El RMSE de la predicción es 37.7816861 que es casi igual al obtenido con la predicción de boosting. Este error no es fiable ya que hemos reducido la muestra considerablemente.

5 Conclusión

Por lo que hemos visto, el mejor modelo para predecir el retraso será el del arbol aleatorio ya que su RMSE no es muy diferente al del modelo lineal pero este utiliza menos varibales. De todas formas, no podemos descartar los modelos de boosting y random forest ya que no hemos podido realizarlos con todos los datos. Si comparamos el RMSE de los modelos con la media del retraso en los vuelos, vemos que el RMSE es mucho mayor que la media en los retrasos, por lo que las predicciones no serán muy buenas.