Actividad 1.Modelamiento de la Demanda

Natalia A Clivio V

2014

Introducción

A continuación se detalla la comparación de varios modelos para pronosticar la demanda.

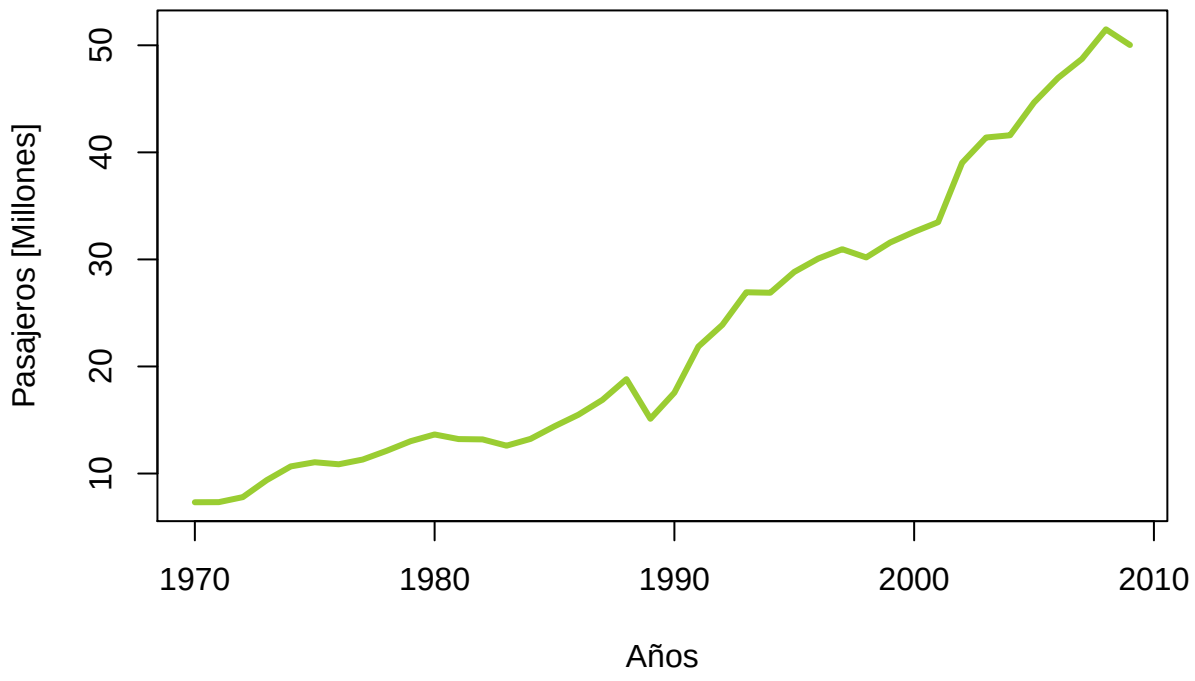
Descripción de los Datos

Se eligieron datos provenientes del paquete **fpp**, del libro Forecasting: Principles and Practice por Rob J Hyndman and George Athanasopoulos. Se trabajo con un dataset con un formato de series de tiempo anual ya que coincide con el formato de la demanda de usuarios requerido para el desarrollo de la tesis.

Preparación de Datos

Es necesario cargar el paquete **fpp**, se trabajará con el data set **ausair** Air Transport Passengers Australia, que contiene el total de pasajeros los años comprendidos entre 1970 y 2009.

```
library(fpp)
data(ausair)

plot(ausair, xlab="Años",ylab="Pasajeros [Millones]",col="yellowgreen",lwd=3)
```

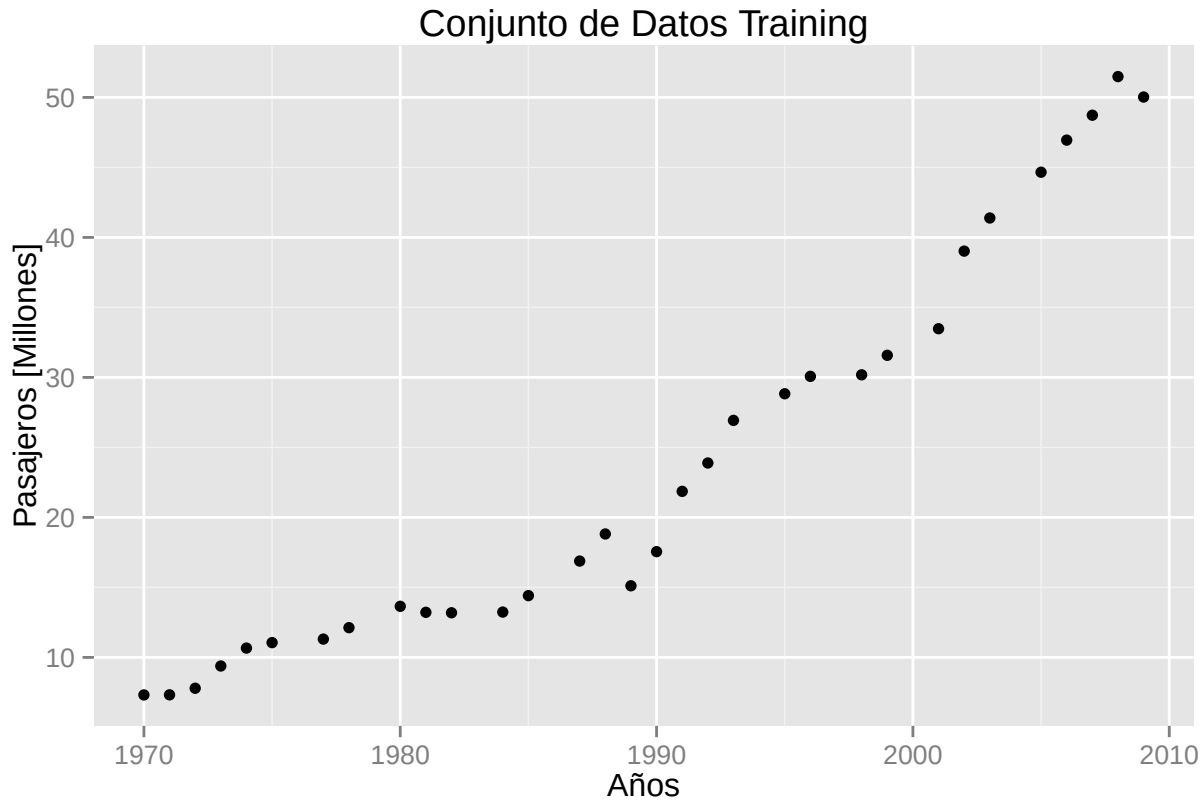


Se submuestrean los datos en dos grupos *training* para aplicar el modelo y *testing* para hacer las predicciones.

```
library(caret)
library(ggplot2)

ts1<-ausair
years<-c(1970:2009)
data<-data.frame(Años=years,Pasajeros=ts1)

inTrain<-createDataPartition(y=data$Años, p=0.75, list=FALSE)
training<-data[inTrain,]
testing<-data[-inTrain,]
```

El conjunto de datos *training*, se muestra a continuación:

```
qplot(Años,Pasajeros,data=training,xlab="Años",ylab="Pasajeros [Millones]",
      main="Conjunto de Datos Training" )
```



Modelos y Predicciones

1.Prediciendo con Modelos Lineales

A continuación se aplica, el modelo de regresión lineal, donde la variable independiente es el tiempo (Años) y la dependiente la cantidad de pasajeros.

```
modglm<-train(Pasajeros~.,data=training,method="glm")

modglm$finalModel
```

```
##
## Call:  NULL
##
## Coefficients:
## (Intercept)         Años
##    -2199.27         1.12
##
## Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
## Null Deviance:       6210
## Residual Deviance: 484   AIC: 184
```

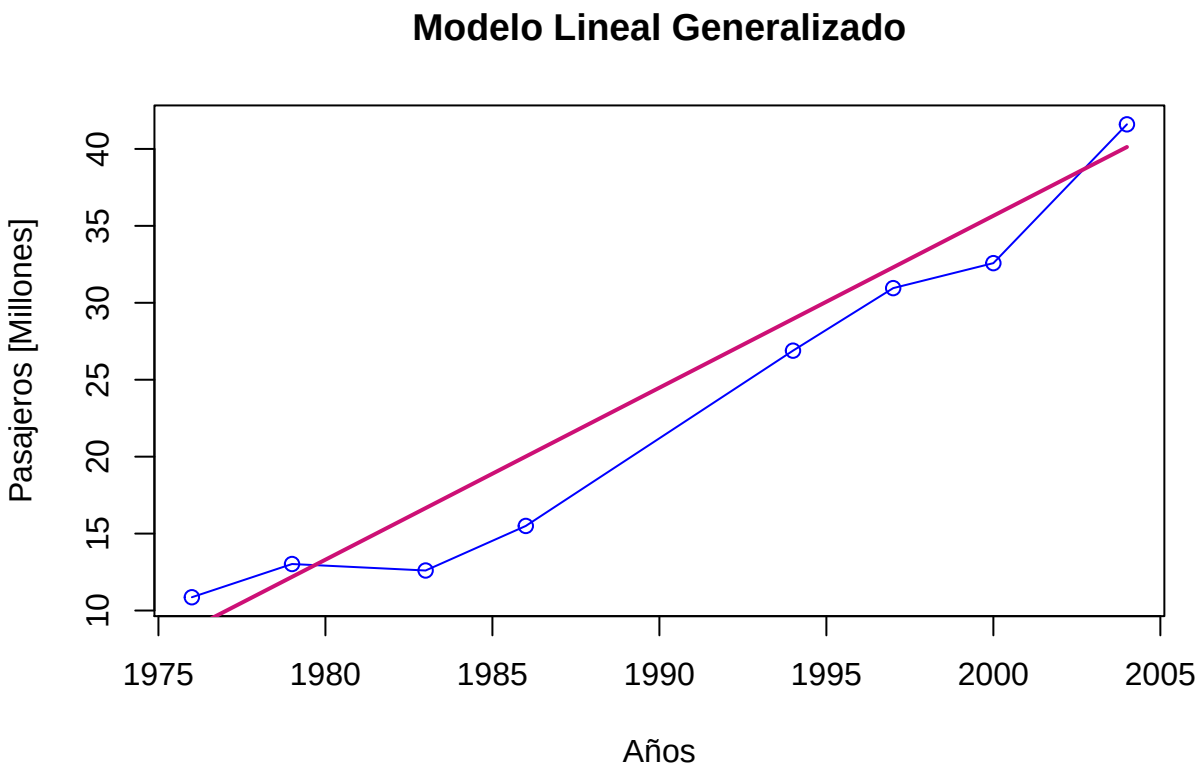
Se hacen las predicciones

```
predglm <- (predict(modglm, testing))
testglm<-data.frame(testing,Predicción_glm=predglm)
testglm
```

```
##    Años Pasajeros Predicción_glm
## 7  1976     10.86          8.836
## 10 1979     13.02         12.188
## 14 1983     12.60         16.658
## 17 1986     15.50         20.010
## 25 1994     26.89         28.950
## 28 1997     30.95         32.302
## 31 2000     32.58         35.655
## 35 2004     41.60         40.125
```

Graficamente la predicción del modelo lineal, se ve asi:

```
plot(testing,xlab="Años",ylab="Pasajeros [Millones]",col="blue",
     main="Modelo Lineal Generalizado")
lines(testing$Años,testing$Pasajeros,lwd=1,col="blue")
lines(testing$Años,predglm,lwd=2,col="deeppink3")
```

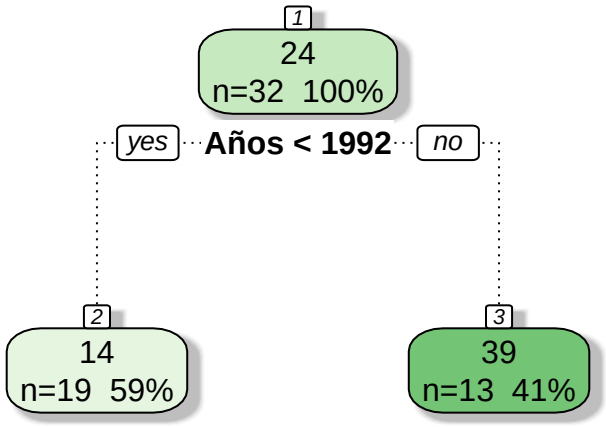


2.Prediciendo con Árboles

Se emplea el algoritmo de áboles de decisión, el cual es mucho mas preciso que los modelos lineales.

```
## n= 32
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 32 6210.0 23.82
##   2) Años< 1992 19  383.8 13.62 *
##   3) Años>=1992 13  964.1 38.72 *
```

Clasificación en Árbol

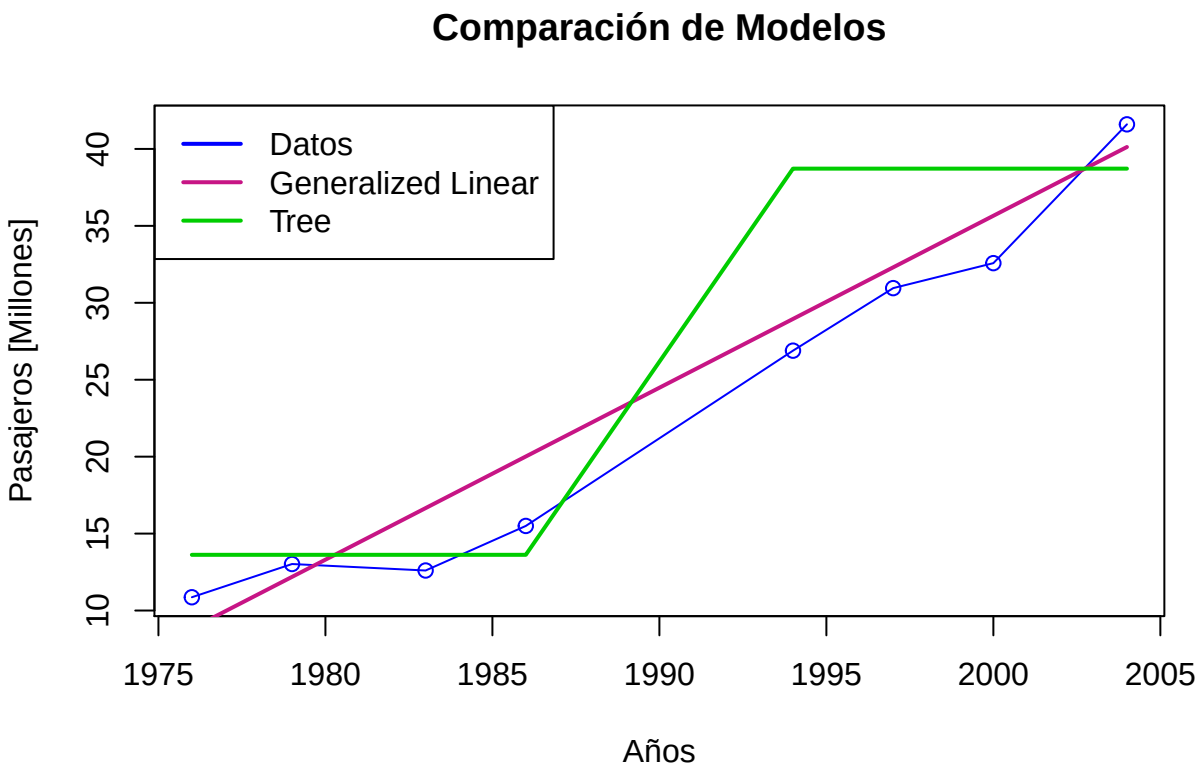


Rattle 2014−oct−25 22:33:41 NataliaA

Se hacen las predicciones

```
predtr <- predict(modtr, newdata=testing)
testtr<-data.frame(testglm,Predicción_tree=predtr)
testtr
```

```
##    Años Pasajeros Predicción_glm Predicción_tree
## 7  1976     10.86          8.836           13.62
## 10 1979     13.02         12.188           13.62
## 14 1983     12.60         16.658           13.62
## 17 1986     15.50         20.010           13.62
## 25 1994     26.89         28.950           38.72
## 28 1997     30.95         32.302           38.72
## 31 2000     32.58         35.655           38.72
## 35 2004     41.60         40.125           38.72
```



3. Prediciendo con "Random Forest"

Este modelo es más robusto, pero aumenta su velocidad de procesamiento, es más dificil de interpretar y puede ocasionar sobreajustes.

```
modrf<-train(Pasajeros~.,data=training,method="rf",prox=TRUE)
modrf
```

```
## Random Forest
##
## 32 samples
##  1 predictors
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 32, 32, 32, 32, 32, 32, ...
##
## Resampling results
##
##   RMSE  Rsquared  RMSE SD  Rsquared SD
##   2     1         0.5      0.02
##
## Tuning parameter 'mtry' was held constant at a value of 2
##
```

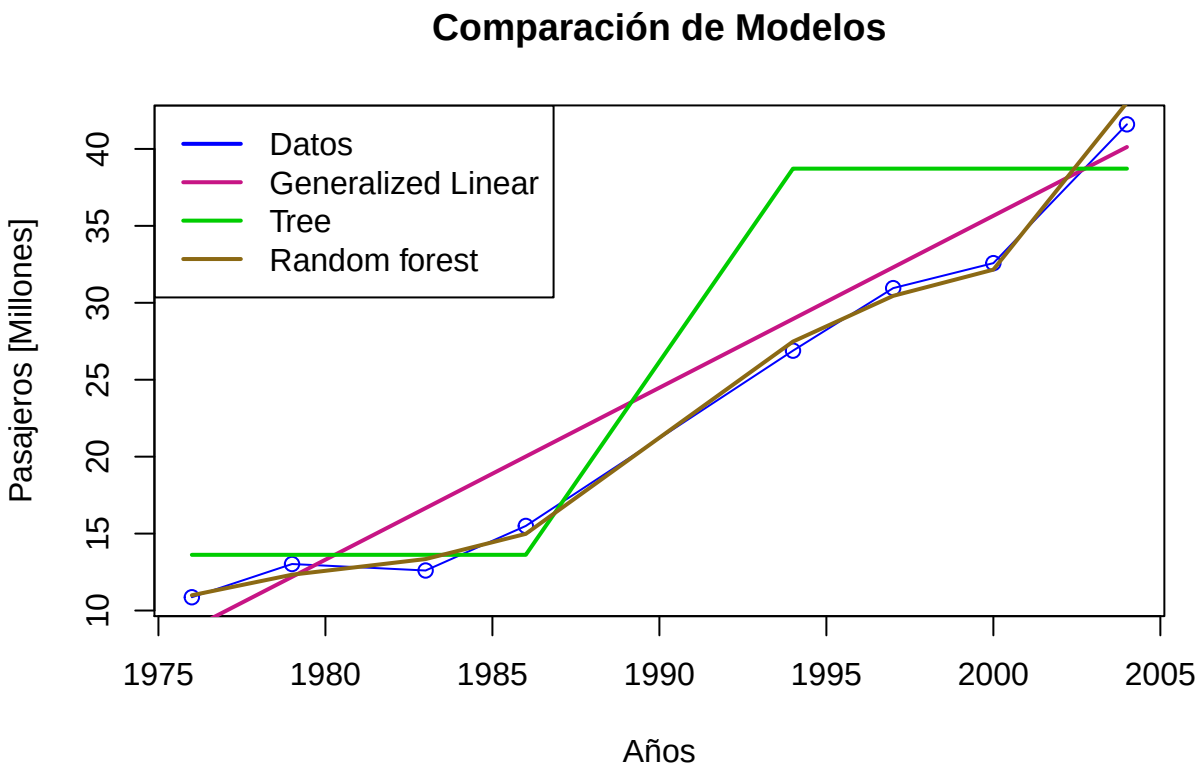
```
modrf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 4.117
##                     % Var explained: 97.88
```

Prediciendo valores

```
predrf <- predict(modrf, testing)
testrf<-data.frame(testtr,Predicción_rf=predrf)
testrf
```

```
##    Años Pasajeros Predicción_glm Predicción_tree Predicción_rf
## 7  1976     10.86          8.836           13.62         10.99
## 10 1979     13.02         12.188           13.62         12.33
## 14 1983     12.60         16.658           13.62         13.34
## 17 1986     15.50         20.010           13.62         14.98
## 25 1994     26.89         28.950           38.72         27.48
## 28 1997     30.95         32.302           38.72         30.45
## 31 2000     32.58         35.655           38.72         32.15
## 35 2004     41.60         40.125           38.72         43.01
```



4. Prediciendo con "Boosting"

La motivación para el algoritmo *Boosting* es un procedimiento que combina las salidas de muchos clasificadores "débiles" para producir un "comité" poderoso. Desde esta perspectiva *Boosting* tiene un parecido al *Bagging* y otros enfoques basados en los comités.

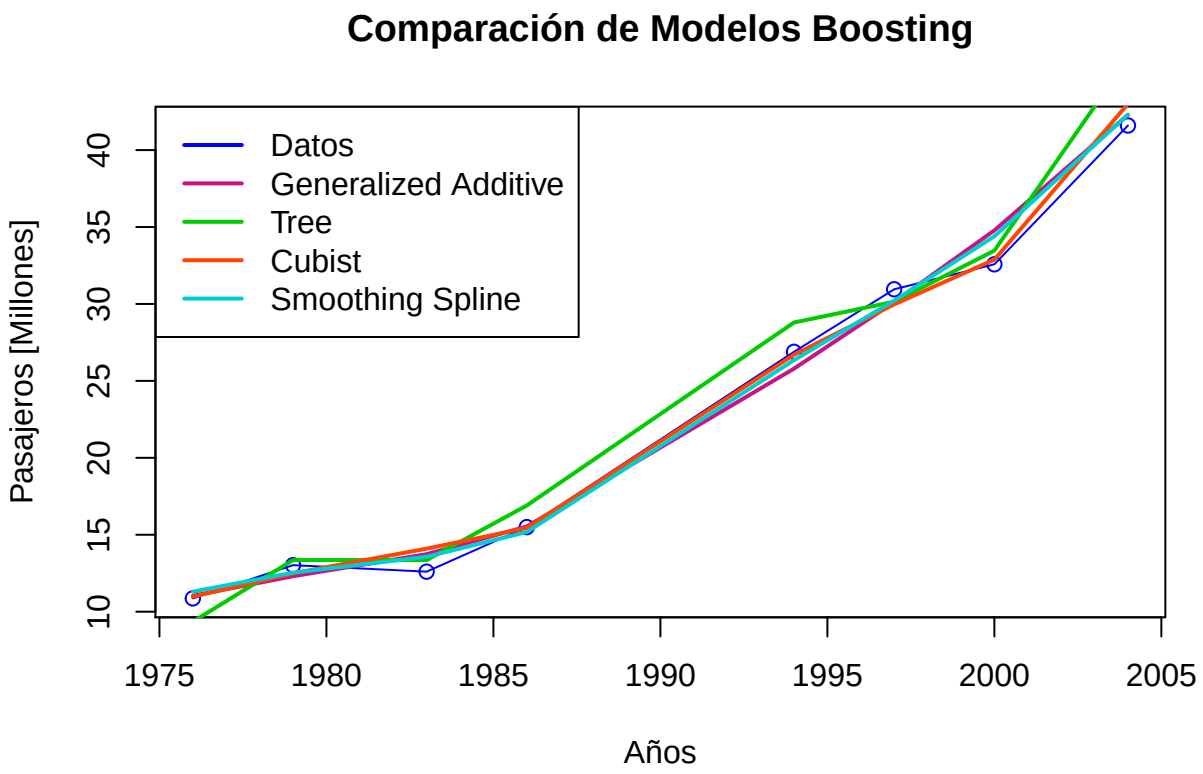
Se basa en predictores débiles y debilidad de los *learners* juegan un papel importante en las técnicas de *Bagging* y *Boosting* que sólo ahora están haciendo su camino en la previsión y análisis de negocio, aunque la comunidad de *machine learning*.

Basandose en http://topepo.github.io/caret/Boosting.html

```
#Boosted Generalized Additive Model
modbsgam <- train(Pasajeros ~ ., method = "gamboost", data = training)
predbsgam <- predict(modbsgam, testing)

#Boosted Tree
modbstt <- train(Pasajeros ~ ., method = "bstTree", data = training)
predbstt <- predict(modbstt, testing)

#Cubist
modbscub <- train(Pasajeros ~ ., method = "cubist", data = training)
predbscub <- predict(modbscub, testing)

#Boosted Smoothing Spline
```

```
modbsSm <- train(Pasajeros ~ ., method = "bstSm", data = training)
predbsSm <- predict(modbsSm, testing)
```

Graficamente las predicciones usando los algoritmos de Boosting



Las predicciones de los anteriores modelos son:

```
testbs<-data.frame(testrf,Pred_Bs_gam=predbsgam,Pred_Bs_cub=predbscub,
                   Pred_Bs_stt=predbstt,Pred_Bs_Sm=predbsSm)
testbs
```

```
##    Años Pasajeros Predicción_glm Predicción_tree Predicción_rf Pred_Bs_gam
## 7  1976     10.86          8.836           13.62         10.99       11.04
## 10 1979     13.02         12.188           13.62         12.33       12.31
## 14 1983     12.60         16.658           13.62         13.34       13.74
## 17 1986     15.50         20.010           13.62         14.98       15.54
## 25 1994     26.89         28.950           38.72         27.48       25.79
## 28 1997     30.95         32.302           38.72         30.45       30.12
## 31 2000     32.58         35.655           38.72         32.15       34.78
## 35 2004     41.60         40.125           38.72         43.01       42.29
##    Pred_Bs_cub Pred_Bs_stt Pred_Bs_Sm
## 7        10.98       9.351      11.30
## 10       12.49      13.356      12.54
## 14       14.10      13.356      13.57
## 17       15.44      16.903      15.18
## 25       26.70      28.795      26.35
```

```
## 28       29.97      30.147      30.21
## 31       32.88      33.468      34.40
## 35       43.00      45.946      42.30
```

Desempeño de los modelos

Para evaluar los modelos, se calcula:

Error medio **(ME)**:Mean Error

Error cuadrado medio **(RMSE)**:Root Mean Square Error

Desviación adboluta media **(MAE)**: Mean Absolute Error

Error porcentual medio **(MPE)**: Mean Porcentual Error

Error porcentual absoluto medio **(MAPE)**: Mean Absolute Porcentual Error

```
library(forecast)

acc_a<-accuracy(predglm,testrf$Pasajeros, test=NULL)
acc_b<-accuracy(predtr,testrf$Pasajeros, test=NULL)
acc_c<-accuracy(predrf,testrf$Pasajeros, test=NULL)
acc_d<-accuracy(predbsgam,testrf$Pasajeros, test=NULL)
acc_e<-accuracy(predbstt,testrf$Pasajeros, test=NULL)
acc_f<-accuracy(predbscub,testrf$Pasajeros, test=NULL)
acc_g<-accuracy(predbsSm,testrf$Pasajeros, test=NULL)

acc_all<-rbind(acc_a,acc_b,acc_c,acc_d,acc_e,acc_f,acc_g)
models<-c("Generalized Linear","Tree","Random forest", "Bs_Generalized Additive"
          ,"Bs_Tree","Bs_Cubist","Bs_Smoothing Spline")

perf<-data.frame(models,acc_all)
perf
```

```
##                     models       ME   RMSE    MAE      MPE   MAPE
## 1      Generalized Linear -1.34016 2.7234 2.4240  -6.7698 13.925
## 2                    Tree -3.16928 5.6872 4.3580 -13.3689 18.126
## 3           Random forest -0.09221 0.7153 0.6264  -0.1354  3.032
## 4 Bs_Generalized Additive -0.20051 1.0641 0.8609  -0.8798  3.944
## 5                 Bs_Tree -0.91486 1.9021 1.4950  -2.6694  6.804
## 6               Bs_Cubist -0.19461 0.8363 0.6343  -1.1115  3.190
## 7     Bs_Smoothing Spline -0.23138 0.8741 0.7520  -1.1051  3.646
```