Universidad Rafael Landívar
Segundo Ciclo 2021
Facultad de Ingeniería
Análisis de Datos
Ing. Dan Stanly Bolaños

Documentación Técnica Examen Final

Andrés Gálvez [1024718]
Alexander Villatoro [1182118]
Luis Chuta [1320016]
Sergio Lara [1044418]

Ciudad de Guatemala, Martes 23 de Noviembre de 2021

Contenido

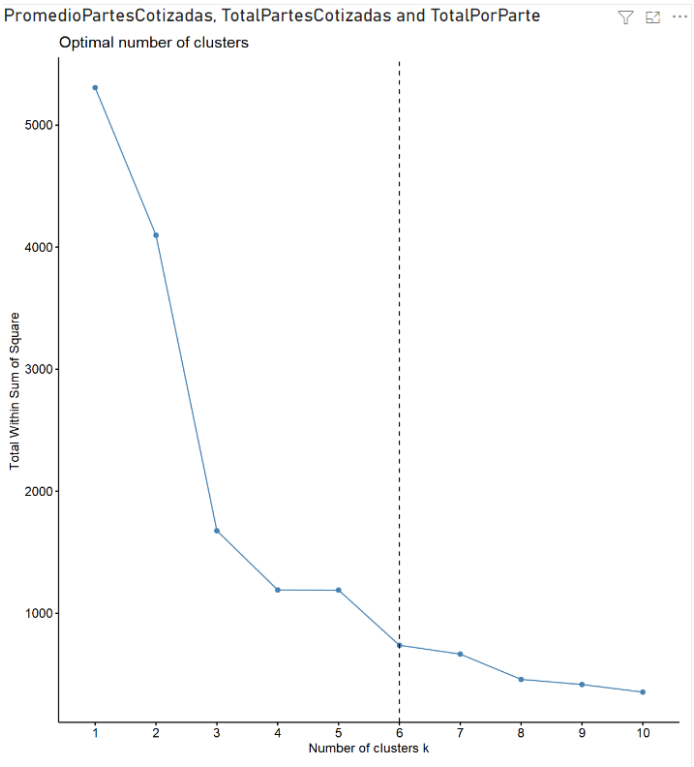
Descripción de objetos

Objetos visuales de PowerBI:

Para la presentación de los gráficos requeridos por el enunciado del proyecto, estos se hicieron en base a los laboratorios hechos en clase. Así que, para presentar, los resultados de nuestros modelos de predicción, clasificación, agrupamiento y líneas de tiempo para los cuales se crearon las siguientes graficas.

Clustering

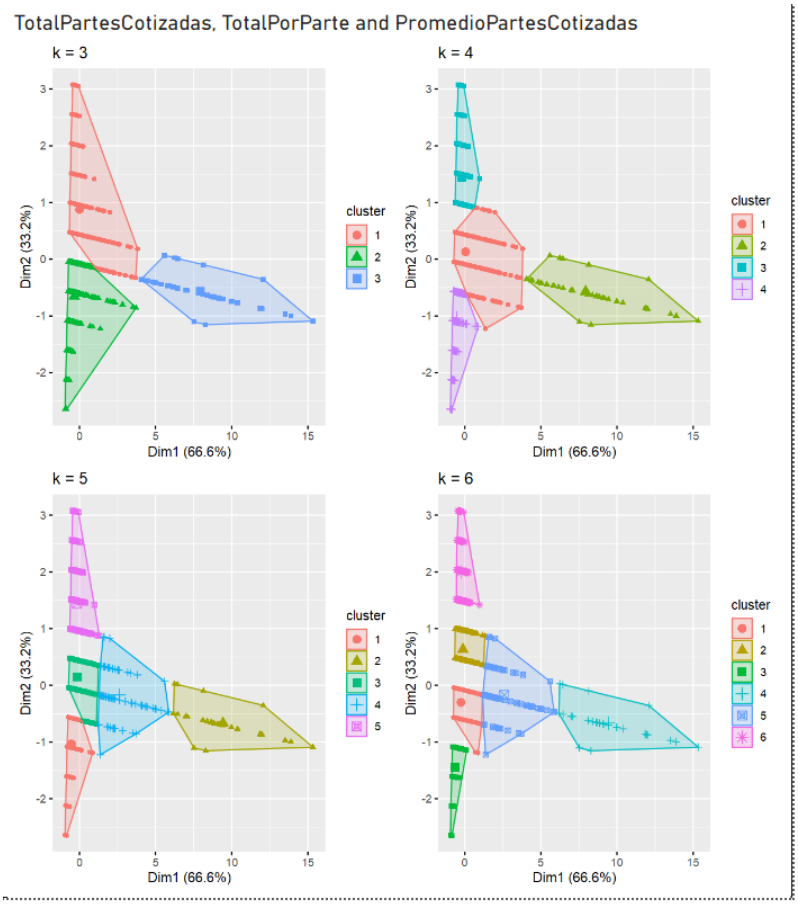
Gráfica WSS



Código

```r
# The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:

# dataset <- data.frame(PromedioPartesCotizadas, TotalPartesCotizadas, TotalPorParte)
# dataset <- unique(dataset)

# Paste or type your script code here:


library(ggplot2)
library(dplyr)
library(DBI)
library(odbc)

rownames(dataset) <- dataset$Nombre
Repuestos <- scale(dataset)
Repuestos <- na.omit(Repuestos) #quitar los registros que estan en null


#install.packages("factoextra") #paquete para graficar
library(factoextra) #libreria de paquete instalado


distancia <- get_dist(Repuestos)
set.seed(123) #seed permitira fijar un id para generar valores random

fviz_nbclust(Repuestos, kmeans, method = "wss") +
  geom_vline(xintercept = 6, linetype = 2)
```

Gráfica de K-means:

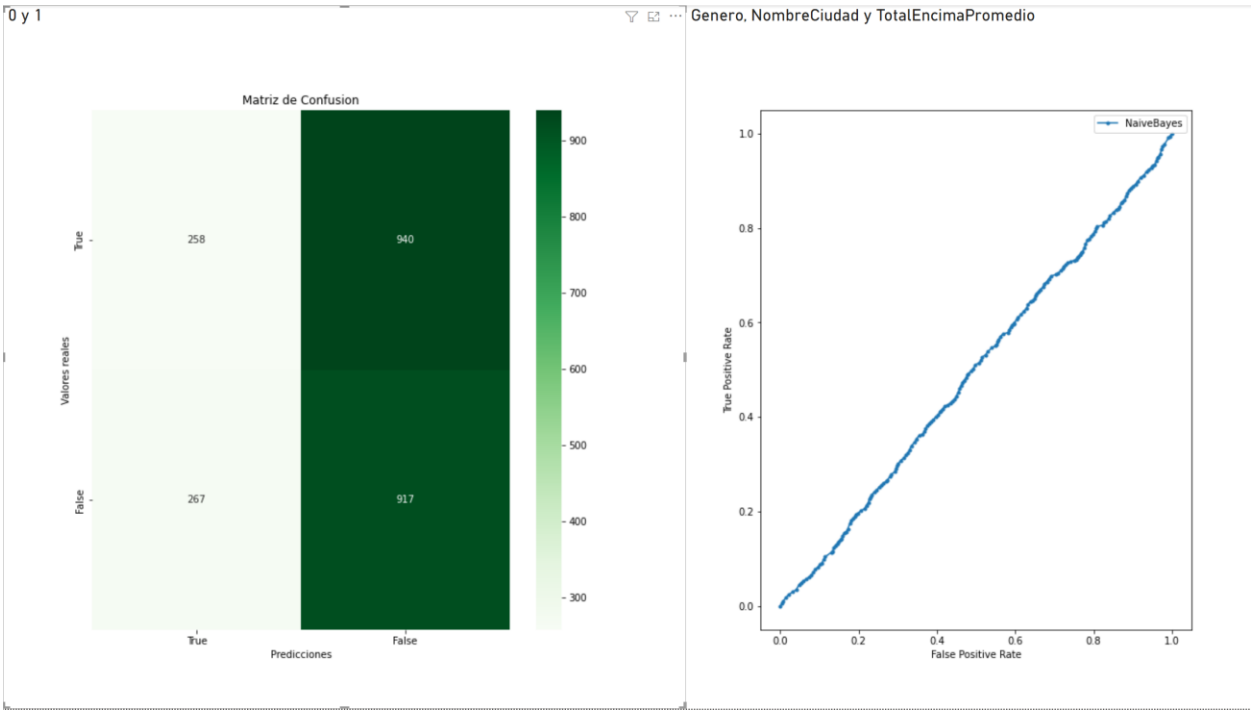


Codigo K-means

```r
# The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:

# dataset <- data.frame(PromedioPartesCotizadas, TotalPartesCotizadas, TotalPorParte)
# dataset <- unique(dataset)

# Paste or type your script code here:


library(ggplot2)
library(dplyr)
library(DBI)
library(odbc)

rownames(dataset) <- dataset$Nombre
Repuestos <- scale(dataset)
Repuestos <- na.omit(Repuestos) #quitar los registros que estan en null


#install.packages("factoextra") #paquete para graficar
library(factoextra) #libreria de paquete instalado


distancia <- get_dist(Repuestos)
set.seed(123) #seed permitira fijar un id para generar valores random


clusterk3 <- kmeans(Repuestos, 3, nstart = 25)
clusterk4 <- kmeans(Repuestos, 4, nstart = 25)
clusterk5 <- kmeans(Repuestos, 5, nstart = 25)
clusterk6 <- kmeans(Repuestos, 6, nstart = 25)
clusterk7 <- kmeans(Repuestos, 7, nstart = 25)

#grafica1 <- fviz_cluster(clusterk2, geom = "point", data = Repuestos) + ggtitle("k = 2")
grafica2 <- fviz_cluster(clusterk3, geom = "point",  data = Repuestos) + ggtitle("k = 3")
grafica3 <- fviz_cluster(clusterk4, geom = "point",  data = Repuestos) + ggtitle("k = 4")
grafica4 <- fviz_cluster(clusterk5, geom = "point",  data = Repuestos) + ggtitle("k = 5")
grafica5 <- fviz_cluster(clusterk6, geom = "point",  data = Repuestos) + ggtitle("k = 6")
#grafica6 <- fviz_cluster(clusterk7, geom = "point",  data = Repuestos) + ggtitle("k = 7")

library(gridExtra)
grid.arrange(grafica2, grafica3, grafica4, grafica5,nrow = 2)
```

Modelos de Clasificación:

Gráfica:



Código Heatmap:

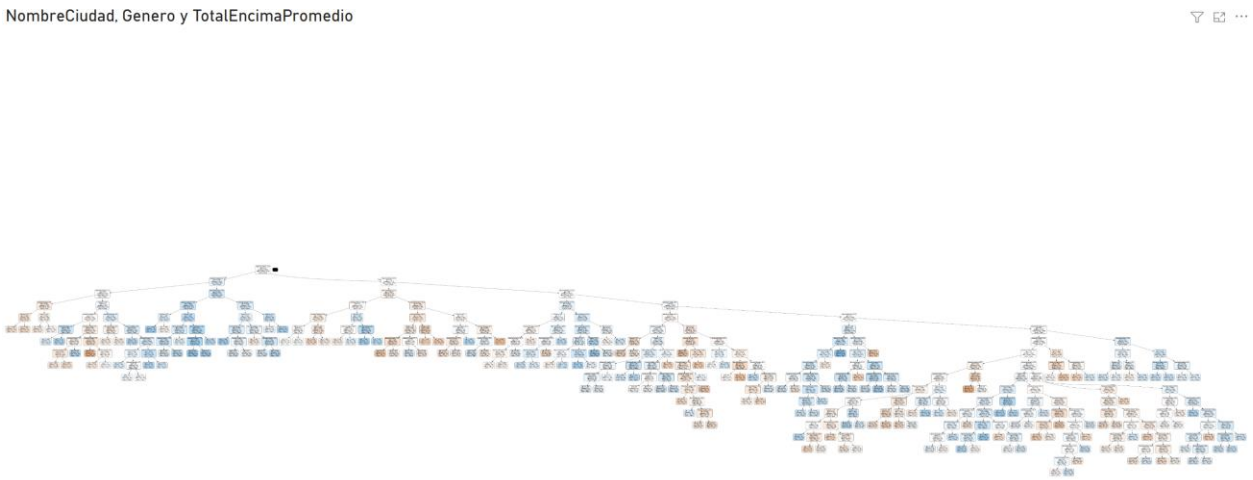
```
Editor de scripts de Python
⚠ Las filas duplicadas se quitarán de los datos.
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(0, 1)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(dataset, annot=True, fmt='g', ax=ax,cmap='Greens');  #annot=True to annotate cells, ftm='g' to disable scientific notation

ax.set_xlabel('Predicciones');ax.set_ylabel('Valores reales');
ax.set_title('Matriz de Confusion');
ax.xaxis.set_ticklabels(['True', 'False']); ax.yaxis.set_ticklabels(['True', 'False']);

plt.show()
```

Código de Gráfica ROC-AUC

```
Editor de scripts de Python
⚠ Las filas duplicadas se quitarán de los datos.
# Pegue o escriba aquí el código de script:
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
from sklearn.preprocessing import LabelEncoder
import os
import pyodbc
conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-7ECACBA;DATABASE=RepuestosWeb;Trusted_Connection=yes;')

query = "select * from VW_OrdenesEncimaPromedio ;"
dataset = pd.read_sql(query, conn)
print(dataset.head(26))
dataset.drop('ID_Parte',axis=1,inplace=True)
dataset.drop('ID_Categoria',axis=1,inplace=True)
dataset.drop('Total_Orden',axis=1,inplace=True)
X = dataset.iloc[:, 0:2].values
y = dataset.iloc[:, -1].values
X = dataset.iloc[:, 0:2].values
y = dataset.iloc[:, -1].values

le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

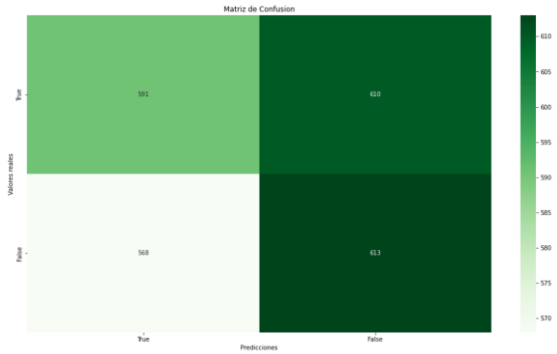
Árbol de decisión:

Gráfica:



Código:

Heatmap del Árbol:



Código del mapa de calor:

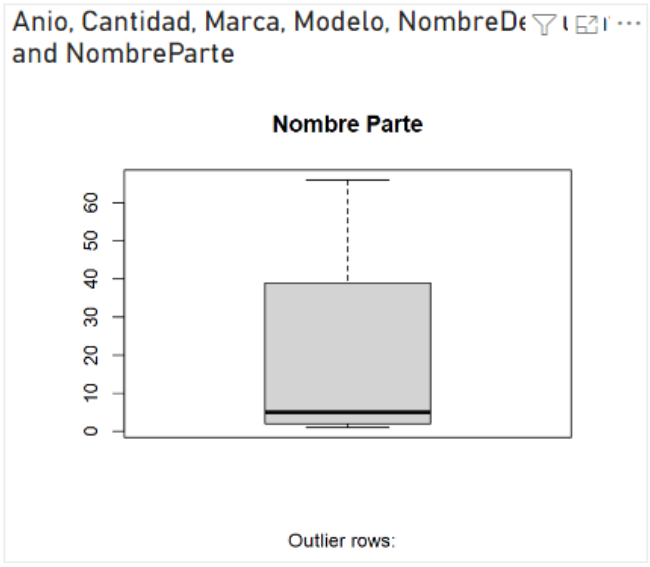
Editor de scripts de Python

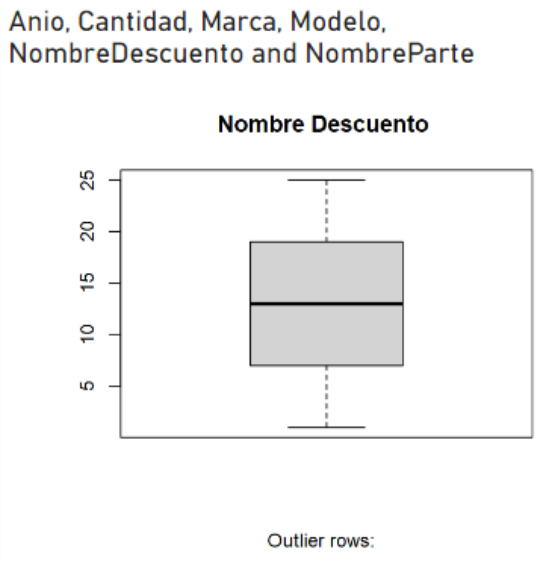
⚠ Las filas duplicadas se quitarán de los datos.

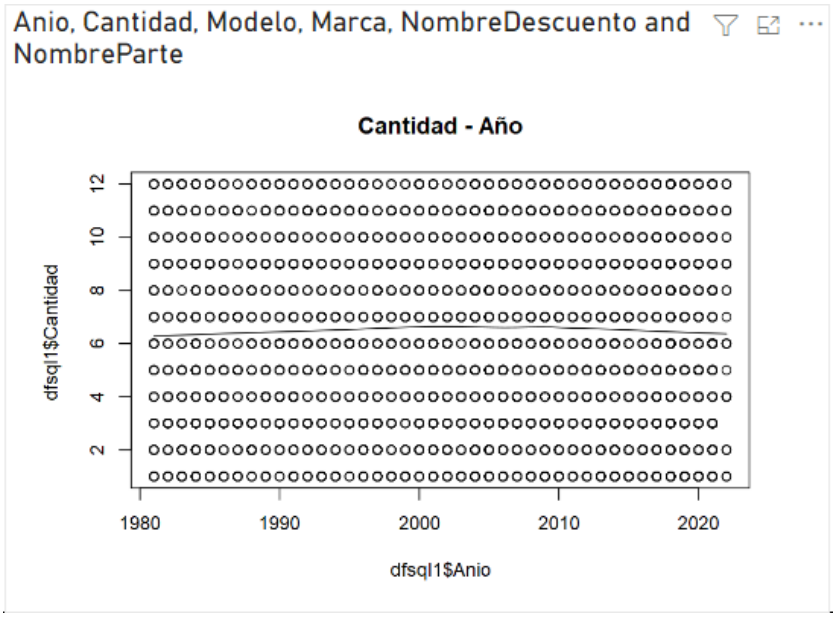
```
# Pegue o escriba aquí el código de script:
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
from sklearn.preprocessing import LabelEncoder
import os
import pyodbc


conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=DESKTOP-7ECAC8A;DATABASE=RepuestosWeb;Trusted_Connection=yes;')

query = "select * from VW_OrdenesEncimaPromedio ;"
df = pd.read_sql(query, conn)
print(df.head(26))


df.drop('ID_Parte',axis=1,inplace=True)
df.drop('ID_Categoria',axis=1,inplace=True)
df.drop('Total_Orden',axis=1,inplace=True)
df


```

Regresión Lineal

Gráficas:







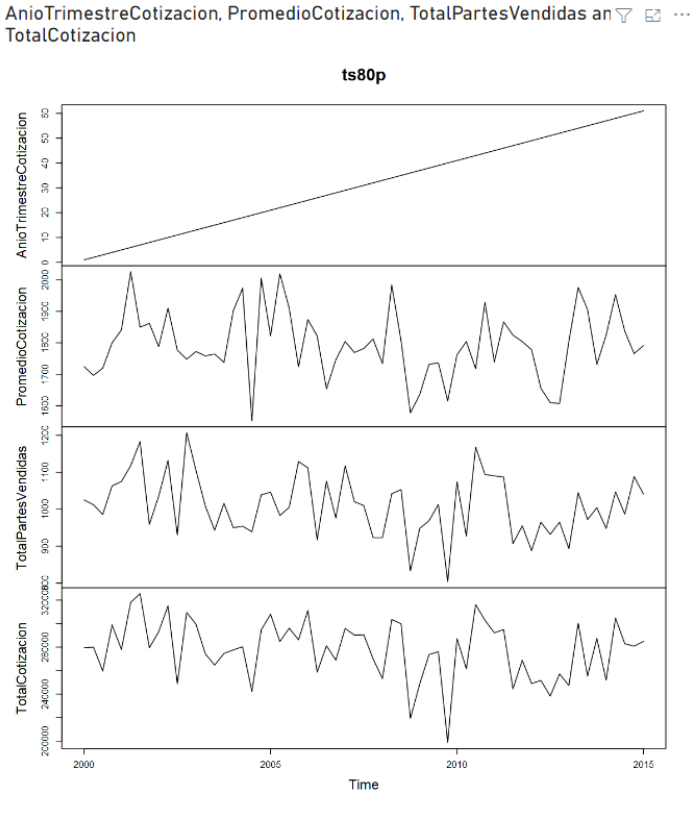
Codigo

```
dfsql <- na.omit(dataset)

dfsqlindex <- sample(1:nrow(dfsql), 0.8*nrow(dfsql))

dfsql1 <- dfsql[dfsqlindex,] #80%
dfsql2 <- dfsql[-dfsqlindex,] #20%


for (i in c("NombreParte","NombreDescuento","Marca","Modelo")) {
  dfsql1[,i]=as.factor(dfsql1[,i])
}


for (i in c("NombreParte","NombreDescuento","Marca","Modelo")) {
  dfsql1[,i]=as.numeric(dfsql1[,i])
}


lmHeight = lm(Cantidad~Anio+Marca+Modelo, data = dfsql1) #Create the linear regression

lmHeight2 = lm(Cantidad~NombreParte, data = dfsql1) #Create the linear regression

lmHeight3 = lm(Cantidad~NombreDescuento, data = dfsql1) #Create the linear regression

boxplot(dfsql1$NombreParte, main="Nombre Parte", sub=paste("Outlier rows: ", boxplot.stats(dfsql1$NombreParte)$out))

boxplot(dfsql1$NombreDescuento, main="Nombre Descuento ", sub=paste("Outlier rows: ", boxplot.stats(dfsql1$NombreDescuento)$out))

boxplot(dfsql1$Marca, main="Marca", sub=paste("Outlier rows: ", boxplot.stats(dfsql1$Marca)$out))
```


```
dfsql <- na.omit(dataset)

dfsqlindex <- sample(1:nrow(dfsql), 0.8*nrow(dfsql))

dfsql1 <- dfsql[dfsqlindex,] #80%
dfsql2 <- dfsql[-dfsqlindex,] #20%


for (i in c("NombreParte","NombreDescuento","Marca","Modelo")) {
  dfsql1[,i]=as.factor(dfsql1[,i])
}


for (i in c("NombreParte","NombreDescuento","Marca","Modelo")) {
  dfsql1[,i]=as.numeric(dfsql1[,i])
}


lmHeight = lm(Cantidad~Anio+Marca+Modelo, data = dfsql1) #Create the linear regression

lmHeight2 = lm(Cantidad~NombreParte, data = dfsql1) #Create the linear regression

lmHeight3 = lm(Cantidad~NombreDescuento, data = dfsql1) #Create the linear regression

scatter.smooth(y=dfsql1$Cantidad, x=dfsql1$Anio, main = "Cantidad - Año")
```

Lineas de Tiempo:

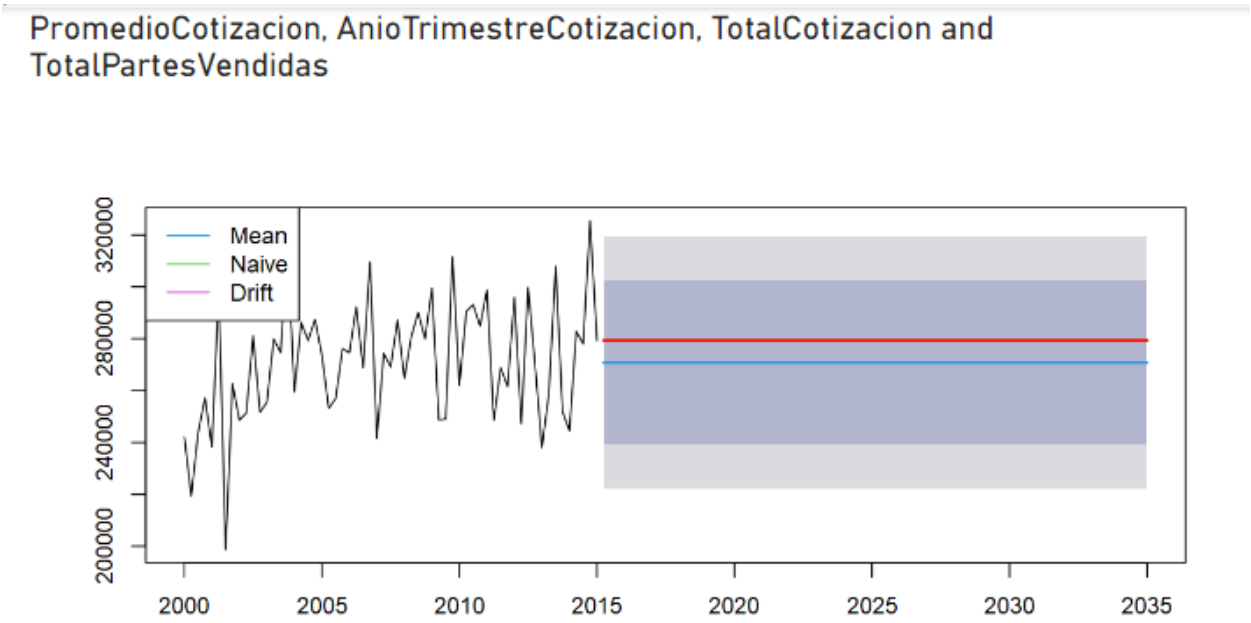
Grafica



Codigo

```
library(ggplot2)

library(forecast)

ts1 <- ts(dataset, start = c(2000,1), frequency = 4)

# la funcion window nos permite dividir un objeto "ts" de un inicio a un fin
ts80p <- window(ts1, start = 2000, end = 2015)
plot(ts80p)
```

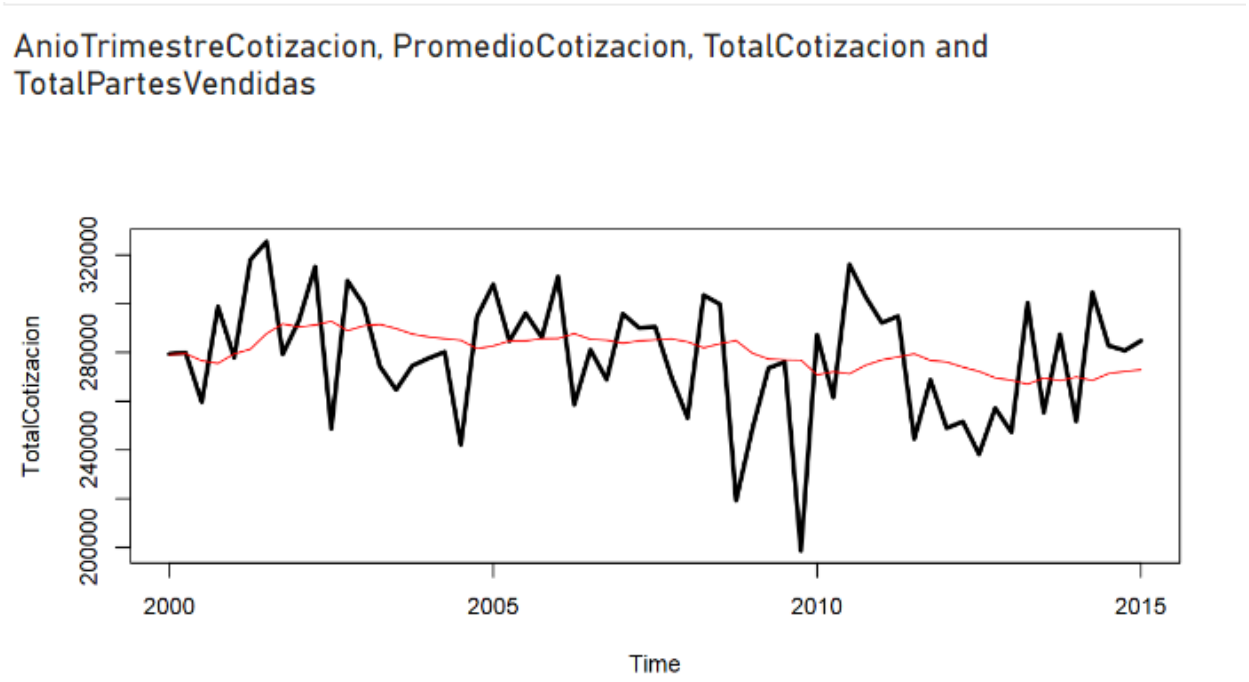
Grafica predicciones:



Codigo Predicciones:

```
library(ggplot2)

library(forecast)

library(ggplot2)
library(dplyr)
library(DBI)
library(odbc)

df1 <- dataset %>% select(TotalCotizacion)

ts1 <- ts(df1, start = c(2000,1), frequency = 4)
ts180p <- window(ts1, start = 2000, end = 2015)
meanmodel <- meanf(ts180p,h=80)
naivemodel <- naive(ts180p, h=80)
driftmodel <- rwf(ts180p, h=80)

plot(meanmodel, plot.conf = F, main = "")
lines(naivemodel$mean, col=123, lwd = 2)
lines(driftmodel$mean, col='red', lwd = 2)
legend("topleft",lty=1,col=c(4,123,22),
       legend=c("Mean","Naive","Drift"))
```

Gráfica predicción ajustando al valor real:



Codigo predicción:

```r
library(ggplot2)

library(forecast)

library(ggplot2)
library(dplyr)
library(DBI)
library(odbc)

df1 <- dataset %>% select(TotalCotizacion)

ts1 <- ts(df1, start = c(2000,1), frequency = 4)
ts180p <- window(ts1, start = 2000, end = 2015)

meanmodel3 <- meanf(ts180p,h=80)
naivemodel3 <- naive(ts180p, h=80)
driftmodel3 <- rwf(ts180p, h=80)


acf(ts180p, lag.max = 20)
pacf(ts180p, lag.max = 20)


auto.arima(ts180p)
arima(ts180p)

myar = auto.arima(ts180p, stepwise = F, approximation = F)

myar

# Ploteamos vs original
plot(ts180p, lwd = 3)
lines(myar$fitted, col = "red")
```