Regresión de árboles de decisión y bosques aleatorios

Porras Braulio, Tinoco Sergio

Instituto Politécnico Nacional, Escuela Superior de Cómputo

Aprendizaje de Máquina 5BV1

18 de Diciembre de 2022

I. Introducción

La regresión de árboles de decisión es un método de aprendizaje automático que se utiliza para realizar predicciones de una variable continua. Se basa en el uso de árboles de decisión, que son diagramas que representan decisiones y sus posibles consecuencias en forma de ramificaciones. Los árboles de decisión son construidos a partir de un conjunto de datos de entrenamiento y suelen ser utilizados para clasificación o predicción.

Un bosque aleatorio es un conjunto de árboles de decisión entrenados y utilizados conjuntamente para hacer predicciones. Los árboles del bosque se construyen de manera independiente, y luego sus predicciones se combinan para obtener un resultado final. Al construir múltiples árboles de decisión y combinar sus predicciones, los bosques aleatorios pueden reducir la variabilidad y mejorar la precisión de las predicciones.

En esta práctica se implementará el algoritmo de regresión de árboles de decisión y bosques aleatorios para un conjunto de datos en los lenguajes de programación Python y R.R.

II. Objetivo

Comprender la implementación de las técnicas de regresión polinomial en R y Python.

III. Marco Teórico

Para esta práctica se va a trabajar con un dataset con el fin poder aplicar una regresión de árboles de decisión y bosques aleatorios, buscando obtener un modelo que prediga el comportamiento de los datos.

IV. Desarrollo

Implementación en Python.

Árboles de decisión

Para comenzar se importan las librerías básicas numpy, matplotlib y sklearn para importar el dataset y generar el modelo .

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

Posterior a esto cargamos el conjunto de datos.

```
boston = datasets.load_boston()
```

Se separan la variable dependiente e independiente.

```
X_bar = boston.data[:, np.newaxis, 5]
y_bar = boston.target
```

Se separan los datos en conjunto de entrenamiento y pruebas.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_bar, y_bar,
test_size=0.2)
```

Se importa el modelo para generar el árbol de decisión.

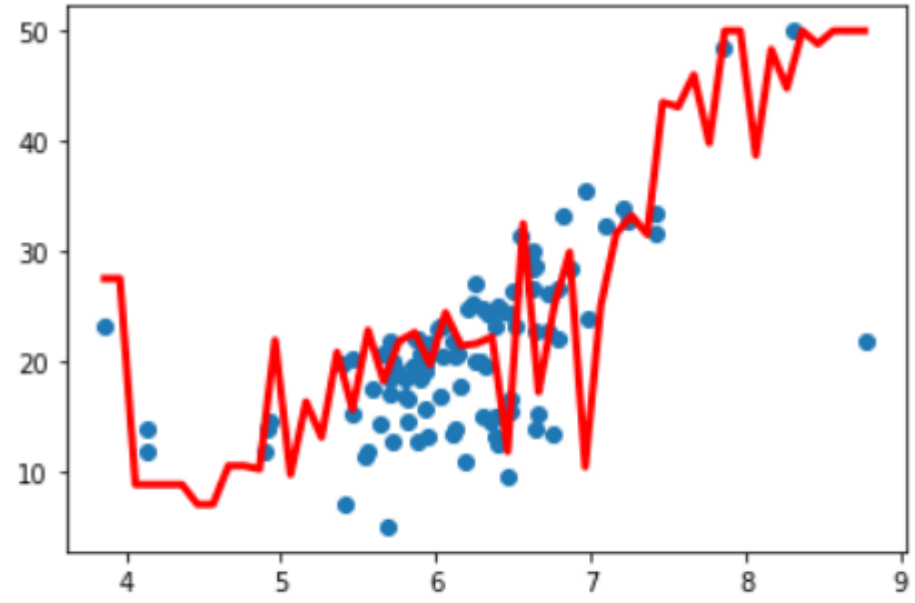
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
```

Se entrena el modelo y se predicen los resultados usando el conjunto de pruebas.

```
regressor.fit(X_train, y_train)
regressor.predict(X_test)
```

Finalmente, graficamos el modelo generado y calculamos su precisión.

```
X_grid = np.arange(min(X_test), max(X_test), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, y_test)
plt.plot(X_grid, regressor.predict(X_grid), color='red', linewidth=3)
plt.show()

print('Precisión del modelo:')
print(regressor.score(X_train, y_train))
```




```
Precisión del modelo:
0.9554052869758145
```

Bosques aleatorios

Para comenzar se importan las librerías básicas numpy, matplotlib y sklearn para importar el dataset y generar el modelo .

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

Posterior a esto cargamos el conjunto de datos.

```
boston = datasets.load_boston()
```

Se separan la variable dependiente e independiente.

```
X_bar = boston.data[:, np.newaxis, 5]
y_bar = boston.target
```

Se separan los datos en conjunto de entrenamiento y pruebas.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_bar, y_bar,
test_size=0.2)
```

Se importa el modelo para generar el bosque aleatorio creando distintos árboles de decisión, la cantidad de estimadores se inicializa en 300 y la profundidad en 8 para los árboles de decisión.

```
from sklearn.ensemble import RandomForestRegressor
```

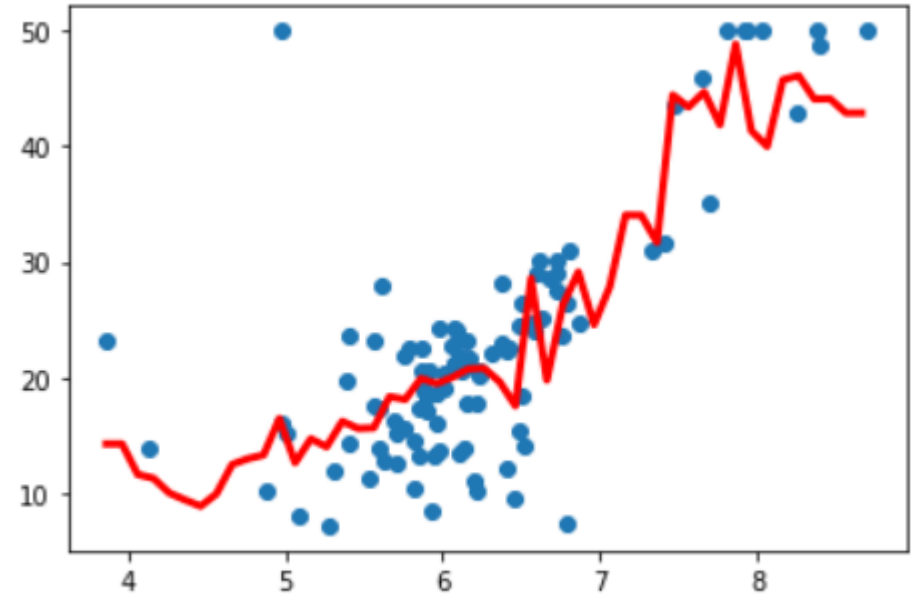
```
bar = RandomForestRegressor(n_estimators = 300, max_depth = 8)
```

Se entrena el modelo y se predicen los resultados usando el conjunto de pruebas.

```
bar.fit(X_train, y_train)
Y_pred = bar.predict(X_test)
```

Finalmente, graficamos el modelo generado y calculamos su precisión.

```
X_grid = np.arange(min(X_test), max(X_test), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, y_test)
plt.plot(X_grid, bar.predict(X_grid), color='red', linewidth=3)
plt.show()

print('Precisión del modelo:')
print(bar.score(X_train, y_train))
```




```
Precisión del modelo:
0.7939916880197722
```

Implementación en R

- Árboles de regresión

Primeramente se carga el dataset de Boston y se seleccionan las columnas correspondientes a las variables.

```
dataset=read.csv("C:/Users/Sergio/Desktop/Practica 7/Boston.csv")
dataset=dataset[,c(7,15)]
```

Se divide el conjunto de datos en prueba y entrenamiento asignando un 80% de los datos para entrenamiento y 20% para pruebas

```
library(caTools)
set.seed((123))
split=sample.split(dataset$medv, SplitRatio=0.8)
training_set=subset(dataset, split==TRUE)
testing_set=subset(dataset, split==FALSE)
```

Posteriormente se crea una instancia de “rpart” para generar el modelo de árbol de regresión. y se ajusta al conjunto de datos de entrenamiento

```
library(rpart)
regression=rpart(formula=medv~.,
                 data=training_set,
                 control=rpart.control(minsplit=2))
```

Se realiza una predicción de un dato perteneciente al conjunto de pruebas por medio del modelo ya ajustado al conjunto de entrenamiento.

```
y_pred=predict(regression,newdata=data.frame(rm=6.096))
```

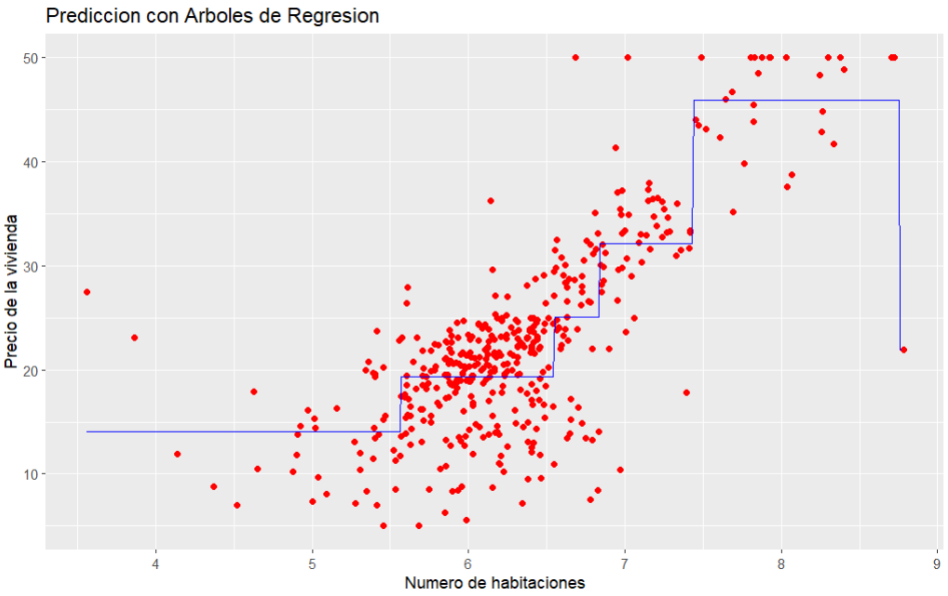
Se genera la gráfica de los datos junto con el modelo.

```
library(ggplot2)
X_grid=seq(min(training_set$rm),max(training_set$rm),0.01)
ggplot()+
  geom_point(aes(x=training_set$rm,y=training_set$medv),
             color="red")+
  geom_line(aes(x=X_grid,y=predict(regression,

newdata=data.frame(rm=X_grid))),
             color="blue")+
  ggtitle("Prediccion con Arboles de Regresion")+
  xlab("Numero de habitaciones")+
  ylab("Precio de la vivienda")
```

Se imprime en consola el valor de la predicción y el valor real.

```
cat("Precio estmidado con el modelo:",y_pred)
cat("Valor real del precio:",testing_set$medv[1])
```

Resultados del modelo de árboles de regresión.




```
Precio estmidado con el modelo: 19.32209
Valor real del precio: 18.2
```

Como se puede observar, la estimación tuvo un error de 1.12 aproximadamente con el modelo de árboles de regresión.

- Bosques aleatorios.

De igual forma se carga el dataset de Boston y se seleccionan las columnas correspondientes.

```
dataset=read.csv("C:/Users/Sergio/Desktop/Practica 7/Boston.csv")
dataset=dataset[,c(7,15)]
```

Se divide el conjunto de datos en prueba y entrenamiento.

```
library(caTools)
set.seed((123))
split=sample.split(dataset$medv, SplitRatio=0.8)
training_set=subset(dataset, split==TRUE)
testing_set=subset(dataset, split==FALSE)
```

Se genera una instancia de “randomForest” y se ajusta el modelo con el conjunto de entrenamiento.

```
library(randomForest)
set.seed(123)
regression=randomForest(x=training_set[1],
                        y=training_set$medv,
                        ntree=500)
```

De igual forma se realiza una predicción de un valor en el conjunto de pruebas con el modelo de bosques aleatorios.

```
y_pred=predict(regression,newdata=data.frame(rm=6.096))
```

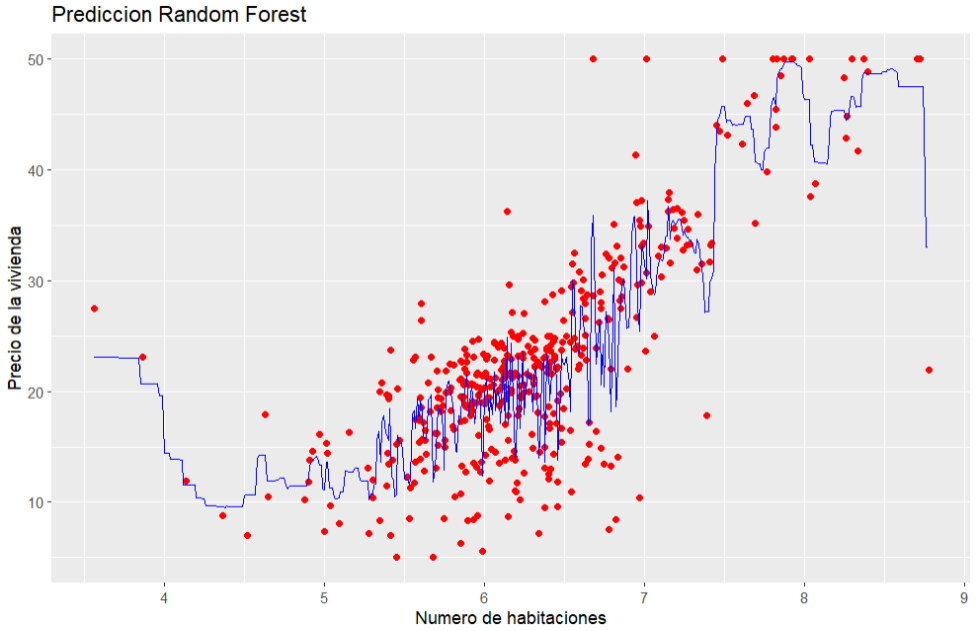
Se genera la gráfica del conjunto de datos y el modelo.

```
library(ggplot2)
X_grid=seq(min(training_set$rm),max(training_set$rm),0.01)
ggplot()+
  geom_point(aes(x=training_set$rm,y=training_set$medv),
             color="red")+
  geom_line(aes(x=X_grid,y=predict(regression,

newdata=data.frame(rm=X_grid))),
             color="blue")+

  ggtitle("Prediccion Random Forest")+
  xlab("Numero de habitaciones")+
  ylab("Precio de la vivienda")
```

Se imprime en consola el valor de la predicción así como el valor real.

```
cat("Precio estmidado con el modelo:",y_pred)
cat("Valor real del precio:",testing_set$medv[1])
```

Resultados del modelo de bosques aleatorios.




```
Precio estmidado con el modelo: 16.92521
Valor real del precio: 18.2
```

Como se puede observar, la estimación tuvo un error de 1.27 aproximadamente con el modelo de bosques aleatorios.

V. Conclusiones

Las regresiones usando árboles de decisión y bosques aleatorios son populares porque son sencillos de entender y utilizar, y suelen dar buenos resultados en muchos problemas. Además, son capaces de manejar conjuntos de datos grandes y con alta dimensionalidad sin necesidad de realizar una selección previa de atributos. Sin embargo, también tienen algunas desventajas, como la necesidad de elegir una profundidad máxima para el árbol o un número adecuado de árboles para el bosque, y pueden ser propensos a sobreajuste si se utilizan sin una validación adecuada.

VI. Referencias

Boston.csv. (2018, 9 julio). Kaggle. https://www.kaggle.com/datasets/puxama/bostoncsv?resource=download

GeeksforGeeks. (2022, 23 agosto). *Python | Decision Tree Regression using sklearn*. https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/

K, G. M. (2021, 15 diciembre). *Machine Learning Basics: Decision Tree Regression - Towards Data Science*. Medium. https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda

Beheshti, N. (2022, 5 marzo). *Random Forest Regression - Towards Data Science*. Medium. https://towardsdatascience.com/random-forest-regression-5f605132d19d