Actividad 4.1 - Edson Raul Cepeda Marquez

May 18, 2021

1 Actividad 4.1 - Redes neuronales artificiales

Construcción y entrenamiento de un modelo de red neuronal par la predicción de precios de casas utilizando el conjunto de datos de Boston_House_Prices.

1.1 Analísis de datos

1.1.1 Carga de datos

[4]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston
```

[3]:
```
boston = load_boston()
```

[5]:
```
data = pd.DataFrame(boston.data)
```

[6]:
```
data.head()
```

[6]:
```
         0     1     2    3      4      5     6       7    8      9    10  \
0  0.00632  18.0  2.31  0.0  0.538  6.575  65.2  4.0900  1.0  296.0  15.3
1  0.02731   0.0  7.07  0.0  0.469  6.421  78.9  4.9671  2.0  242.0  17.8
2  0.02729   0.0  7.07  0.0  0.469  7.185  61.1  4.9671  2.0  242.0  17.8
3  0.03237   0.0  2.18  0.0  0.458  6.998  45.8  6.0622  3.0  222.0  18.7
4  0.06905   0.0  2.18  0.0  0.458  7.147  54.2  6.0622  3.0  222.0  18.7

       11    12
0  396.90  4.98
1  396.90  9.14
2  392.83  4.03
3  394.63  2.94
4  396.90  5.33
```

Asigando precio y nombre de columnas

[7]:
```
data.columns = boston.feature_names
```

[17]:
```
data['PRICE'] = boston.target
```

[18]:
```
data.head()
```

[18]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT  PRICE
0     15.3  396.90   4.98   24.0
1     17.8  396.90   9.14   21.6
2     17.8  392.83   4.03   34.7
3     18.7  394.63   2.94   33.4
4     18.7  396.90   5.33   36.2
```

[19]:
```
print(data.shape)
```

```
(506, 14)
```

Corroborando si existen valores nulos

[20]:
```
data.isnull().sum()
```

[20]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

Estadisticas descriptivas

[21]:
```
data.describe()
```

[21]:
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
```

```
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT       PRICE
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

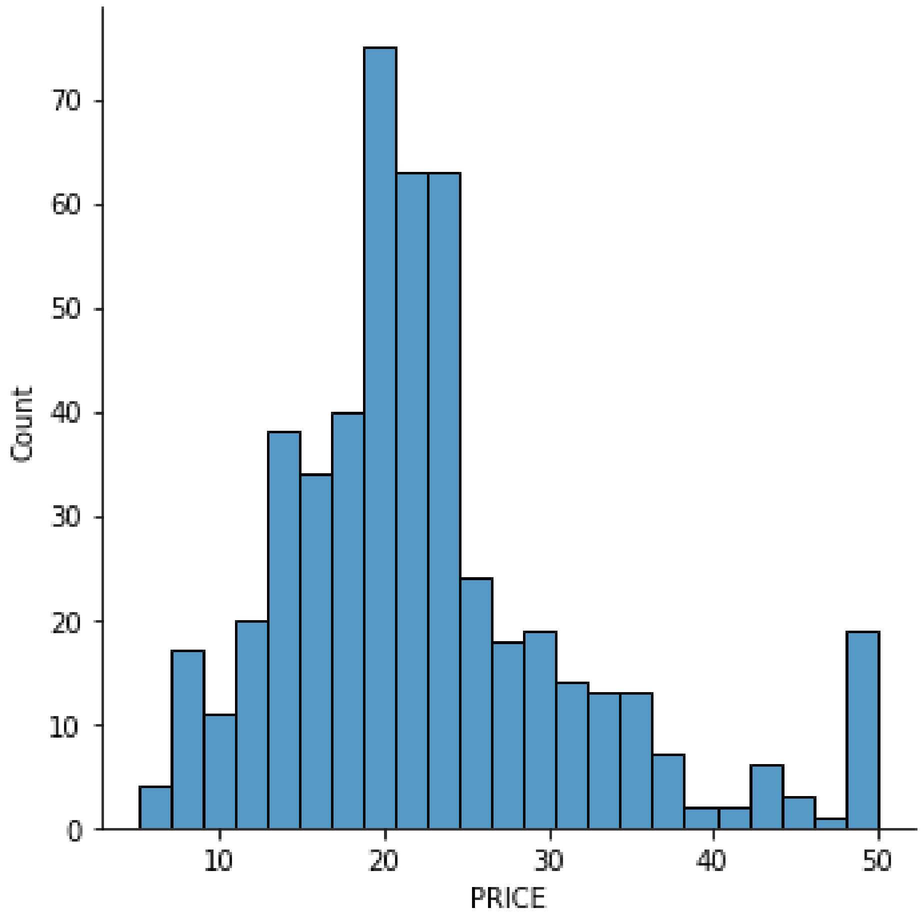
[22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  PRICE    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

Observando la distribución de los precios

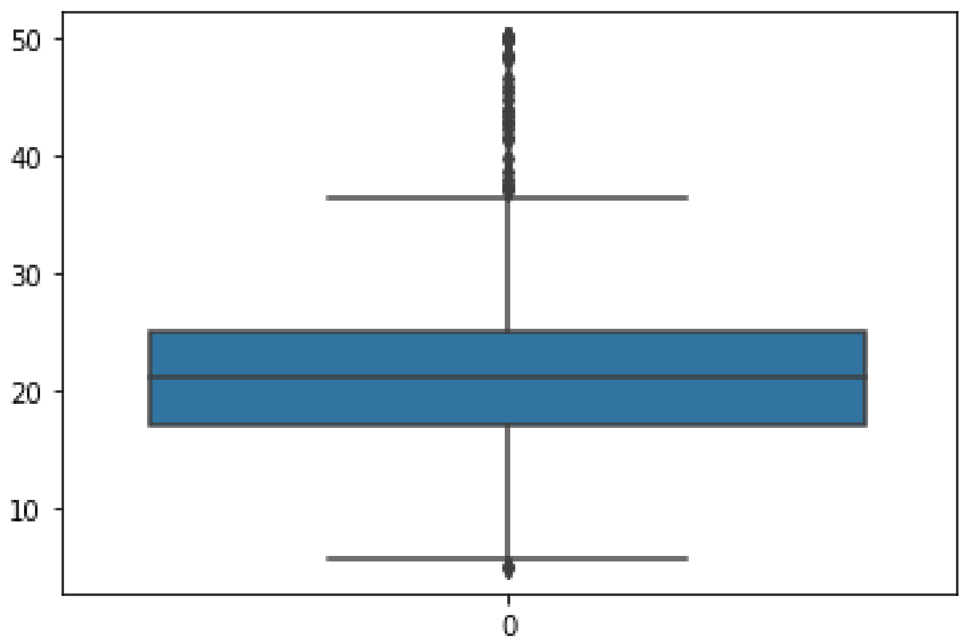
[26]:
```
import seaborn as sns
sns.displot(data=data.PRICE)
```

[26]: <seaborn.axisgrid.FacetGrid at 0x7f575af2e730>



[27]:
```
sns.boxplot(data=data.PRICE)
```

[27]: <AxesSubplot:>



Calculando los coefficientes de correlación

[28]:
```
correlation = data.corr()
```

[29]:
```
correlation.loc['PRICE']
```

[29]:
```
CRIM      -0.388305
ZN         0.360445
INDUS     -0.483725
CHAS       0.175260
NOX       -0.427321
RM         0.695360
AGE       -0.376955
DIS        0.249929
RAD       -0.381626
TAX       -0.468536
PTRATIO   -0.507787
B          0.333461
LSTAT     -0.737663
PRICE      1.000000
Name: PRICE, dtype: float64
```

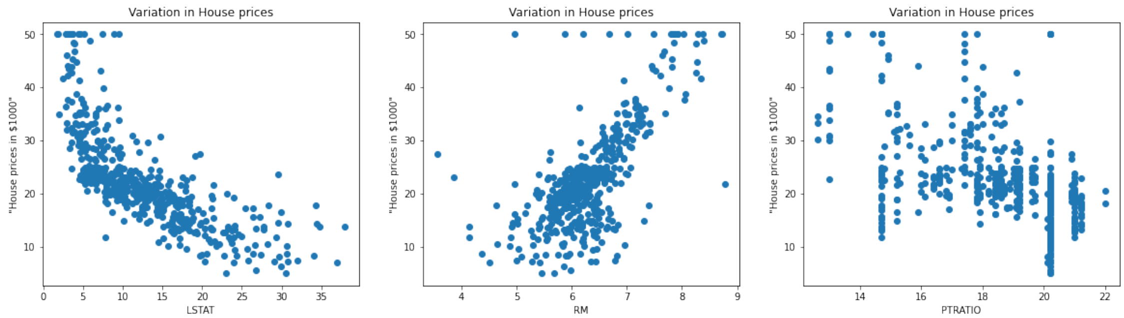
[31]:
```
import matplotlib.pyplot as plt
```

[33]:
```
fig, axes = plt.subplots(figsize=(15,12))
sns.heatmap(correlation, square=True, annot=True)
```

[33]: <AxesSubplot:>

Observando las variables más llamativas obtenidas del analísis de correlación

[34]:
```
plt.figure(figsize=(20,5))
features = ['LSTAT', 'RM', 'PTRATIO']
for i, col in enumerate(features):
    plt.subplot(1, len(features), i+1)
    x = data[col]
    y = data.PRICE
    plt.scatter(x, y, marker='o')
    plt.title("Variation in House prices")
    plt.xlabel(col)
    plt.ylabel('"House prices in $1000"')
```



1.1.2 Preparando los datos para el entrenamiento

[36]:
```
X = data.iloc[:,:-1]
```

[39]:
```
y = data.PRICE
```

[40]:
```
from sklearn.model_selection import train_test_split
```

[41]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,␣
↪random_state = 4)
```

[42]:
```
from sklearn.preprocessing import StandardScaler
```

[43]:
```
sc = StandardScaler()
```

[44]:
```
X_train = sc.fit_transform(X_train)
```

[45]:
```
X_test = sc.transform(X_test)
```

1.1.3 Construcción del modelo de red neuronal

[50]:
```
import keras
from keras.layers import Dense, Activation, Dropout
from keras.models import Sequential
```

[51]:
```
model = Sequential()
```

Definiendo la arquitectura de la red neuronal

[53]:
```
model.add(Dense(128, activation='relu', input_dim = 13))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

Entrenando la red neuronal

[54]:
```
model.fit(X_train, y_train, epochs=100)
```

```
Epoch 1/100
13/13 [==============================] - 15s 2ms/step - loss: 586.9261
Epoch 2/100
13/13 [==============================] - 0s 5ms/step - loss: 504.9662
Epoch 3/100
13/13 [==============================] - 0s 3ms/step - loss: 329.5313
Epoch 4/100
13/13 [==============================] - 0s 3ms/step - loss: 114.0957
Epoch 5/100
13/13 [==============================] - 0s 3ms/step - loss: 55.6977
Epoch 6/100
13/13 [==============================] - 0s 3ms/step - loss: 26.8154
Epoch 7/100
13/13 [==============================] - 0s 2ms/step - loss: 17.7073
Epoch 8/100
13/13 [==============================] - 0s 3ms/step - loss: 21.5348
Epoch 9/100
13/13 [==============================] - 0s 3ms/step - loss: 16.5987
Epoch 10/100
13/13 [==============================] - 0s 3ms/step - loss: 13.6926
Epoch 11/100
13/13 [==============================] - 0s 3ms/step - loss: 11.2430
Epoch 12/100
13/13 [==============================] - 0s 3ms/step - loss: 10.3592
Epoch 13/100
13/13 [==============================] - 0s 3ms/step - loss: 11.7095
Epoch 14/100
13/13 [==============================] - 0s 3ms/step - loss: 14.8494
Epoch 15/100
13/13 [==============================] - 0s 3ms/step - loss: 11.6141
Epoch 16/100
13/13 [==============================] - 0s 3ms/step - loss: 9.1067
Epoch 17/100
13/13 [==============================] - 0s 3ms/step - loss: 9.6613
Epoch 18/100
13/13 [==============================] - 0s 3ms/step - loss: 14.2203
Epoch 19/100
13/13 [==============================] - 0s 3ms/step - loss: 9.2219
Epoch 20/100
13/13 [==============================] - 0s 3ms/step - loss: 7.5127
Epoch 21/100
13/13 [==============================] - 0s 3ms/step - loss: 11.1469
Epoch 22/100
13/13 [==============================] - 0s 4ms/step - loss: 9.8665
```

```
Epoch 23/100
13/13 [==============================] - 0s 3ms/step - loss: 9.1630
Epoch 24/100
13/13 [==============================] - 0s 4ms/step - loss: 8.4459
Epoch 25/100
13/13 [==============================] - 0s 3ms/step - loss: 8.4768
Epoch 26/100
13/13 [==============================] - 0s 3ms/step - loss: 8.2416
Epoch 27/100
13/13 [==============================] - 0s 3ms/step - loss: 8.9542
Epoch 28/100
13/13 [==============================] - 0s 3ms/step - loss: 7.7439
Epoch 29/100
13/13 [==============================] - 0s 3ms/step - loss: 8.6205
Epoch 30/100
13/13 [==============================] - 0s 5ms/step - loss: 10.0104
Epoch 31/100
13/13 [==============================] - 0s 3ms/step - loss: 6.8064
Epoch 32/100
13/13 [==============================] - 0s 3ms/step - loss: 6.4711
Epoch 33/100
13/13 [==============================] - 0s 4ms/step - loss: 7.3245
Epoch 34/100
13/13 [==============================] - 0s 3ms/step - loss: 6.3654
Epoch 35/100
13/13 [==============================] - 0s 3ms/step - loss: 8.3157
Epoch 36/100
13/13 [==============================] - 0s 3ms/step - loss: 6.1892
Epoch 37/100
13/13 [==============================] - 0s 3ms/step - loss: 7.7888
Epoch 38/100
13/13 [==============================] - 0s 3ms/step - loss: 8.2177
Epoch 39/100
13/13 [==============================] - 0s 4ms/step - loss: 6.7666
Epoch 40/100
13/13 [==============================] - 0s 3ms/step - loss: 6.2496
Epoch 41/100
13/13 [==============================] - 0s 3ms/step - loss: 6.5795
Epoch 42/100
13/13 [==============================] - 0s 3ms/step - loss: 5.3427
Epoch 43/100
13/13 [==============================] - 0s 3ms/step - loss: 6.4896
Epoch 44/100
13/13 [==============================] - 0s 3ms/step - loss: 5.1552
Epoch 45/100
13/13 [==============================] - 0s 3ms/step - loss: 6.7148
Epoch 46/100
13/13 [==============================] - 0s 3ms/step - loss: 5.3051
```

```
Epoch 47/100
13/13 [==============================] - 0s 3ms/step - loss: 4.8733
Epoch 48/100
13/13 [==============================] - 0s 3ms/step - loss: 5.7832
Epoch 49/100
13/13 [==============================] - 0s 3ms/step - loss: 4.8653
Epoch 50/100
13/13 [==============================] - 0s 3ms/step - loss: 5.7771
Epoch 51/100
13/13 [==============================] - 0s 3ms/step - loss: 4.5463
Epoch 52/100
13/13 [==============================] - 0s 4ms/step - loss: 4.6723
Epoch 53/100
13/13 [==============================] - 0s 4ms/step - loss: 4.7725
Epoch 54/100
13/13 [==============================] - 0s 4ms/step - loss: 3.8493
Epoch 55/100
13/13 [==============================] - 0s 5ms/step - loss: 4.4639
Epoch 56/100
13/13 [==============================] - 0s 4ms/step - loss: 4.0582
Epoch 57/100
13/13 [==============================] - 0s 3ms/step - loss: 3.6706
Epoch 58/100
13/13 [==============================] - 0s 3ms/step - loss: 4.4159
Epoch 59/100
13/13 [==============================] - 0s 3ms/step - loss: 3.9814
Epoch 60/100
13/13 [==============================] - 0s 4ms/step - loss: 3.7978
Epoch 61/100
13/13 [==============================] - 0s 5ms/step - loss: 3.7584
Epoch 62/100
13/13 [==============================] - 0s 4ms/step - loss: 4.7628
Epoch 63/100
13/13 [==============================] - 0s 4ms/step - loss: 3.7640
Epoch 64/100
13/13 [==============================] - 0s 4ms/step - loss: 4.2632
Epoch 65/100
13/13 [==============================] - 0s 3ms/step - loss: 3.6611
Epoch 66/100
13/13 [==============================] - 0s 4ms/step - loss: 3.4057
Epoch 67/100
13/13 [==============================] - 0s 3ms/step - loss: 4.2729
Epoch 68/100
13/13 [==============================] - 0s 4ms/step - loss: 4.0470
Epoch 69/100
13/13 [==============================] - 0s 4ms/step - loss: 4.6172
Epoch 70/100
13/13 [==============================] - 0s 3ms/step - loss: 5.8933
```

```
Epoch 71/100
13/13 [==============================] - 0s 4ms/step - loss: 4.0761
Epoch 72/100
13/13 [==============================] - 0s 3ms/step - loss: 3.6938
Epoch 73/100
13/13 [==============================] - 0s 3ms/step - loss: 3.5321
Epoch 74/100
13/13 [==============================] - 0s 3ms/step - loss: 5.2945
Epoch 75/100
13/13 [==============================] - 0s 3ms/step - loss: 3.7234
Epoch 76/100
13/13 [==============================] - 0s 4ms/step - loss: 3.1456
Epoch 77/100
13/13 [==============================] - 0s 4ms/step - loss: 3.1319
Epoch 78/100
13/13 [==============================] - 0s 4ms/step - loss: 3.7920
Epoch 79/100
13/13 [==============================] - 0s 3ms/step - loss: 3.1307
Epoch 80/100
13/13 [==============================] - 0s 3ms/step - loss: 4.2429
Epoch 81/100
13/13 [==============================] - 0s 3ms/step - loss: 3.5938
Epoch 82/100
13/13 [==============================] - 0s 3ms/step - loss: 3.0202
Epoch 83/100
13/13 [==============================] - 0s 3ms/step - loss: 3.1307
Epoch 84/100
13/13 [==============================] - 0s 4ms/step - loss: 3.3729
Epoch 85/100
13/13 [==============================] - 0s 3ms/step - loss: 2.8520
Epoch 86/100
13/13 [==============================] - 0s 3ms/step - loss: 3.1072
Epoch 87/100
13/13 [==============================] - 0s 3ms/step - loss: 2.5570
Epoch 88/100
13/13 [==============================] - 0s 3ms/step - loss: 2.7645
Epoch 89/100
13/13 [==============================] - 0s 3ms/step - loss: 2.9271
Epoch 90/100
13/13 [==============================] - 0s 3ms/step - loss: 2.6950
Epoch 91/100
13/13 [==============================] - 0s 3ms/step - loss: 2.8767
Epoch 92/100
13/13 [==============================] - 0s 3ms/step - loss: 2.4233
Epoch 93/100
13/13 [==============================] - 0s 4ms/step - loss: 2.7048
Epoch 94/100
13/13 [==============================] - 0s 3ms/step - loss: 2.7012
```

```
Epoch 95/100
13/13 [==============================] - 0s 3ms/step - loss: 2.7719
Epoch 96/100
13/13 [==============================] - 0s 3ms/step - loss: 2.4201
Epoch 97/100
13/13 [==============================] - 0s 3ms/step - loss: 2.4178
Epoch 98/100
13/13 [==============================] - 0s 4ms/step - loss: 2.5083
Epoch 99/100
13/13 [==============================] - 0s 3ms/step - loss: 2.5458
Epoch 100/100
13/13 [==============================] - 0s 3ms/step - loss: 2.5559
```

[54]: <keras.callbacks.History at 0x7f5718059310>

1.1.4 Evaluando los resultados

[55]:
```
y_pred = model.predict(X_test)
```

[56]:
```
from sklearn.metrics import r2_score
```

[57]:
```
r2 = r2_score(y_test, y_pred)
```

[58]:
```
print(r2)
```

```
0.9051913696866637
```

[59]:
```
from sklearn.metrics import mean_squared_error
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))
print(rmse)
```

```
2.9676084866841217
```