lab2_new

September 18, 2023

Homework 2

The Boston housing data set was collected in the 1970s to study the relationship between house price and various factors such as the house size, crime rate, socio-economic status, etc. Since the variables are easy to understand, the data set is ideal for learning basic concepts in machine learning.

[1]:
```
# Importing data
import pandas as pd
import numpy as np

names =[
    'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM',
    'AGE',  'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'PRICE'
]
# To do 1: Complete the code
df=pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪housing/housing.data",header=None,delim_whitespace=True,␣
 ↪names=names,na_values='?')
```

[2]:
```
# To do 2 : Display the first six rows of the data frame

df.head(6)
```

[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7

Basic Manipulations of the data

[3]:
```
# TODO 3: What is the shape of the data? How many attributes are there? How many samples?
df.shape
```

[3]:
```
(506, 14)
```

[4]:
```
d=len(df.columns)
```

[5]:
```
e=len(df.index)
```

[6]:
```
print("num samples={0:5.1f}, num attributes={1:5.1f}".format(d,e))
```

```
num samples= 14.0, num attributes=506.0
```

The shape of the data is (506,14) The number of samples is 506 and the number of attributes is 14

[7]:
```
# TODO 4: Create a response vector y with the values in the column PRICE

y=df['PRICE']
y.head(15)
```

[7]:
```
0     24.0
1     21.6
2     34.7
3     33.4
4     36.2
5     28.7
6     22.9
7     27.1
8     16.5
9     18.9
10    15.0
11    18.9
12    21.7
13    20.4
14    18.2
Name: PRICE, dtype: float64
```

[8]:
```
# TODO 5: Use the response vector y to find the mean house price in thousands and the fraction of homes that are above $40k
ny=np.mean(y)
ny_mean=np.array(df['PRICE'])  # converting into an array
ny_40k=(np.mean(y>40))*100
print("The mean house price is{0:7.2f} thousands of dollars, Only {1:7.2f}percent are above $40k".format(ny,ny_40k))
```

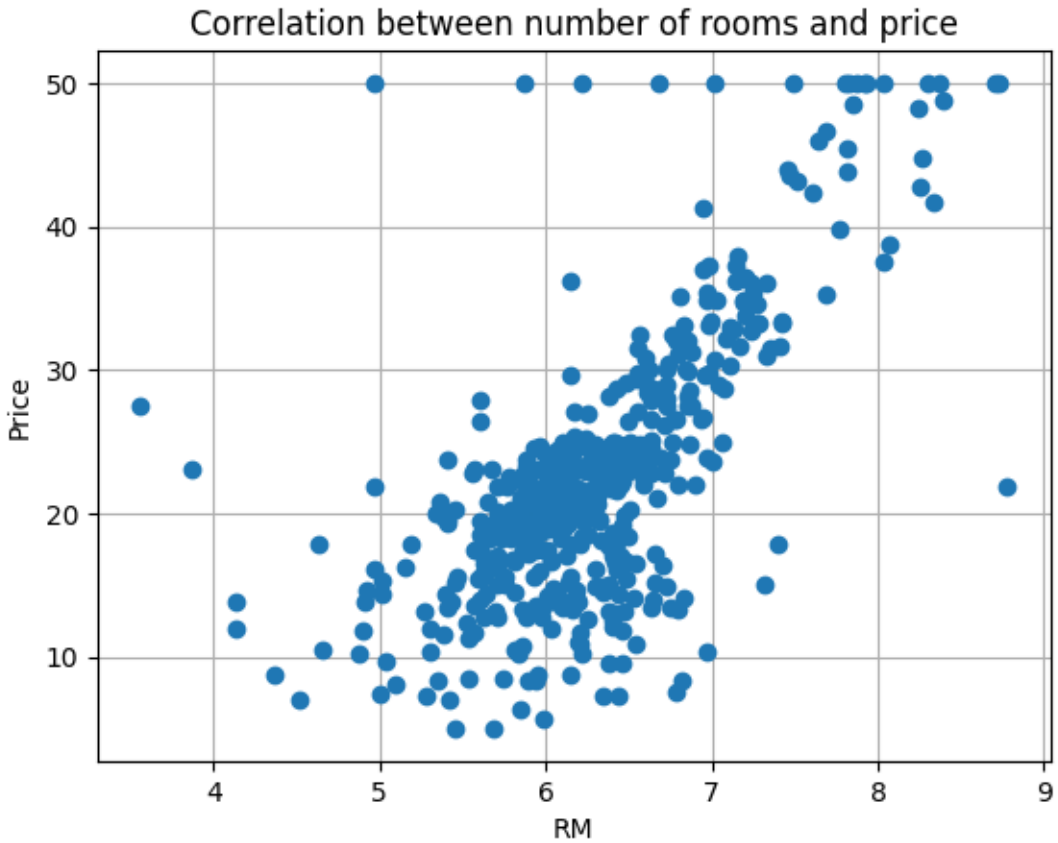
```
The mean house price is  22.53 thousands of dollars, Only    6.13 percent are
above $40k
```

Visualizing the data

[9]:
```
import matplotlib
import matplotlib.pyplot as plt

# TODO 6: create a predictor vector x containing the values in the RM column

x=np.array(df['RM'])
y=np.array(df['PRICE'])
 #TODO 7: Create a scatter plot of the price vs. the RM attribute. Make sure␣
 ↪your plot has grid lines and label the axes with
x=np.array(df['RM'])
y=np.array(df['PRICE'])
plt.plot(x,y,'o')
plt.xlabel('RM')
plt.ylabel('Price')
plt.title('Correlation between number of rooms and price')
plt.grid(True)
```



Fitting a Simple Linear Model

[10]:
```
 # TODO 8: complete the following code
    # beta0 = ...
    # beta1 = ...
    # rsq = ...
def fit_linear(x,y):
  xm=np.mean(x)
  ym=np.mean(y)
  syy=np.mean((y-ym)**2)
  syx=np.mean((y-ym)*(x-xm))
  sxx=np.mean((x-xm)**2)
  beta1=syx/sxx
  beta0=ym-beta1*xm
  yhat=beta0+beta1*x
  rsq=np.mean((y-yhat)**2)
  return beta0, beta1,rsq
```

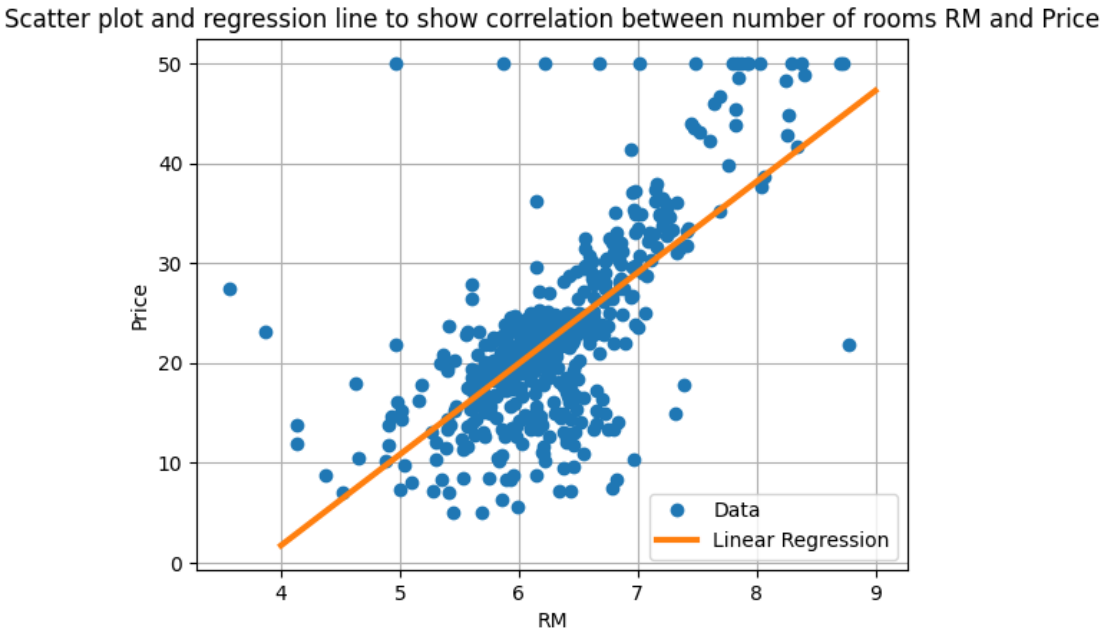
[11]:
```
# TODO 9: print the values beta0, beta1 and rsq for the linear model of price␣
 ↪vs. number of rooms.
result=fit_linear(x,y)
print("The value of beta0 is={0:7.2f}, The value of beta 1 is={1:7.2f}, The␣
 ↪value of RSS is={2:7.2f}".format(result[0],result[1],result[2]))
```

```
The value of beta0 is= -34.67, The value of beta 1 is=   9.10, The value of RSS
is=  43.60
```

[12]:
```
# TODO 10: Replot the scatter plot above, but now with the regression line.
```

[13]:
```
import matplotlib
import matplotlib.pyplot as plt
result=fit_linear(x,y)
xp=np.linspace(4,9)
yp=result[1]*xp+result[0]
plt.plot(x,y,'o')
plt.plot(xp,yp,'-',linewidth=3)
plt.xlabel('RM')
plt.ylabel('Price')
plt.title('Scatter plot and regression line to show correlation between number␣
 ↪of rooms RM and Price')
plt.grid(True)
plt.legend(['Data','Linear Regression'])
```

[13]:
```
<matplotlib.legend.Legend at 0x7aea6568de70>
```



Another Approach of Linear Regression

[14]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
x_new=np.array([x]).reshape((-1,1))
model.fit(x_new,y)
```

[14]:
```
LinearRegression()
```

[15]:
```
model=LinearRegression().fit(x_new,y)
r_sq=model.score(x_new,y)
print(f"Coefficient of determination:{r_sq}")
```

```
Coefficient of determination:0.48352545599133423
```

[16]:
```
print(f"intercept:{model.intercept_}")
```

```
intercept:-34.67062077643857
```

[17]:
```
print(f"slope:{model.coef_}")
```

```
slope:[9.10210898]
```

[18]:
```
# Predictor Response
y_pred=model.predict(x_new)
print(f"predicted response:\n{y_pred}")
```

```
predicted response:
[25.17574577 23.77402099 30.72803225 29.02593787 30.38215211 23.85593997
 20.05125842 21.50759586 16.5833549  19.97844155 23.3735282  20.02395209
 18.93169901 19.47782555 20.81583557 18.43108302 19.35039603 19.85101202
 14.99048582 17.45715736 16.02812625 19.6234593  21.23453259 18.23993873
 19.25027283 16.29208741 18.23993873 20.36983223 24.44757706 26.07685456
 17.32972783 20.59738496 19.48692766 17.22050253 20.81583557 19.33219181
 18.49479778 18.57671676 19.63256141 25.35778795 29.26259271 26.95065703
 21.48028953 21.86257811 20.57007863 17.04756245 17.99418179 20.21509638
 14.47166561 16.31939374 19.60525508 20.98877564 24.5932108  19.92382889
 18.9225969  31.31056723 23.42814085 27.36935404 21.26183891 19.27757916
 17.58458688 19.63256141 24.09259481 26.87784015 29.99076143 22.58164472
 18.0032839  18.83157581 16.24657686 18.89529058 23.73761256 19.58705086
 20.53367019 22.17204981 22.42690886 22.54523628 22.48152152 21.21632837
 22.05372239 18.79516738 26.55926634 25.57623857 22.69087002 21.46208531
 23.4827535  25.67636177 20.07856475 21.0433883  29.10785685 29.7632087
 23.73761256 23.62838725 23.96516528 21.86257811 22.20845825 25.63085122
 21.42567687 38.77429659 36.50787146 32.83061943 26.55926634 27.05078022
 23.62838725 21.18902204 21.46208531 18.58581887 18.44928724 21.09800095
 24.25643277 22.02641607 21.71694436 26.45004103 19.15014963 20.77942714
 22.25396879 19.28668126 21.54400429 20.1331774  18.77696316 17.49356579
 18.75875894 19.97844155 19.58705086 18.63132942 18.84067792 19.81460358
 16.41951693 17.14768565 23.86504208 16.63796755 24.11079902 22.90932064
 23.32801765 18.32185771 17.73022063 22.99123962 19.41411079 24.07439059
 18.64043153 21.31645157 21.52580007 11.0128642  14.50807405 15.09971113
  9.95701956 21.12530728 16.55604857 10.16636806 12.5329164  16.27388319
 21.05249041 14.51717616 10.94914944 17.2933194  21.11620517 21.32555368
 13.31569777 28.52532188 20.5427723  24.58410869 22.21756036 33.49507338
 36.34403349 41.55954194 18.6131252  20.86134612 37.50000134 18.82247371
 22.84560588 23.60108092 18.80426949 18.84978003 16.04633047 23.72851045
 18.65863574 24.91178461 20.12407529 22.80919744 27.76984683 28.86209991
 36.00725546 21.2527368  30.45496898 25.06652047 16.33759795 21.33465578
 36.60799466 27.05988233 25.0028057  30.72803225 28.59813875 26.66849165
 30.66431749 27.2237203  25.43970694 37.00848745 31.65644737 30.01806775
 31.53811995 28.81658937 30.2729268  21.41657477 34.59642857 36.80824105
 38.45572278 18.94990323 22.90932064 17.96687546 20.52456809 13.97104962
 19.57794875 14.51717616 18.18532608 23.35532398 14.58999303 21.59861695
 18.9225969  25.78558708 19.49602977 23.33711976 28.59813875 21.43477898
 27.94278691 25.56713646 40.56741206 44.74528008 38.51033543 30.52778586
 35.28818885 24.96639727 19.76909304 32.79421099 41.2136618  40.39447199
 26.55016423 20.72481448 25.68546388 32.30269711 24.32014753 25.45791115
 28.10662487 20.80673346 23.20058813 23.51916194 16.23747476 16.34670006
 20.92506088 21.99910974 23.8832463  26.47734736 24.37476018 23.92875684
 28.65275141 40.5036973  20.92506088 18.8133716  33.17649957 44.5541358
 32.07514438 27.60600887 30.89187022 33.77723876 41.76889045 32.02053173
 30.91917654 15.93710516 29.17157162 40.84957744 33.32213331 19.21386439
 18.63132942 22.12653927 24.83896774 35.3336994  26.84143172 27.71523418
 31.47440519 27.46037513 24.32924964 27.3329456  36.50787146 28.7528746
```

```
34.91500238 37.44538868 29.84512768 24.06528848 22.03551818 21.84437389
22.80919744 25.08472469 27.77894894 30.39125422 25.67636177 21.09800095
20.02395209 26.113263   24.93909094 18.03059022 23.08226071 29.41732856
27.86997003 25.31227741 24.44757706 28.88030413 31.19223981 25.54893224
32.86702786 27.66972364 25.72187231 19.68717406 10.59416719 21.05249041
20.15138162 22.3631941  25.1029289  17.25691096 19.15925174 17.95777335
23.41903874 20.97057143 23.81953154 23.36442609 20.31521958 17.28421729
23.71940834 23.86504208 22.78189111 20.69750816 18.74055473 22.9730354
21.2527368  17.26601307 20.22419849 22.81829955 22.76368689 20.27881114
18.74965683 18.98631167 20.47905754 19.80550148 19.65076562 31.23775036
24.85717196 26.27710096 27.89727636 20.06946264 19.01361799 24.63872134
25.72187231 28.48891344 24.40206651 25.21215421 18.88618847 26.56836845
16.87462238 19.35949814 21.87168021 23.53736616 21.09800095 20.96146932
23.56467249 22.22666246 14.13488758 18.14891764 45.24589608 -2.25801069
10.5031461   0.49082622 10.56686086 26.15877354 29.18977584 21.90808865
18.80426949  9.98432589  2.99390619 31.8931022  25.84930184 27.16910764
23.40083452 21.97180341 28.7528746  24.90268251 15.71865454 15.5730208
 5.08739125 13.36120832  7.6723902  10.83992413  9.74767105 14.38974663
17.32972783 20.40624067 11.16760005 21.69874014 18.9134948  24.22912644
23.62838725 17.63919954 14.9631795  18.59492098 19.82370569 23.06405649
23.61928514 14.01656016 15.673144   17.05666456  2.99390619 16.37400639
16.45592537 27.69702996 17.73022063 25.92211871  7.45393959 12.25075102
 6.46180971 23.89234841 27.05988233 13.60696526 19.55064242 27.44217091
23.6829999  19.99664576 16.73809075 20.87955034 15.9826157  18.99541378
18.45838935 21.78065912 21.69874014 23.40083452 23.10956704 27.52408989
23.81042943 23.91055263 21.83527178 25.66725966 24.13810535 21.32555368
19.35039603 16.54694646 18.28544928 23.63748936 21.93539498 24.35655597
18.6131252  24.11990113 23.04585227 22.22666246 21.62592327 23.73761256
26.75951274 25.90391449 22.64535948 32.62127092 26.56836845 24.72064033
19.7235825  19.35949814 22.68176791 20.67930394 26.32261151 23.36442609
22.82740166 24.61141502 21.84437389 17.74842485 19.50513188 19.96933944
19.26847705 17.32972783 21.46208531 22.02641607 23.91965474 28.86209991
14.72652466 21.41657477 24.34745386 13.60696526 21.62592327 22.02641607
22.14474348 26.76861485 29.59937074 17.77573117 18.76786105 22.78189111
20.97967353 19.07733276 14.97228161 14.60819725 11.68642026 19.78729726
19.78729726 17.27511518 19.26847705 16.93833715 14.38974663 18.06699866
20.11497318 16.01902414 20.18779005 25.33958374 21.03428619 28.82569148
27.16910764 20.21509638]
```

Compute Coefficients of Determination

[19]:
```
# TODO 11:  compute the  ^2 values for all the predictors and output the values␣
 ↪in a table.

df.iloc[1]

# assigning the np.array
x1=np.array(df['CRIM'])
```

```
x2=np.array(df['ZN'])
x3=np.array(df['INDUS'])
x4=np.array(df['CHAS'])
x5=np.array(df['NOX'])
x6=np.array(df['AGE'])
x7=np.array(df['DIS'])
x8=np.array(df['RAD'])
x9=np.array(df['TAX'])
x10=np.array(df['PTRATIO'])
x11=np.array(df['B'])
x12=np.array(df['LSTAT'])

# defining the r_square function
def r_square(x1,y):
  xm1=np.mean(x1)
  ym=np.mean(y)
  syy1=np.mean((y-ym)**2)
  syx1=np.mean((y-ym)*(x1-xm1))
  sxx1=np.mean((x1-xm1)**2)
  beta11=syx1/sxx1
  beta01=ym-beta11*xm1
  yhat1=beta01+beta11*x1
  rsq1=np.mean((y-yhat1)**2)
  return beta01, beta11,rsq1

r_sq_1=r_square(x1,y)
r_sq_2=r_square(x2,y)
r_sq_3=r_square(x3,y)
r_sq_4=r_square(x4,y)
r_sq_5=r_square(x5,y)
r_sq_6=r_square(x6,y)
r_sq_7=r_square(x7,y)
r_sq_8=r_square(x8,y)
r_sq_9=r_square(x9,y)
r_sq_10=r_square(x10,y)
r_sq_11=r_square(x11,y)
r_sq_12=r_square(x12,y)


print("The r_square of CM1 is:{0:7.2f}".format(r_sq_1[2]))
print("The r_square of ZN is:{0:7.2f}".format(r_sq_2[2]))
print("The r_square of INDUS is:{0:7.2f}".format(r_sq_3[2]))
print("The r_square of CHAS is:{0:7.2f}".format(r_sq_4[2]))
print("The r_square of NOX is:{0:7.2f}".format(r_sq_5[2]))
print("The r_square of AGE is:{0:7.2f}".format(r_sq_6[2]))
print("The r_square of DIS is:{0:7.2f}".format(r_sq_7[2]))
print("The r_square of RAD is:{0:7.2f}".format(r_sq_8[2]))
```

```
print("The r_square of TAX is:{0:7.2f}".format(r_sq_9[2]))
print("The r_square of PTRATIO is:{0:7.2f}".format(r_sq_10[2]))
print("The r_square of B is:{0:7.2f}".format(r_sq_11[2]))
print("The r_square of LSTAT is:{0:7.2f}".format(r_sq_12[2]))


#df.iloc[1]
```

```
The r_square of CM1 is:  71.69
The r_square of ZN is:  73.45
The r_square of INDUS is:  64.67
The r_square of CHAS is:  81.83
The r_square of NOX is:  69.00
The r_square of AGE is:  72.42
The r_square of DIS is:  79.15
The r_square of RAD is:  72.12
The r_square of TAX is:  65.89
The r_square of PTRATIO is:  62.65
The r_square of B is:  75.03
The r_square of LSTAT is:  38.48
```