Linear Regression

1. Download the housing dataset from https://archive.ics.uci.edu/ml/machine-learning-databaseshousing/

2. Pick a number of features that you think may be correlated and plot pairs of them to confirm that they are correlated.

3. Pick one feature that you think can be predicted by the other features in the dataset. The feature to be predicted needs to have numerical values.

4. Separate the data into training, validation, and test sets.

5. Apply various Scikit Learn regression methods to the data.

6. Quantify how good a job each method did in predicting the value of the dependent variable (using your test data set). Discuss why you believe each method perform as it did.

7. Repeat steps 3. to 6. For a second feature.

8. Upload your .ipynb file. Include your discussions (Step 6) in either a word or pdf file.

Installing the required libraries

In [1]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from matplotlib import figure
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
nums = np.genfromtxt('housing.data')
num_columns = [
    'CRIM',
    'ZN',
    'INUS',
    'CHAS',
    'NOX',
    'RM',
    'AGE',
    'DIS',
    'RAD',
    'TAX',
    'PTRATIO',
    'B',
    'LSTAT',
    'MEDV'
]
df = pd.DataFrame(data=nums, columns=num_columns)
```

```
In [3]: df = df.astype({'CHAS': 'int32', 'RAD': 'int32'})
```

Attribute Information:

```
1. CRIM      per capita crime rate by town
2. ZN        proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS     proportion of non-retail business acres per town
4. CHAS      Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX       nitric oxides concentration (parts per 10 million)
6. RM        average number of rooms per dwelling
7. AGE       proportion of owner-occupied units built prior to 1940
8. DIS       weighted distances to five Boston employment centres
9. RAD       index of accessibility to radial highways
10. TAX      full-value property-tax rate per $10,000
11. PTRATIO  pupil-teacher ratio by town
12. B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT    % lower status of the population
14. MEDV     Median value of owner-occupied homes in $1000's
```

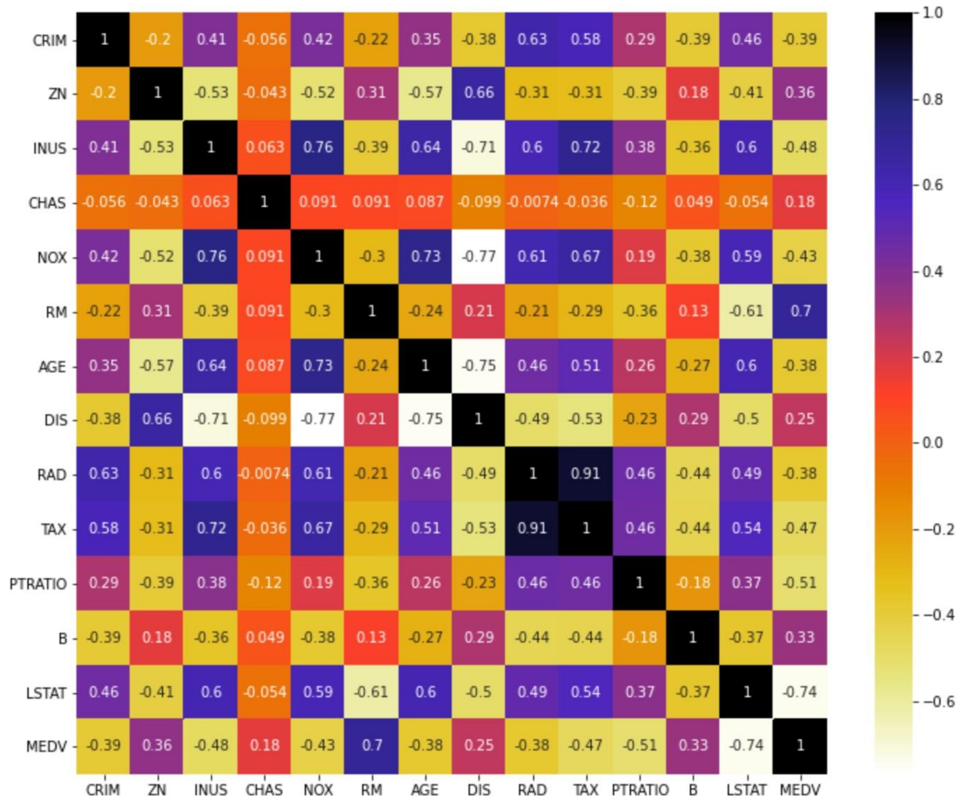
```
In [4]: df.head(10)
```

Out[4]:

	CRIM	ZN	INUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
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
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311.0	15.2	386.71	17.10	18.9

Plotting a correlation heat map of all the features in the dataset

```
In [5]: plt.figure(figsize=(12,10))
        cor=df.corr()
        sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
        plt.show()
```



	CRIM	ZN	INUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	-0.2	0.41	-0.056	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39	0.46	-0.39
ZN	-0.2	1	-0.53	-0.043	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
INUS	0.41	-0.53	1	0.063	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
CHAS	-0.056	-0.043	0.063	1	0.091	0.091	0.087	-0.099	-0.0074	-0.036	-0.12	0.049	-0.054	0.18
NOX	0.42	-0.52	0.76	0.091	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
RM	-0.22	0.31	-0.39	0.091	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
AGE	0.35	-0.57	0.64	0.087	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
DIS	-0.38	0.66	-0.71	-0.099	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
RAD	0.63	-0.31	0.6	-0.0074	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
TAX	0.58	-0.31	0.72	-0.036	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.39	0.18	-0.36	0.049	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
LSTAT	0.46	-0.41	0.6	-0.054	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
MEDV	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

Through this correlation heat map of different features we can easily see and confirm which features are correlated and what is the value of correlation between them on a scale of 0 to 1.

Separating the data into training, validation, and test sets.

Selecting the first target feature (MEDV)
Splitting the data and fitting it to a model, training the model to predict MEDV on the basis of other features using linear regression model.

```
In [6]: df1=df.drop('MEDV',axis='columns')
        target=df.MEDV

In [7]: x_train, x_test, y_train, y_test = train_test_split(df1, target, test_size=0.2, random_state=10)
```

Applying Scikit Learn regression methods to the data

Linear regression

```
In [8]: model = LinearRegression()

In [9]: model.fit(x_train,y_train)
Out[9]: LinearRegression()
```

Testing accuracy

```
In [10]: model.score(x_test,y_test)
Out[10]: 0.6709339839115628
```

Training accuracy

```
In [11]: model.score(x_train,y_train)
Out[11]: 0.750121534530608
```

Bayesian ridge model

```
In [12]: from sklearn.linear_model import BayesianRidge

In [13]: model1 = BayesianRidge()

In [14]: model1.fit(x_train,y_train)
Out[14]: BayesianRidge()
```

Testing accuracy

```
In [15]: model1.score(x_test,y_test)
Out[15]: 0.6445714222801723
```

Training accuracy

```
In [16]: model1.score(x_train,y_train)
Out[16]: 0.7393697905033569
```

<u>Decision tree regression</u>

```
In [17]: from sklearn import tree
```

```
In [18]: dec = tree.DecisionTreeRegressor(max_depth=1)
         dec.fit(x_train,y_train)
Out[18]: DecisionTreeRegressor(max_depth=1)
```

Training accuracy

```
In [19]: dec.score(x_train,y_train)
Out[19]: 0.4530739966160767
```

Testing accuracy

```
In [20]: dec.score(x_test,y_test)
Out[20]: 0.3682588795705317
```

Choosing a second feature (LSTAT)

```
In [21]: df2=df.drop('LSTAT',axis='columns')
         target2=df.LSTAT
```

```
In [22]: x_train, x_test, y_train, y_test = train_test_split(df2, target2, test_size=0.2, random_state=10)
```

<u>Linear regression model</u>

```
In [23]: model1 = LinearRegression()
```

```
In [24]: model1.fit(x_train,y_train)
Out[24]: LinearRegression()
```

Training accuracy

```
In [25]: model1.score(x_train,y_train)
Out[25]: 0.7288262380910075
```

Testing accuracy

```
In [26]: model1.score(x_test,y_test)
Out[26]: 0.6436715095177872
```

Bayesian ridge regression

```
In [27]: from sklearn.linear_model import BayesianRidge
```

```
In [28]: model1 = BayesianRidge()
```

```
In [29]: model1.fit(x_train,y_train)
Out[29]: BayesianRidge()
```

Training accuracy

```
In [31]: model1.score(x_train,y_train)
Out[31]: 0.7216263084578604
```

Testing accuracy

```
In [30]: model1.score(x_test,y_test)
Out[30]: 0.6156838478923587
```

Decision tree regression

```
In [32]: from sklearn import tree
```

```
In [33]: dec = tree.DecisionTreeRegressor(max_depth=1)
         dec.fit(x_train,y_train)
Out[33]: DecisionTreeRegressor(max_depth=1)
```

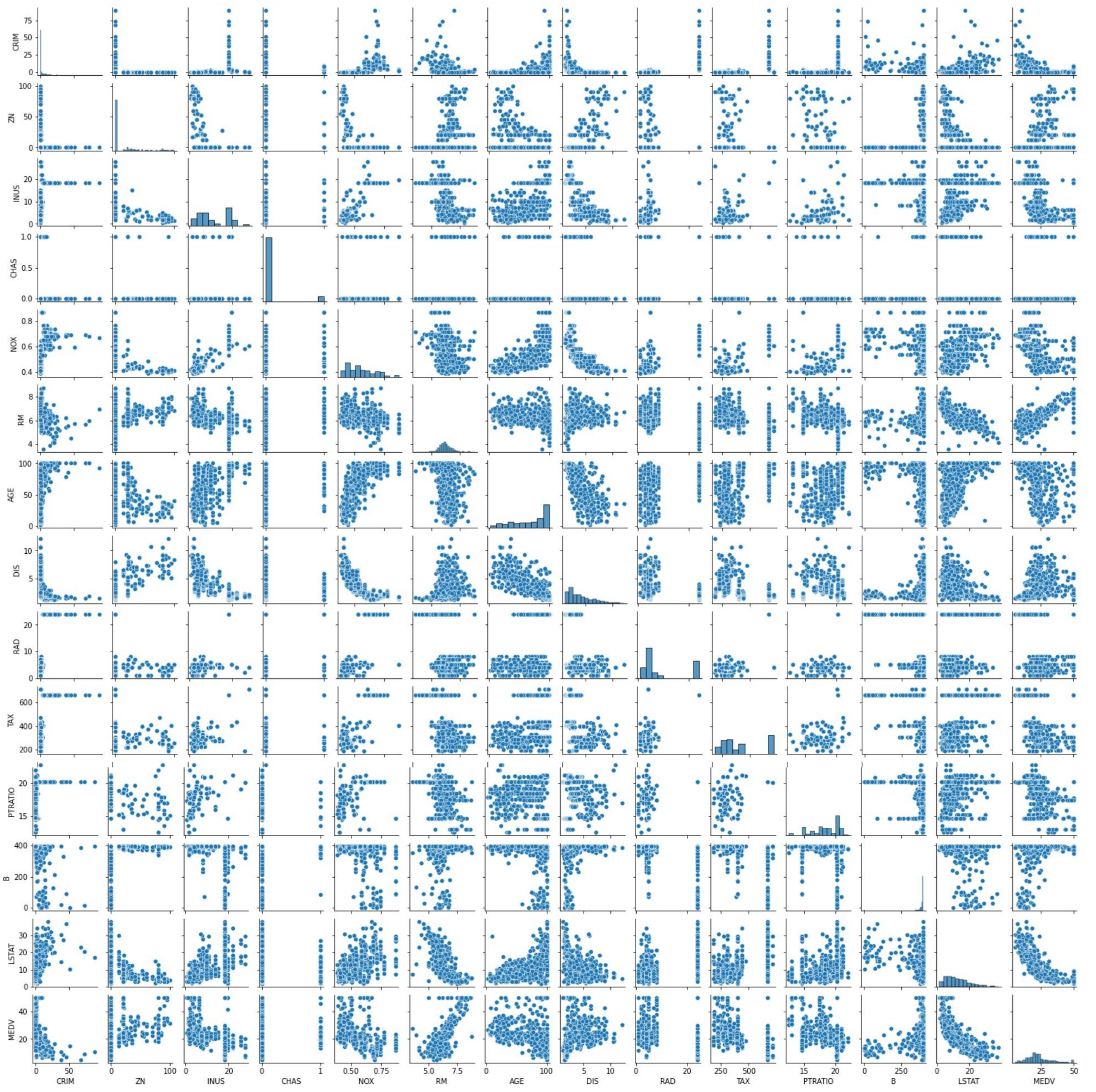
Training accuracy

```
In [34]: dec.score(x_train,y_train)
Out[34]: 0.5261767192618003
```

Testing accuracy

```
In [35]: dec.score(x_test,y_test)
Out[35]: 0.4581326687510152
```

```
In [36]: import seaborn as sns
         sns.pairplot(df,height= 1.5)
Out[36]: <seaborn.axisgrid.PairGrid at 0x7ff71c55d7f0>
```



Observation: From the above experiment, it is clear that:
The best performing model was linear regression with nearly 64% accuracy.
Next came, bayesian ridge regression with 61% accuracy.
The model which performed the least in comparison to others was the decision tree model with only 45% accuracy.

The performance of any model is hugely influences by the type of the data set and the functioning of the model. Based off these factors, we decide which model is best suited for which data set.

In the case of this particular dataset, the features show to have linear correlation. Hence, linear regression would be the best suited model.
If the dataset didn't exhibit linear correlation, then decision tree would have given better accuracy.
Hence, in conclusion, linear regression out-performed decision tree in this particular case.