Practical Activity 4.3.2 Linear regression

\

1 Week 4 Hands-on Task 4.3.2: Linear Regression

This notebook is an excercise for developing a linear regression models for house price prediction. We apply the concepts discussed in - Concept 4.1: Introduction to regression analysis - Concept 4.3: Linear regression

We will use the following python libraries for this practical. - Pandas: https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html - scikit-learn: https://scikit-learn.org/stable/index.html

2 The Housing dataset

We will use the same dataset as Practical Activity 4.1 which contains information about houses in the suburbs of Boston. It has the following features or attributes: - CRIM: Per capita crime rate by town - ZN: Proportion of residential land zoned for lots over 25,000 sq. ft. - INDUS: Proportion of non-retail business acres per town - CHAS: Charles River dummy variable (= 1 if tract bounds river and 0 otherwise) - NOX: Nitric oxide concentration (parts per 10 million) - RM: Average number of rooms per dwelling - AGE: Proportion of owner-occupied units built prior to 1940 - DIS: Weighted distances to five Boston employment centers - RAD: Index of accessibility to radial highways - TAX: Full-value property tax rate per $10,000 - PTRATIO: Pupil-teacher ratio by town - B: 1000(Bk – 0.63)2, where Bk is the proportion of [people of African American descent] by town - LSTAT: Percentage of lower status of the population - MEDV: Median value of owner-occupied homes in $1000s

Goal: our goal for this practical activity is to develop a linear regression model to predict the value of a house.

3 Linear regression model

3.1 Simple regerssion:

$y = w_0x_0 + w_1x_1$ where, $x_0 = 1$

3.2 Multiple linear regerssion:

$y = w_0x_0 + w_1x_1 + \cdots + w_mx_m$ where, $x_0 = 1$

4 Data Loading and preprocessing

For this task, we will use the data from sklearn library.

[7]:
```
#laoding from sklearn
from sklearn.datasets import load_boston
import pandas as pd

dataset = load_boston()

# Read the DataFrame, first using the feature data
df = pd.DataFrame(dataset.data, columns = dataset.feature_names)

# Add a target column, and fill it with the target data
df['MEDV'] = dataset.target

df.head()
```

[7]:
```
      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  MEDV
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
```

[8]:
```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size = 0.3)

X_train = train.drop('MEDV', axis=1)
y_train = train['MEDV']

X_test = test.drop('MEDV', axis = 1)
y_test = test['MEDV']
```

[9]:
```
#Feature Scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(X_train)
#reverting back to df
```

```
X_train = pd.DataFrame(x_train_scaled, columns=X_train.columns)

x_test_scaled = scaler.fit_transform(X_test)
X_test = pd.DataFrame(x_test_scaled, columns=X_test.columns)
```

[10]:
```
df.shape
```

[10]: (506, 14)

5 Building a simple linear regression model

To build a simple regression model, we need to select one explanatory variable. In this dataset, there are 13 explanatory variables. How to select one of them?

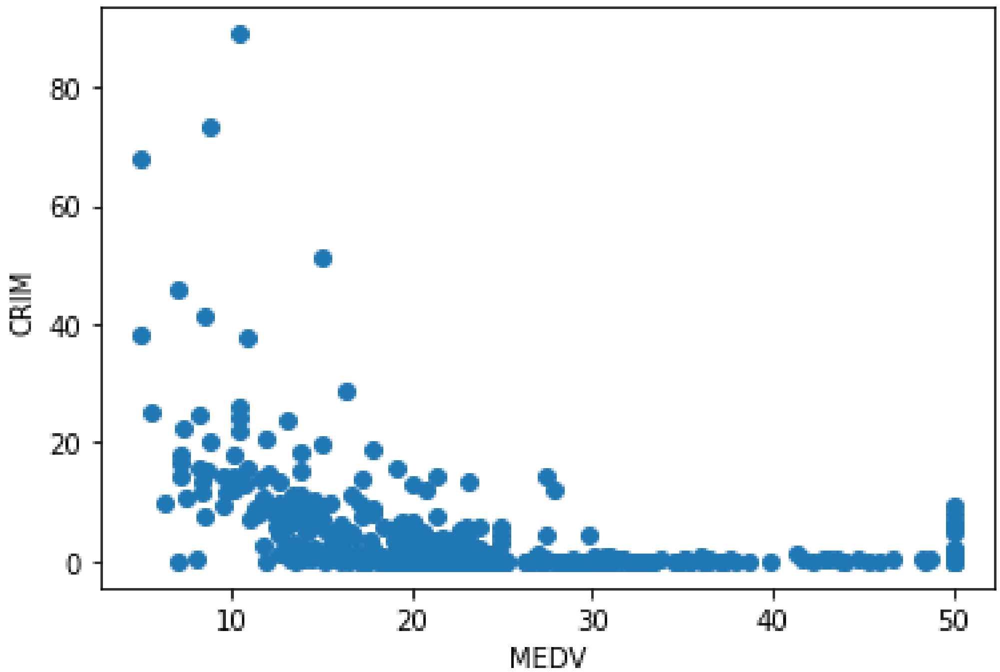
There are two important things we need to explore to decide the variable to use for developing a simple regression model. 1. The relationship between target (MEDV in this case) and explanatory variable should be linear. 1. The variable should be highly correlated to target.

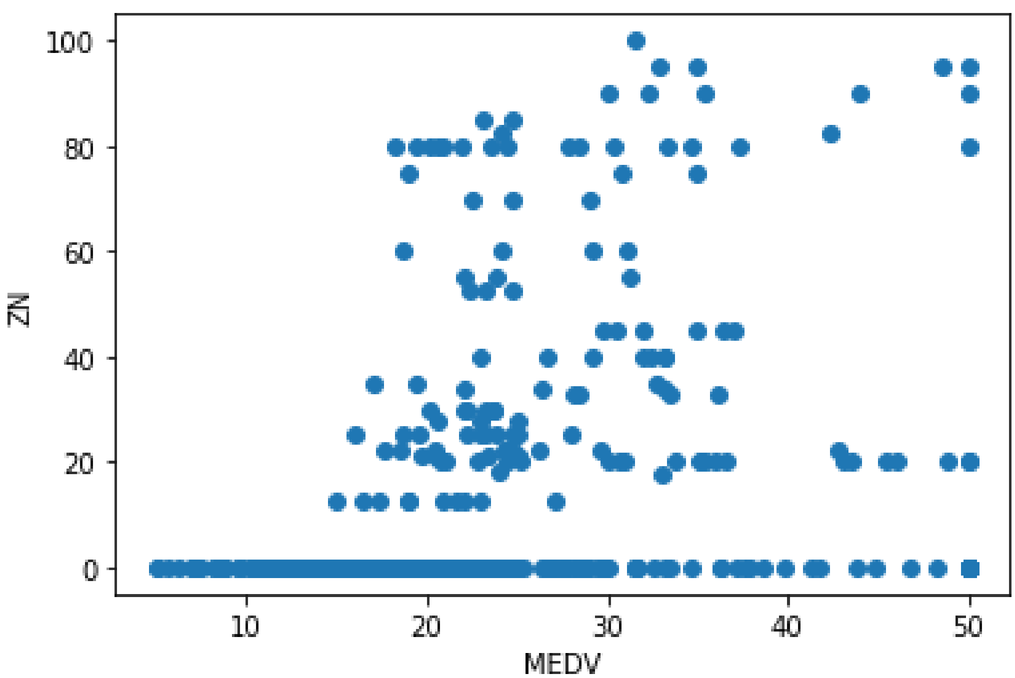
We will now see how the above two can be observed using python libraries.

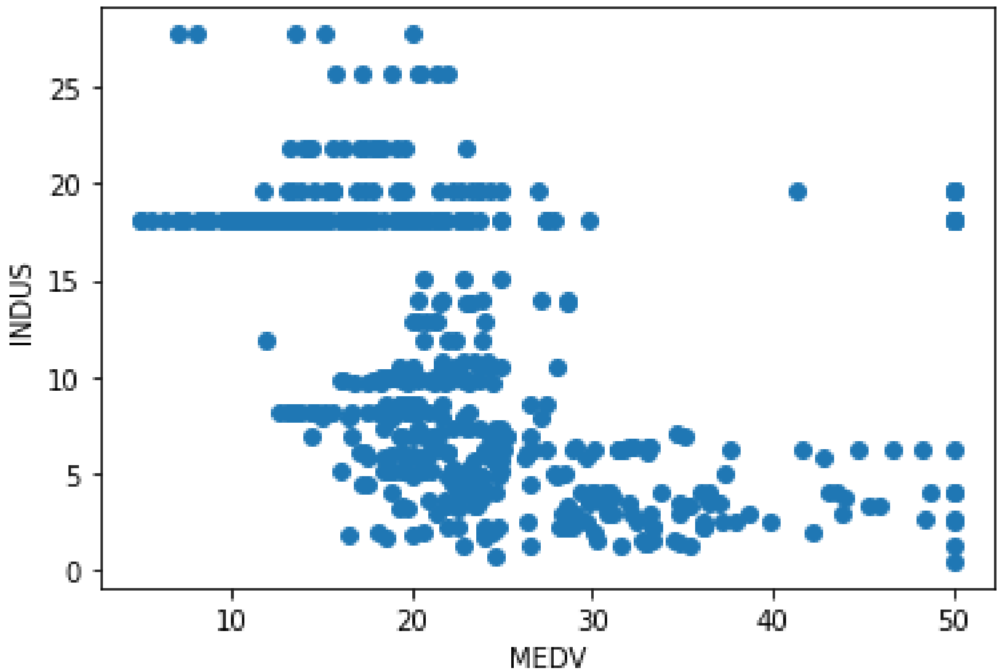
5.1 Relationship between target and explanatory variables

Exploratory data analysis (EDA) is an important and recommended first step prior to the training of a machine learning model. We will use a simple technique from the graphical EDA toolbox to visually detect the relationship between target and explanatory feature.

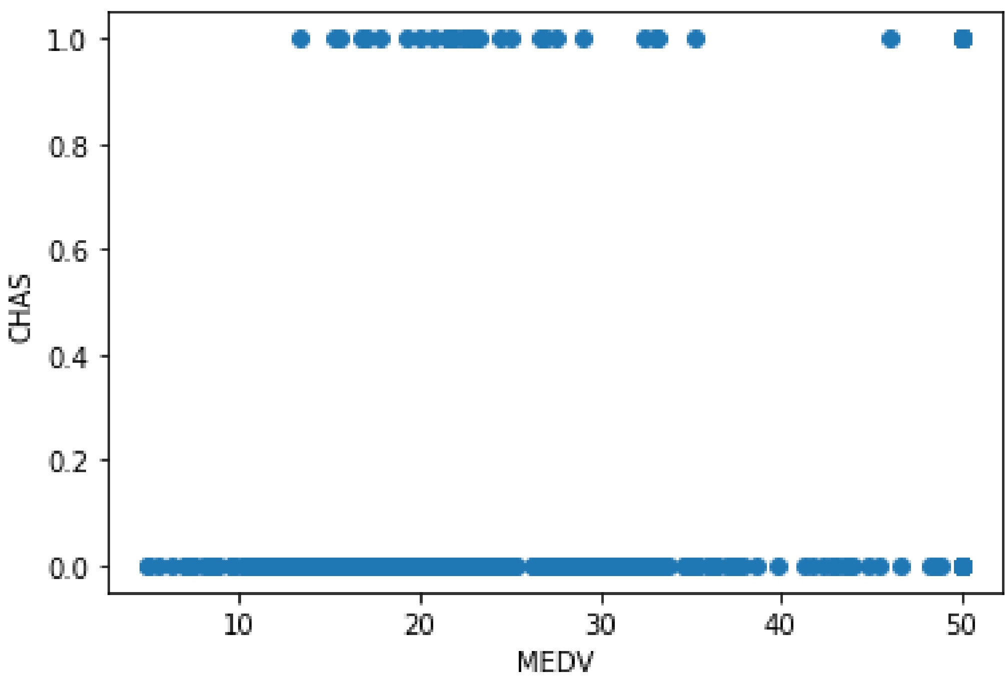
[11]:
```
import matplotlib.pyplot as plt



for column in df.columns:
    if column != 'MEDV':
        fig, ax = plt.subplots()
        ax.set_xlabel('MEDV')
        ax.set_ylabel(column)

        ax.scatter(df['MEDV'], df[column])
        plt.show()
```


CRIM
MEDV
0
20
40
60
80
10
20
30
40
50

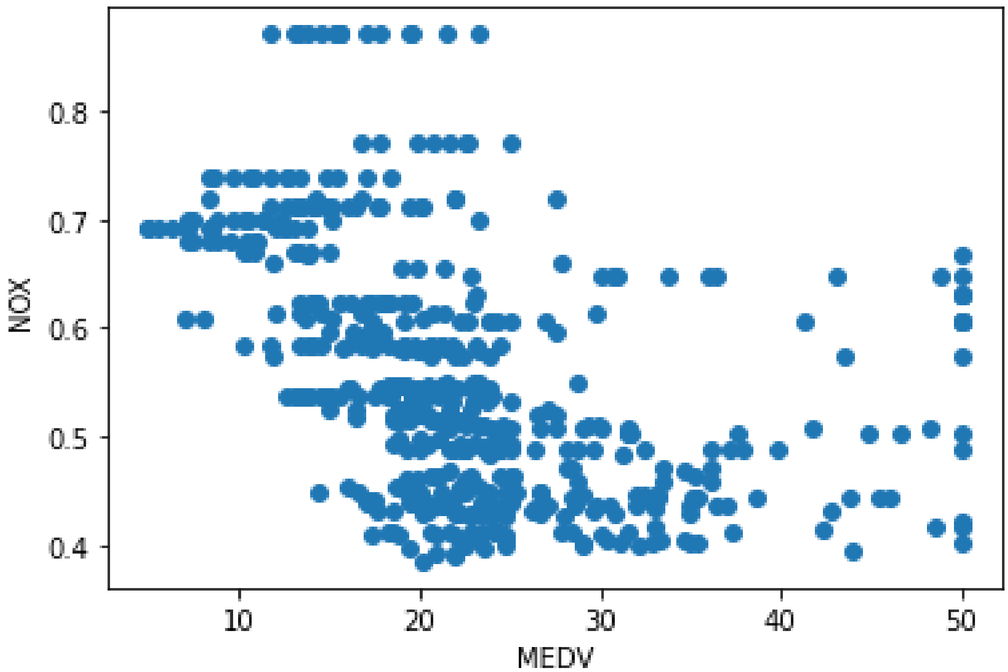

ZN
MEDV
0
20
40
60
80
100
10
20
30
40
50



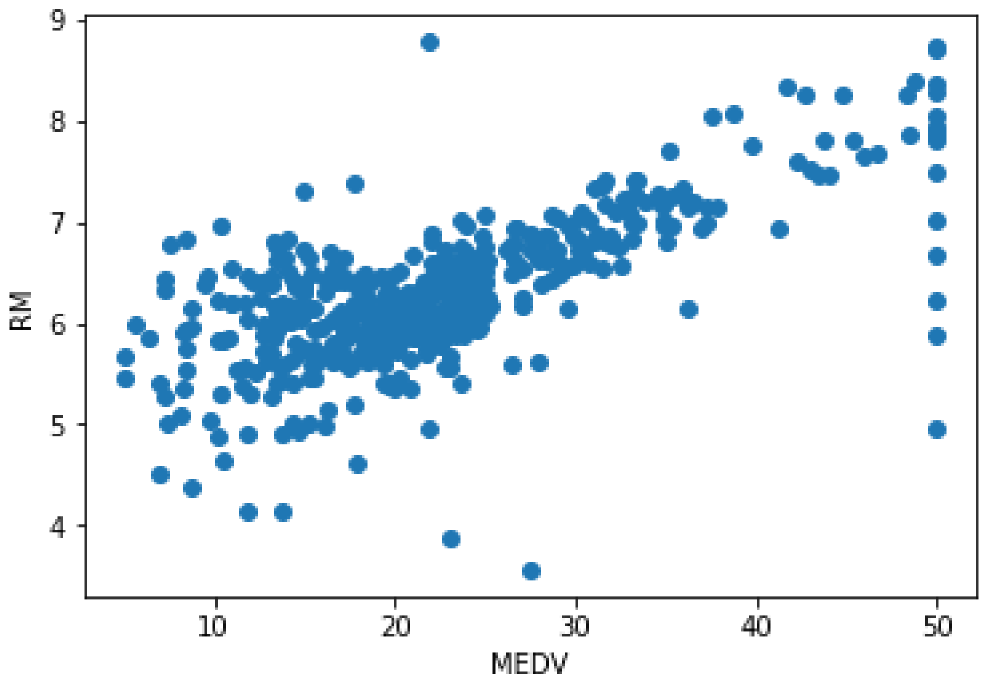
INDUS
MEDV
0
5
10
15
20
25
10
20
30
40
50



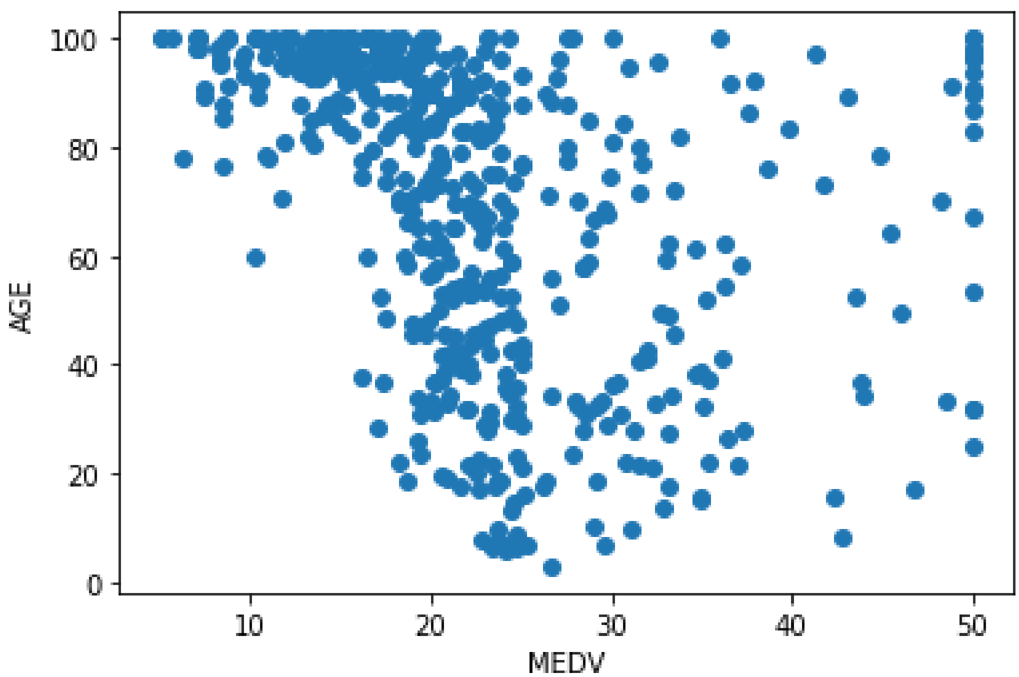
CHAS
MEDV
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40
50

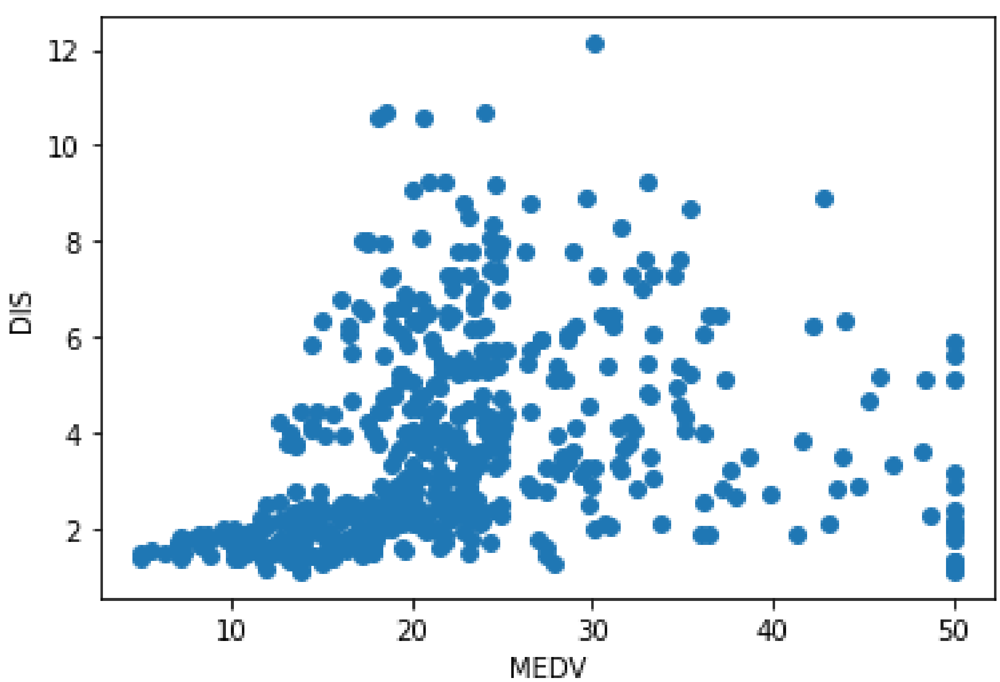


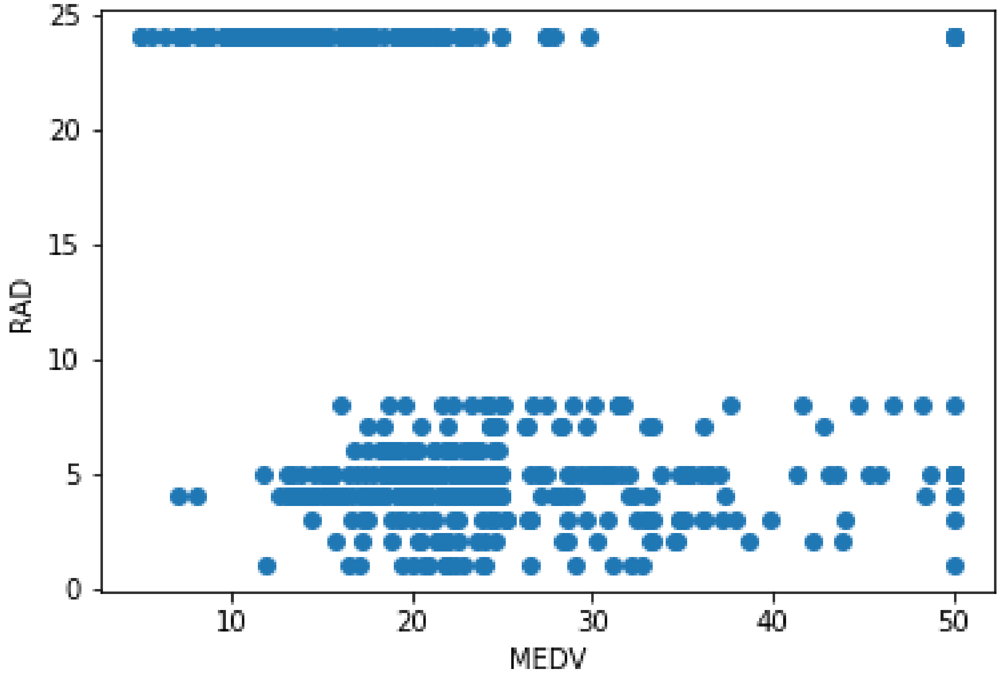
NOX
MEDV
0.4
0.5
0.6
0.7
0.8
10
20
30
40
50



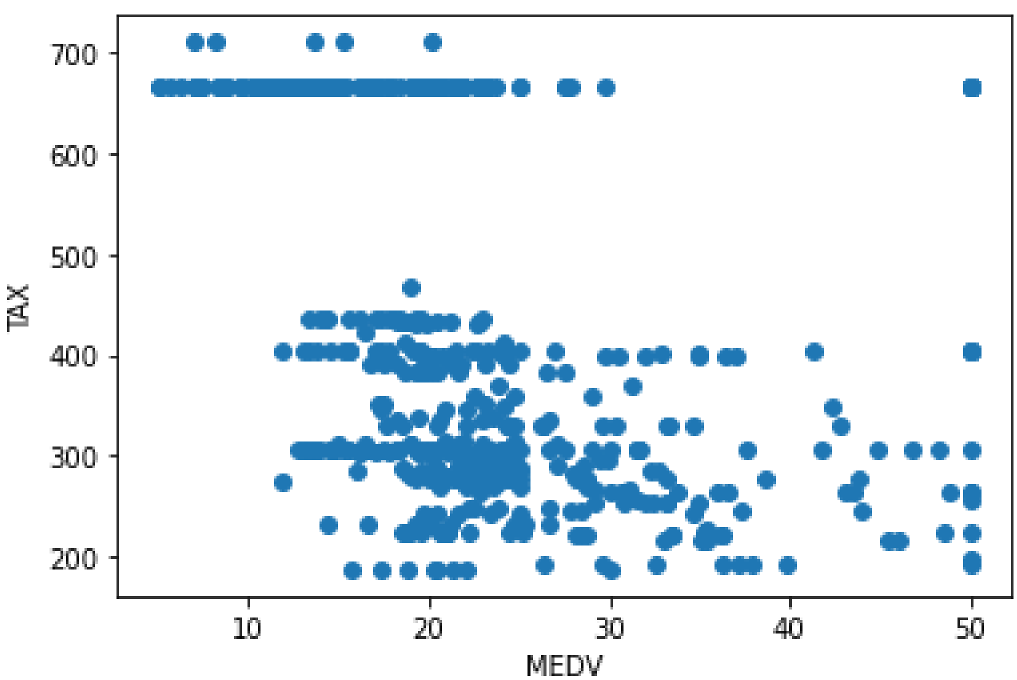
RM
MEDV
4
5
6
7
8
9
10
20
30
40
50


AGE
MEDV
0
20
40
60
80
100
10
20
30
40
50

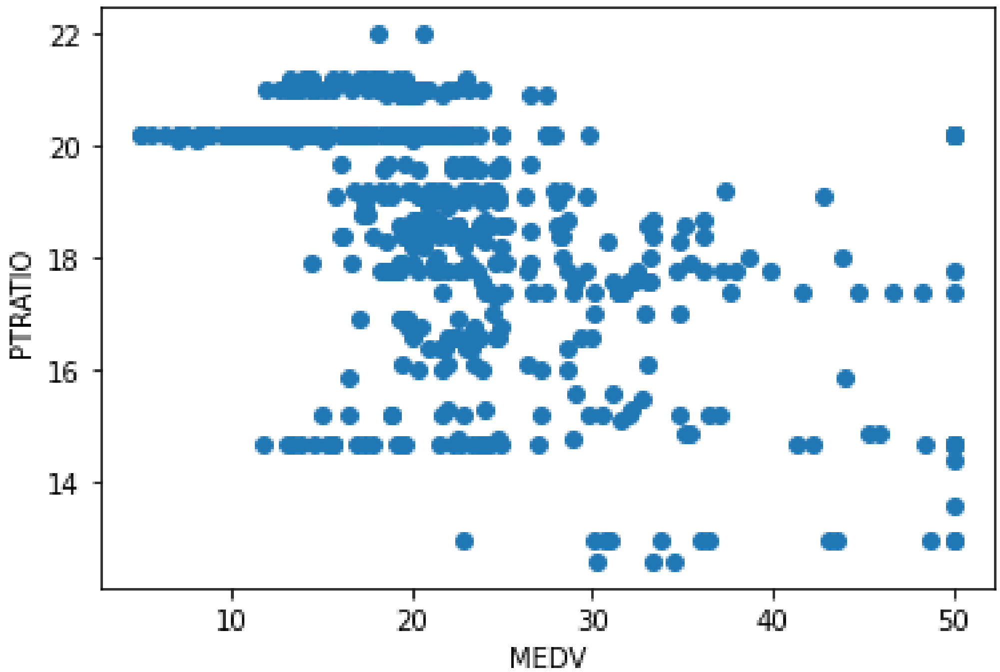

DIS
MEDV
2
4
6
8
10
12
10
20
30
40
50

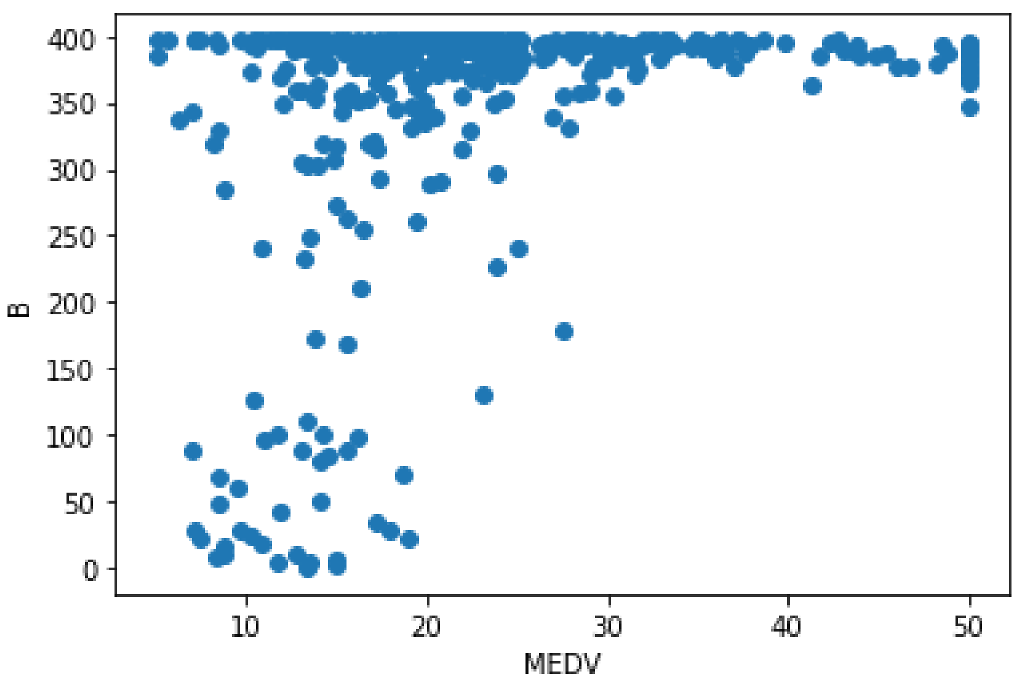


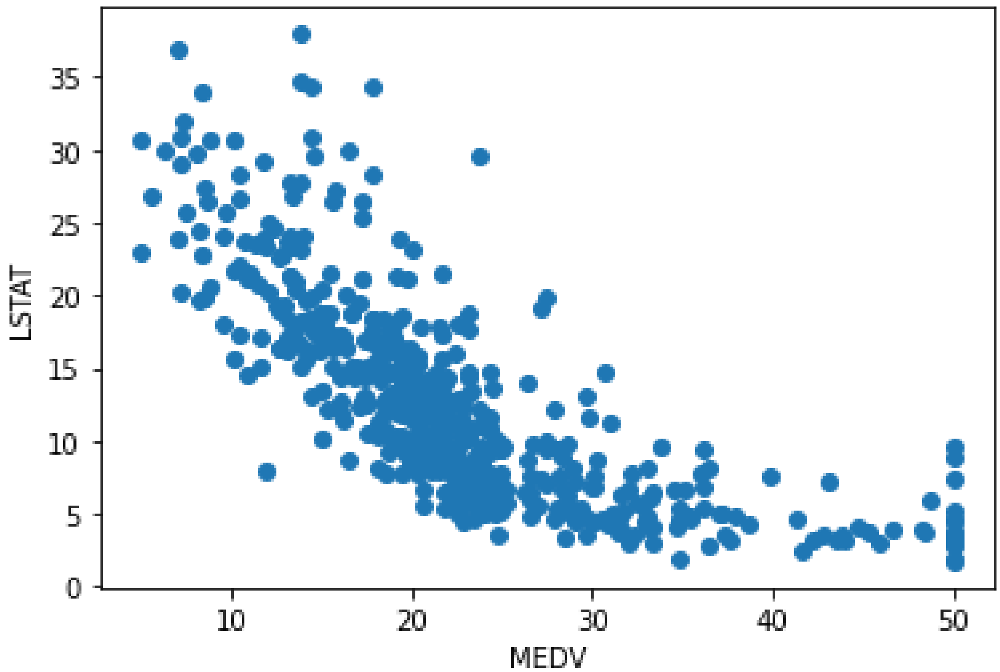
RAD
MEDV
0
5
10
15
20
25
10
20
30
40
50



TAX
MEDV
200
300
400
500
600
700
10
20
30
40
50


PTRATIO
MEDV
22
20
18
16
14
10
20
30
40
50


B
MEDV
400
350
300
250
200
150
100
50
0
10
20
30
40
50



Using the above scatter plots, we can see that there is a linear relationship between RM and MEDV. Also some degree of linearity exists between MEDV and LSTAT.

5.2 Correlation between taget and explanatory variables

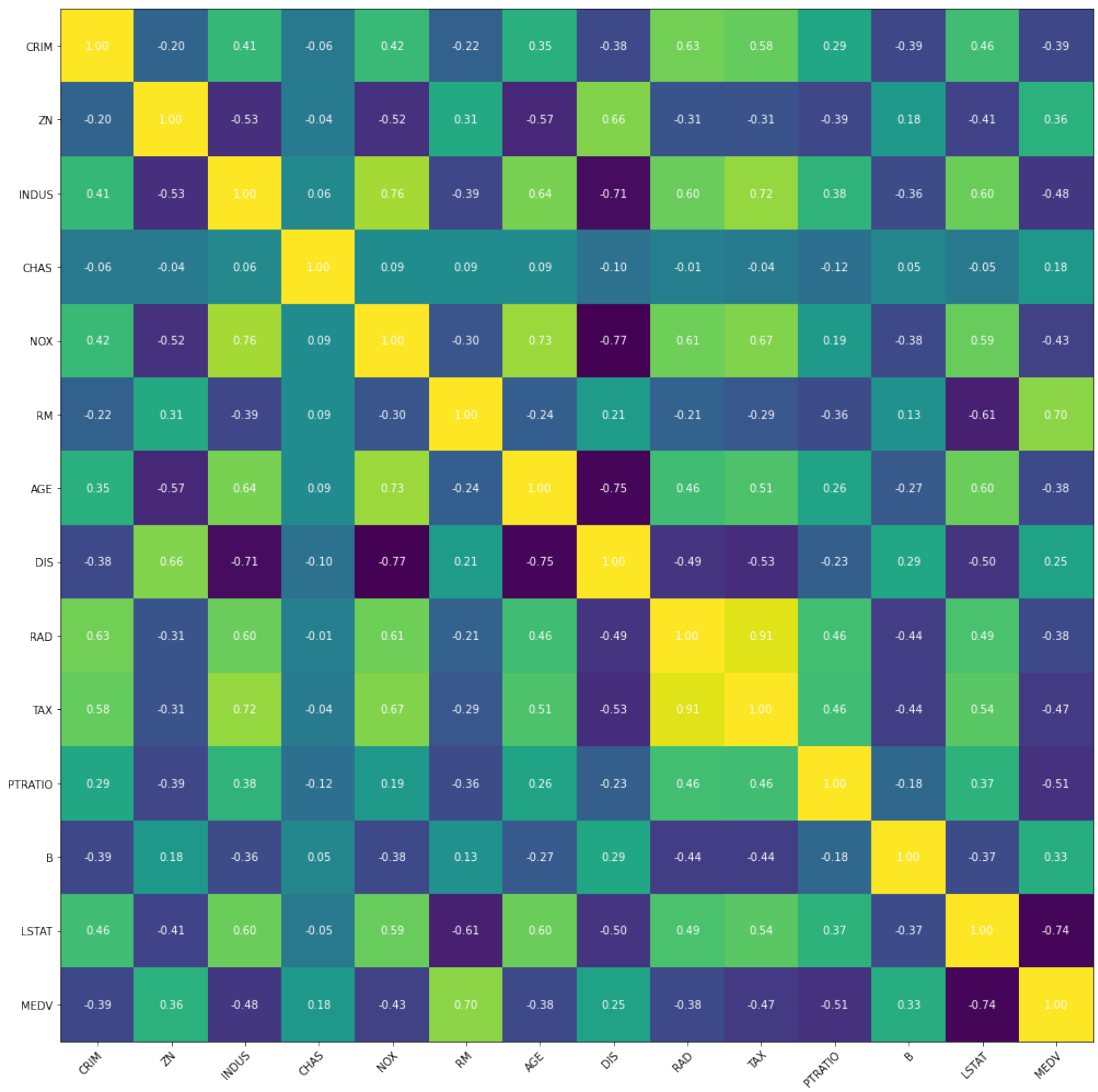
Correlation can be investigated using correlation matrix which is a square matrix that contains the Pearson product-moment correlation coefficient (often abbreviated as Pearson's r), which measures the linear dependence between pairs of features.

The correlation coefficients are in the range –1 to 1. Two features have a perfect positive correlation if r = 1, no correlation if r = 0, and a perfect negative correlation if r = –1.

In the following code example, we will use NumPy's corrcoef function and we will use heatmap function to plot the correlation matrix array as a heat map.

[12]:
```
import numpy as np

cm = np.corrcoef(df.values.T)

fig, ax = plt.subplots(figsize=(len(df.columns),len(df.columns)))
im = ax.imshow(cm)

# We want to show all ticks...
ax.set_xticks(np.arange(len(df.columns)))
ax.set_yticks(np.arange(len(df.columns)))
```

```
ax.set_xticklabels(df.columns)
ax.set_yticklabels(df.columns)

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right", rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
for i in range(len(df.columns)):
    for j in range(len(df.columns)):
        text = ax.text(j, i, f'{cm[i, j]:.2f}', ha="center", va="center",␣
↪color="w")

fig.tight_layout()
plt.show()
```



To fit a linear regression model, we are interested in those features that have a high correlation with our target variable, MEDV. Looking at the correlation matrix, we see that MEDV shows the largest correlation with the LSTAT variable (-0.74); and the correlation between RM and MEDV is also relatively high (0.70).

Given the linear relationship between RM and MEDV and high correlation from the matrix, RM seems to be a good choice for developing a simple linear regression model.

5.3 Model training

[17]:
```
from sklearn.linear_model import LinearRegression
slr = LinearRegression()

slr.fit(X_train[['RM']], y_train)
```

[17]:
```
LinearRegression()
```

6 Model Evaluation

[20]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

#evaluation
pred  = slr.predict(X_test[['RM']]) #make prediction on test set

error = sqrt(mean_squared_error(y_test,pred)) #calculate rmse

print(f'RMSE value = {error}')
```

```
RMSE value = 7.514850831549924
```

[25]:
```
#slope and intercept
print(f'Slope: {slr.coef_[0]:.3f}')
print(f'Intercept: {slr.intercept_:.3f}')
```

```
Slope: 46.476
Intercept: -1.840
```