boston-housing-data-analysis-day2

July 9, 2023

0.1 Small Project

Build a linear regression model to predict housing prices based on a given dataset.

- **Step 1:** Load and explore the dataset, including visualizing the features and target variable.
- **Step 2:** Split the dataset into training and testing sets.
- **Step 3:** Preprocess the data by handling missing values and performing feature scaling.
- **Step 4:** Train a linear regression model on the training data.
- **Step 5:** Evaluate the model's performance on the testing data using metrics such as mean squared error (MSE) or R-squared.

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; 0 otherwise).

nox: Nitrogen oxides concentration (parts per 10 million).

rm: Average number of rooms per dwelling.

age: Proportion of owner-occupied units built prior to 1940.

dis: Weighted mean of distances to five Boston employment centres.

rad: Index of accessibility to radial highways.

tax: Full-value property-tax rate per $10,000.

ptratio: Pupil-teacher ratio by town.

black: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.

lstat: Lower status of the population (percent).

medv: Median value of owner-occupied homes in $1000s.

0.2 Load Data

[7]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("train.csv")
df
```

[7]:
```
      ID     crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0      1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1      2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2      4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
3      5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
4      7  0.08829  12.5   7.87     0  0.524  6.012  66.6  5.5605    5  311
..   …       …    …     …    …      …    …    …     …  …  …
328  500  0.17783   0.0   9.69     0  0.585  5.569  73.5  2.3999    6  391
329  502  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
330  503  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
331  504  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
332  506  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio   black  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       18.7  394.63   2.94  33.4
3       18.7  396.90   5.33  36.2
4       15.2  395.60  12.43  22.9
..       …      …    …    …
328     19.2  395.77  15.10  17.5
329     21.0  391.99   9.67  22.4
330     21.0  396.90   9.08  20.6
331     21.0  396.90   5.64  23.9
332     21.0  396.90   7.88  11.9

[333 rows x 15 columns]
```

[8]:
```
df.head(10)
```

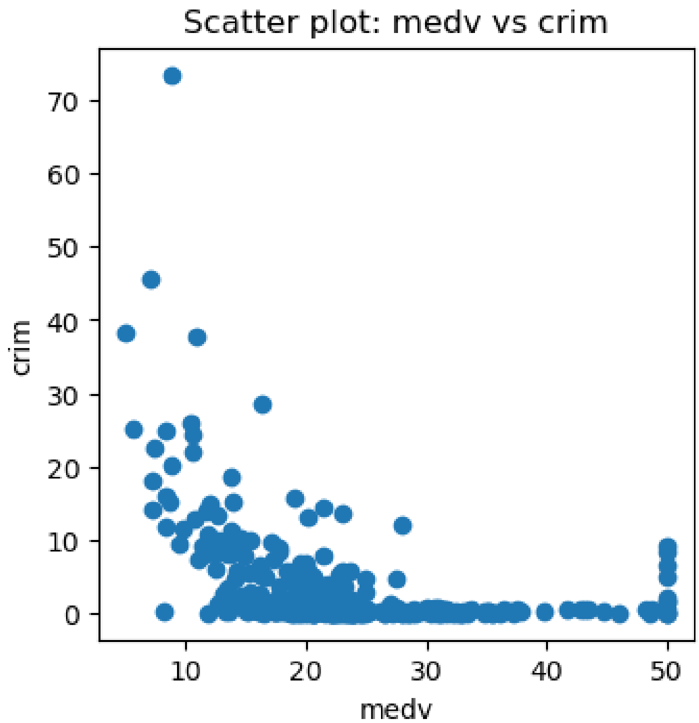
[8]:
```
   ID     crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0   1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1   2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2   4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
3   5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
4   7  0.08829  12.5   7.87     0  0.524  6.012  66.6  5.5605    5  311
5  11  0.22489  12.5   7.87     0  0.524  6.377  94.3  6.3467    5  311
6  12  0.11747  12.5   7.87     0  0.524  6.009  82.9  6.2267    5  311
7  13  0.09378  12.5   7.87     0  0.524  5.889  39.0  5.4509    5  311
8  14  0.62976   0.0   8.14     0  0.538  5.949  61.8  4.7075    4  307
9  15  0.63796   0.0   8.14     0  0.538  6.096  84.5  4.4619    4  307

   ptratio   black  lstat  medv
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     18.7  394.63   2.94  33.4
3     18.7  396.90   5.33  36.2
```

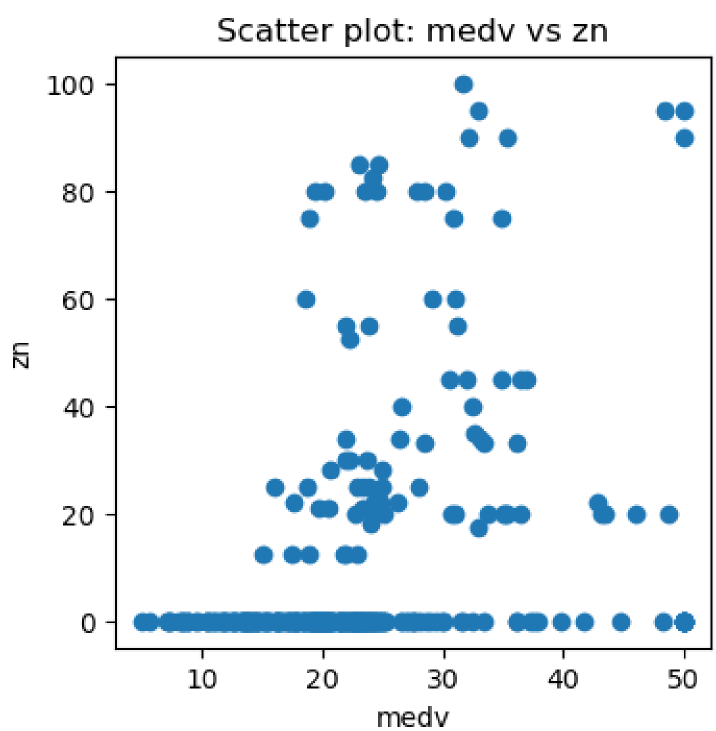
```
4     15.2  395.60  12.43  22.9
5     15.2  392.52  20.45  15.0
6     15.2  396.90  13.27  18.9
7     15.2  390.50  15.71  21.7
8     21.0  396.90   8.26  20.4
9     21.0  380.02  10.26  18.2
```

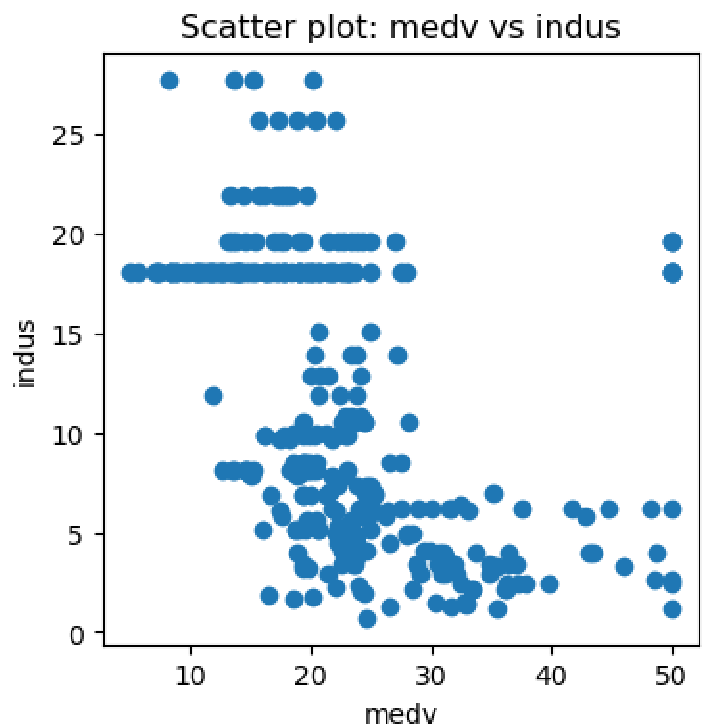
0.3 Visualize Data using Scatter Charts

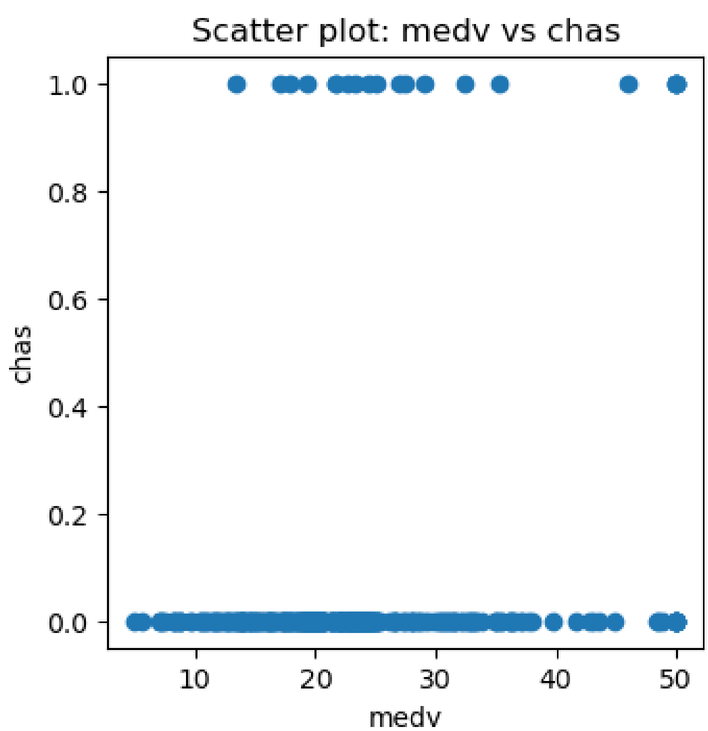
[137]:
```python
# df.scatter()
# plt.show()
# plt.tight_layout()

# plt.scatter(x=df['crim'], y=df['medv'])
# plt.xlabel("Crim")
# plt.ylabel("Medv")

for column in df.columns[1:]:
    plt.figure(figsize=(4,4))
    plt.scatter(df['medv'], df[column],  cmap='Spectral', alpha=1)
    plt.xlabel('medv')
    plt.ylabel(column)
    plt.title(f'Scatter plot: medv vs {column}')
    plt.show()


# for col in df.columns:
#     plt.scatter(x=df['medv'], y=col)
```

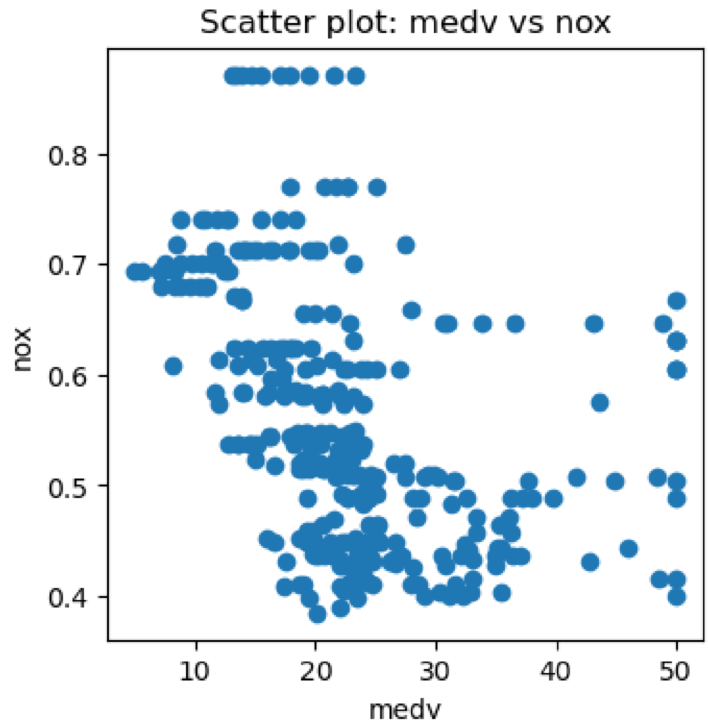
```
C:\Users\Hp\AppData\Local\Temp\ipykernel_15644\105836776.py:11: UserWarning: No
data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(df['medv'], df[column],  cmap='Spectral', alpha=1)
```

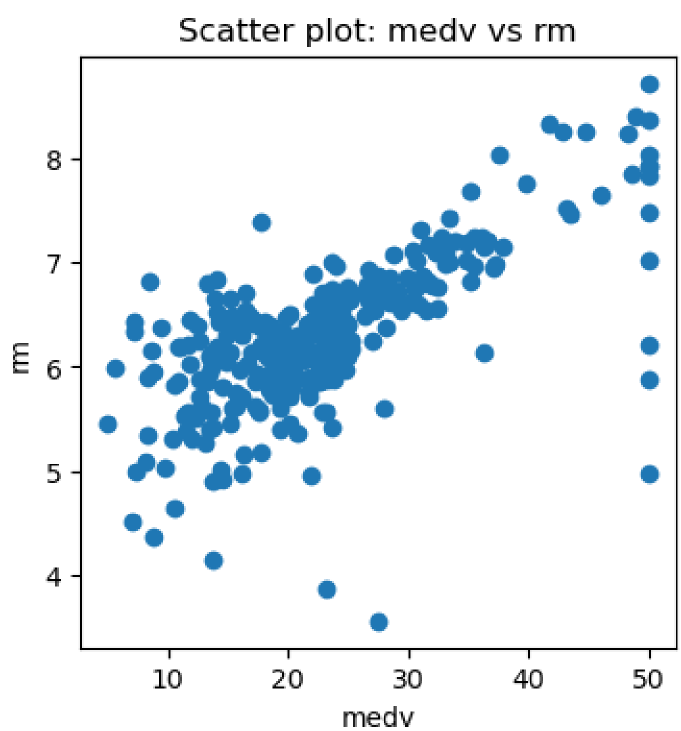

Scatter plot: medv vs crim
crim
medv
0
10
20
30
40
50
60
70
10
20
30
40
50

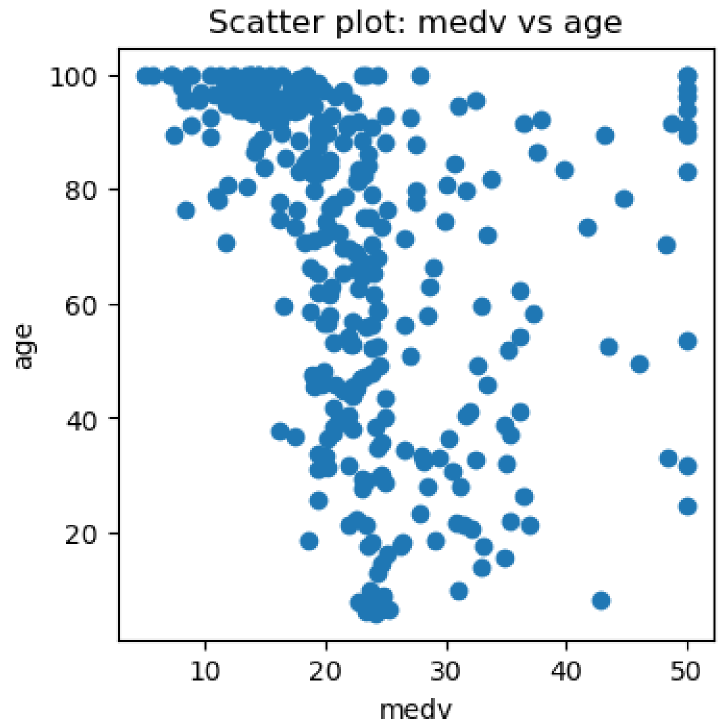

Scatter plot: medv vs zn
zn
medv
0
20
40
60
80
100
10
20
30
40
50

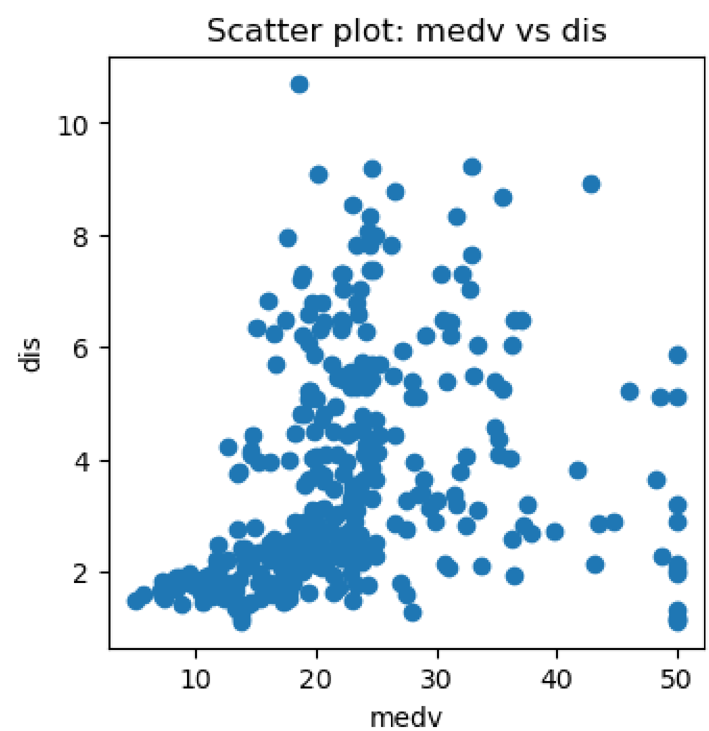

Scatter plot: medv vs indus
indus
medv
0
5
10
15
20
25
10
20
30
40
50

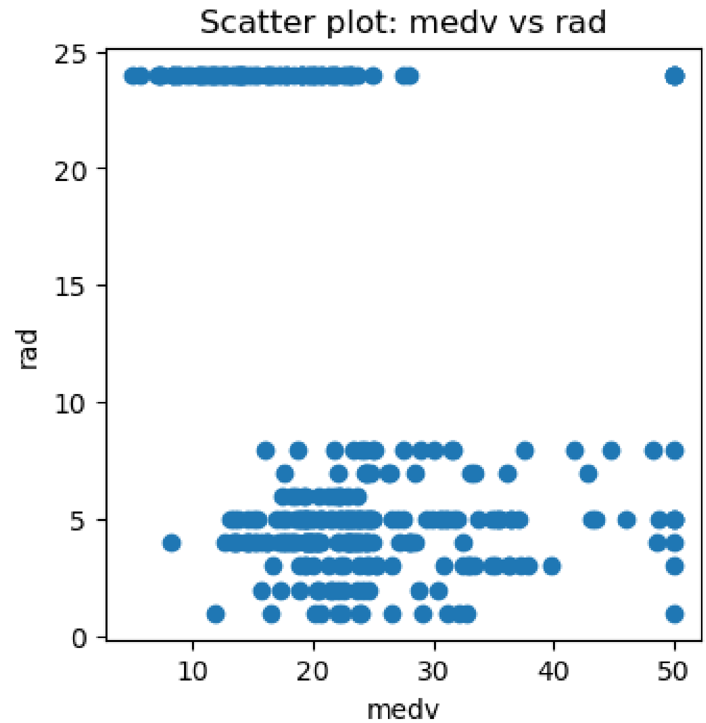

Scatter plot: medv vs chas
chas
medv
0.0
0.2
0.4
0.6
0.8
1.0
10
20
30
40
50

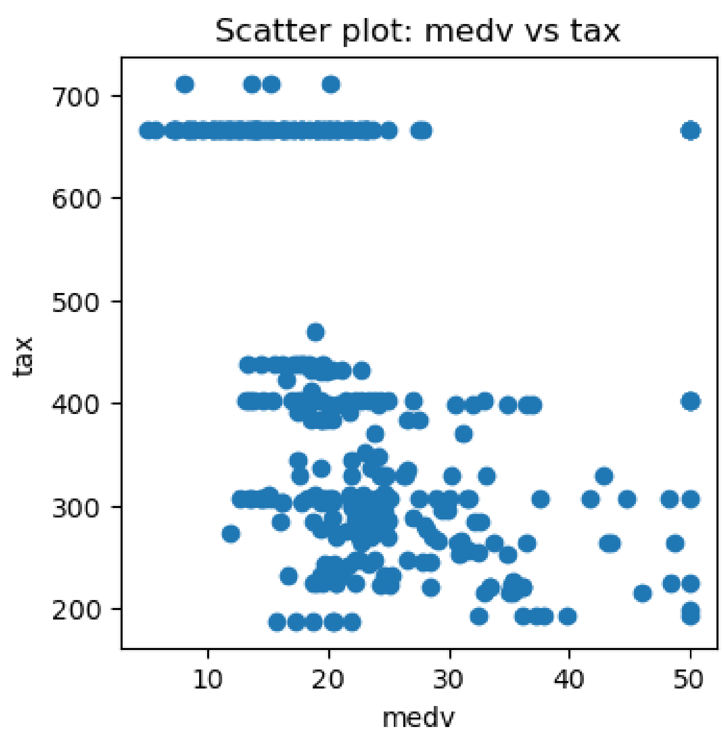

Scatter plot: medv vs nox
nox
0.4
0.5
0.6
0.7
0.8
10
20
30
40
50
medv

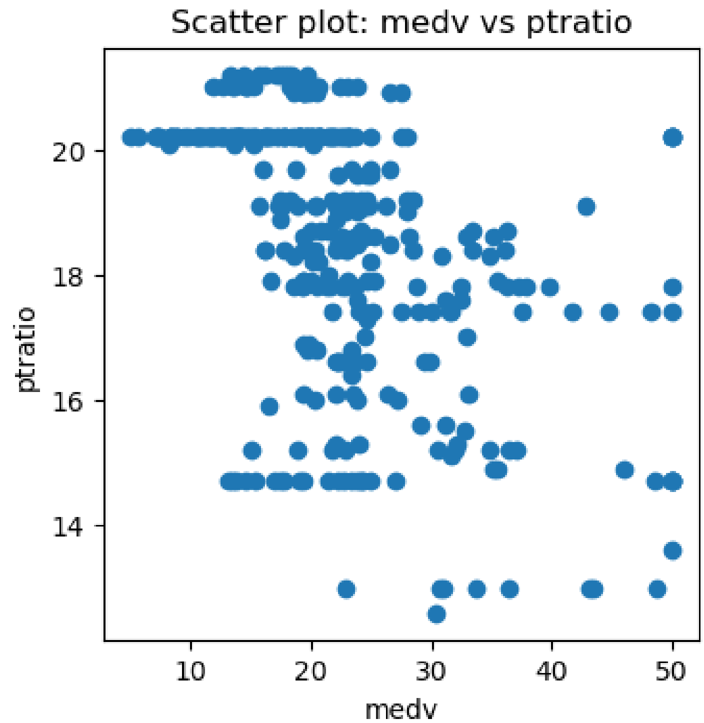

Scatter plot: medv vs rm
rm
4
5
6
7
8
10
20
30
40
50
medv

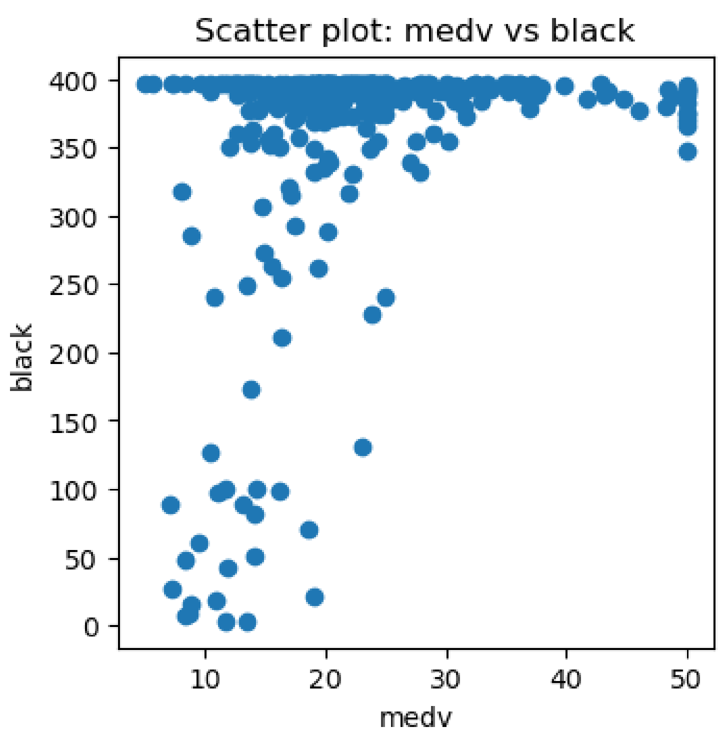

Scatter plot: medv vs age
age
medv
100
80
60
40
20
10
20
30
40
50

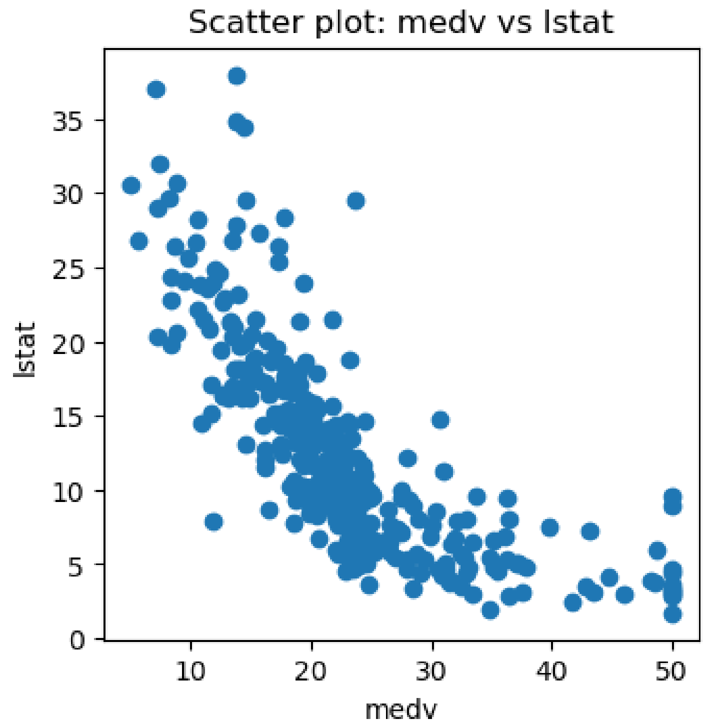

Scatter plot: medv vs dis
dis
medv
10
8
6
4
2
10
20
30
40
50

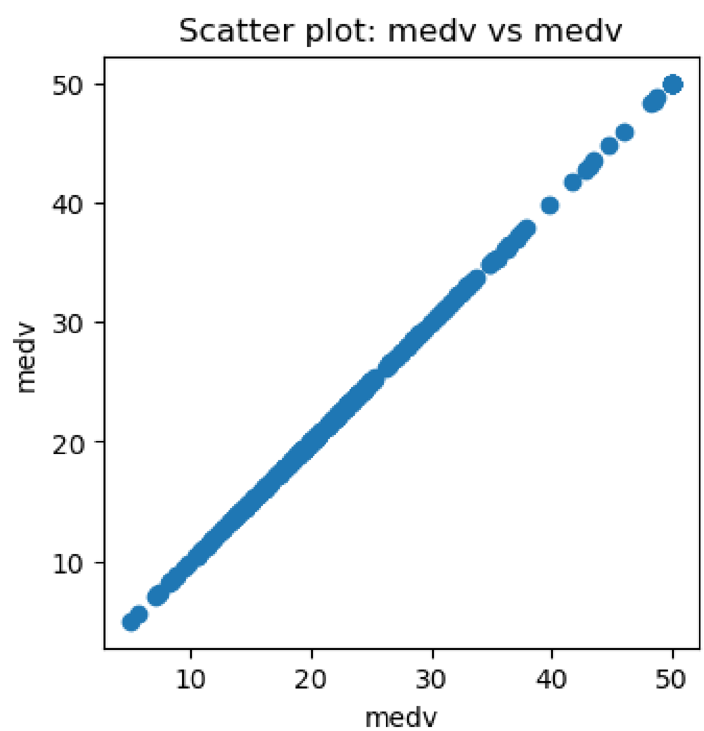

Scatter plot: medv vs rad
rad
medv
0
5
10
15
20
25
10
20
30
40
50


Scatter plot: medv vs tax
tax
medv
200
300
400
500
600
700
10
20
30
40
50


Scatter plot: medv vs ptratio
ptratio
medv
20
18
16
14
10
20
30
40
50


Scatter plot: medv vs black
black
medv
400
350
300
250
200
150
100
50
0
10
20
30
40
50


Scatter plot: medv vs lstat
lstat
medv
0
5
10
15
20
25
30
35
10
20
30
40
50


Scatter plot: medv vs medv
medv
medv
10
20
30
40
50
10
20
30
40
50

0.4 Fill Empty Data

[111]:
```
df.medv.fillna(df.medv.mean(), inplace=True)

df
```

[111]:
```
      ID     crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0      1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1      2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2      4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
3      5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
4      7  0.08829  12.5   7.87     0  0.524  6.012  66.6  5.5605    5  311
..   …       …    …     …    …      …    …     …      …   …   …
328  500  0.17783   0.0   9.69     0  0.585  5.569  73.5  2.3999    6  391
329  502  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
330  503  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
331  504  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
332  506  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio   black  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       18.7  394.63   2.94  33.4
3       18.7  396.90   5.33  36.2
4       15.2  395.60  12.43  22.9
..       …       …     …    …
328     19.2  395.77  15.10  17.5
329     21.0  391.99   9.67  22.4
330     21.0  396.90   9.08  20.6
331     21.0  396.90   5.64  23.9
332     21.0  396.90   7.88  11.9

[333 rows x 15 columns]
```

0.5 Separate Training and Testing Data

[118]:
```
from sklearn.model_selection import train_test_split

# Split arrays or matrices into random train and test subsets
x = df.drop('medv', axis=1)
y = df['medv']

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=25)

y_train
```

[118]:
```
204    33.4
40     16.0
23     34.9
100    17.8
126    39.8
       …
255     7.4
317    23.7
143    22.6
318    21.8
132    34.9
Name: medv, Length: 266, dtype: float64
```

0.6 Feature Scaling

[120]:
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train_scaled
```

[120]:
```
array([[ 0.39386662, -0.42470702,  0.90238868, …, -0.02218378,
         0.43006366, -0.83126475],
       [-1.29151012, -0.41236258,  0.56749452, …,  0.59645738,
         0.21981071,  0.25378715],
       [-1.43597098, -0.42999823,  2.66058302, …, -0.06977156,
         0.41576378, -1.44254242],
       …,
       [-0.30092134, -0.41686853, -0.47904973, …,  0.07299179,
         0.43006366, -0.23223986],
       [ 1.61146533, -0.07442652, -0.47904973, …,  0.83439629,
         0.38559996, -0.29350377],
       [-0.41098676, -0.42360001,  1.40472992, …, -1.54499279,
         0.43006366, -0.97829813]])
```

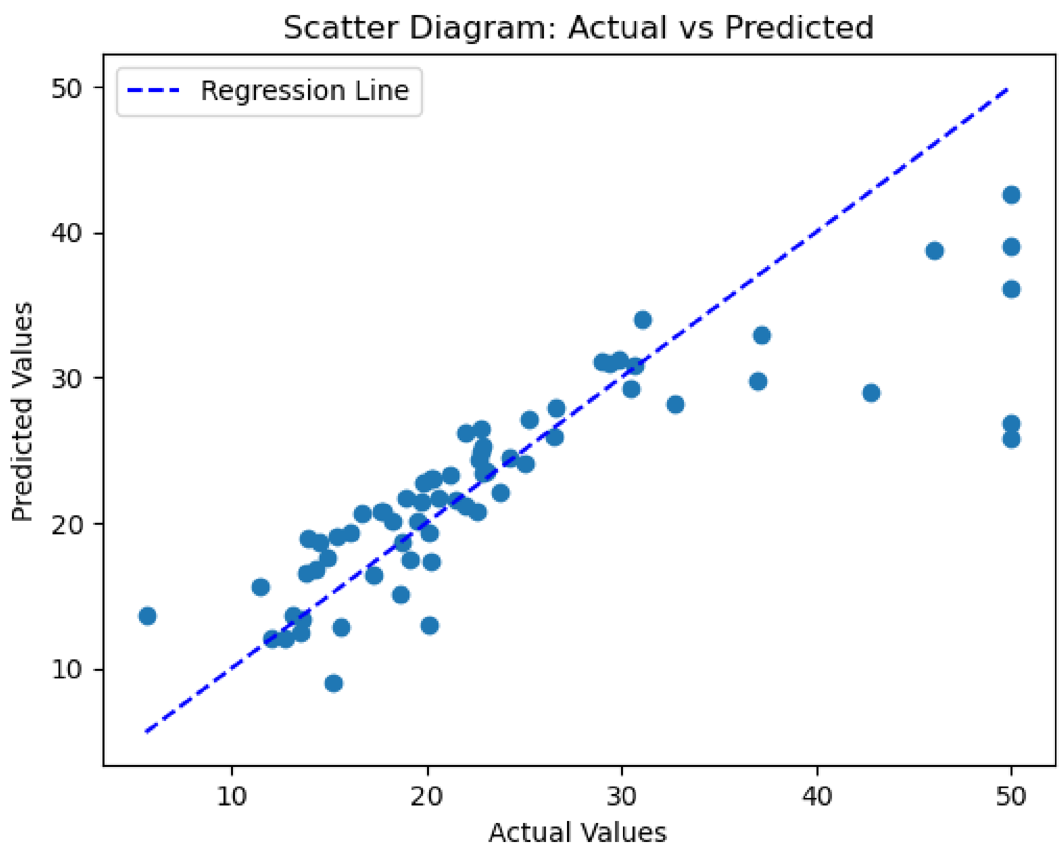
0.7 Linear Regression and Visualing Output

[128]:
```python
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

[128]:
```
LinearRegression()
```

[133]:

```
y_pred = model.predict(X_test_scaled)

# Plot the scatter diagram
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'b--',␣
↪label='Regression Line')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Scatter Diagram: Actual vs Predicted')
plt.legend()
plt.show()
```



0.8 Mean Square and R-squared

[136]:

```
# Mean Square and R-squared
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import math
```

```
mean_sqrt = math.sqrt(mean_squared_error(y_test, y_pred))
r_sqrt = r2_score(y_test,y_test)

print("Mean Squared {}".format(mean_sqrt))
print("R-Squared {}".format(r_sqrt))
```

```
Mean Squared 5.8183976341242385
R-Squared 1.0
```

[ ]:

```

```