Uczenie liniowej hipotezy dla problemu przewidywania cen nieruchomości (Housing Data)

1. Wprowadzenie teoretyczne

Uczenie nadzorowane

- do systemu przekazywane są dane wejściowe z oczekiwanymi rozwiązaniami nazywanymi etykietami. Celem systemu jest znalezienie związków zachodzących w danych i poprawne wygenerowanie predykcji na nieznanych dotąd nieoznaczonych próbkach.

Atrybuty

- (cechy) to pewne cechy charakteryzujące właściwości rozważanych obiektów.

Korelacja

- rodzaj zależności pomiędzy zmiennymi losowymi, z których każda wyznaczona jest przez pewną cechę, ze względu na którą bada się daną populację.

Zbiór uczący

- to zbiór przykładów, obserwacji, próbek służący do wytrenowania klasyfikatora.

Klasa

- to zbiór obiektów charakteryzujących się pewnymi wspólnymi właściwościami.

Odległość euklidesowa

$d(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Odległość pomiędzy dwoma punktami $p_1 = (x_1, y_1)$ oraz $p_2 = (x_2, y_2)$

Przeuczenie, przetrenowanie

- Nadmierne dopasowanie w uczeniu maszynowym występuje, gdy model zbyt dobrze pasuje do danych treningowych i w rezultacie nie może dokładnie przewidzieć niezamierzonych danych testowych. Innymi słowy, model po prostu zapamiętał konkretne wzorce i szum w danych treningowych, ale nie jest wystarczająco elastyczny, aby przewidywać rzeczywiste dane.

Wykres rozrzutu

- Jest to graficzna interpretacji korelacji pomiędzy cechami.

Selekcja cech

- Celem selekcji cech jest wybrane ze zbioru cech takich, które zapewniają nam możliwie najlepszą klasyfikację. W zbiorze cech mogą wystąpić cechy redundantne, czyli niosące identyczną informację jak

istniejące już cechy lub cechy wprowadzające szum, obniżające skuteczność klasyfikacji.

Selekcja ma wpływ na:

- poprawę wyników predykcji,
- zmniejszenie wymagań obliczeniowych,
- zmniejszenie wymagań odnośnie gromadzenia danych,
- redukcję kosztów przyszłych pomiarów,
- poprawę jakości danych.

Normalizacja zmiennych

- zabieg który stosuje się w przypadku gdy zmienne o większej skali mogą zdominować te o mnie mniejszej. Działanie: Dla każdej zmiennej wejściowej X_j , najmniejszą wartość x_j, min zmień na 0, największą x_j, max na 1, a pozostałe wyznaczam proporcjonalnie.

$$x_{ij} -> \frac{x_{ij} - x_j,min}{x_j,max - x_j,min}$$

Wadą tej metody jest brak odporności na wartości odstające.

Standaryzacja zmiennych -

Dla każdej zmiennej wejściowej X_j , wyznacz wartość średnią i odchylenie standardowe: Wartość średnia:

$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$

Odchylenie standardowe:

$S_j = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x_j})^2}$

następnie zamieniamy: $x_{ij} -> \frac{x_{ij} - \bar{x_j}}{S_j}$

Przykład:

Wartości: (1, 2, 3, 4, 5).

Średnia: x = 3

Odchylenie: s = 1.581

Nowe wartości: (−1.265, −0.632, 0, 0.632, 1.265)

Metoda ta jest znacznie odporniejsza na wartości odstające niż normalizacja. Jednak w przypadku zerowego odchylenia standardowego pojawiają się problemy.

Współczynnik determinacji (R2 score)

Informuje o tym, jaka część zmienności (wariancji) zmiennej objaśnianej w próbie pokrywa się z korelacjami ze zmiennymi zawartymi w modelu. Jest on miarą stopnia, w jakim model pasuje do próby. Współczynnik determinacji przyjmuje wartości z przedziału [0;1] jeśli w modelu występuje wyraz wolny, a do estymacji parametrów wykorzystano metodę najmniejszych kwadratów. Jego wartości najczęściej są wyrażane w procentach. Dopasowanie modelu jest tym lepsze, im wartość R^2 jest bliższa jedności. Wyraża się on wzorem:

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} >= 0$$

y_i - i - ta obserwacja zmiennej y

$\hat{y}_i$ -wartość teoretyczna zmiennej prognozowanej na podstawie modelu

$\bar{y}$ - średnia arytmetyczna empirycznych wartości zmiennej prognozowanej

Współczynnik zbieżności $\phi 2$

- opisuje jaka część danych nie jest wytłumaczona przez model. Im większe R^2 tym mniejsze $\phi 2$ – suma obu współczynników sumuje się do 1 co jest oczywiste: np. skoro model wyjaśnia 90% tzn. że nie wyjaśnia 10%.

Regresja liniowa

- jest najprostszym wariantem regresji w statystyce. Zakłada ona, że zależność pomiędzy zmienną objaśnianą a objaśniająca jest zależnością liniową. Tak jak w analizie korelacji, jeżeli jedna wartość wzrasta to druga wzrasta (dodatnia korelacji) lub spada (korelacja ujemna). W regresji liniowej zakłada się, że wzrostowi jednej zmiennej (predyktor, predyktory) towarzyszy wzrost lub spadek na drugiej zmiennej. Co więcej, nazwa regresji liniowej odnosi się, że funkcja regresji przyjmuje postać funkcji liniowej, czyli

$y = bx + a$

Aby wyznaczyć linię regresji, a tym samym wzór modelu regresji liniowej należy obliczyć współczynniki linii prostej, a i b. W tym celu wykorzystuje się metodę najmniejszych kwadratów błędu (opisana poniżej). Metoda ta dostarcza nam takich współczynników a i b, które powodują, że linia regresji jest najlepiej dopasowana do zebranych danych. Wracając do wzoru na linię prostą, analiza regresji oblicza: współczynnik b, zwany współczynnikiem regresji wartość a, zwaną wyrazem wolnym.

Gdy poprzez analizę regresji liniowej oszacujemy wzór regresji, wzór na linię prostą w modelu, czyli nasze współczynniki a i b, to będziemy mogli oszacować wartości zmiennej zależnej (zmiennej objaśnianej, Y) na podstawie wartości predyktora (zmienna objaśniająca, X) podstawiając odpowiednią wartość X do uzyskanego wzoru. Dlatego też mówimy, że analiza regresji służy do przewidywania wartości jednej zmiennej na podstawie innych. W tym przypadku modeli regresji wielorakiej, wielokrotnej, gdy mamy większą liczbę predyktorów stosujemy następujący wzór na linię regresji:

$$Y = b_1 x_1 + b_2 x_2 + .. + b_n x_n + a$$

gdzie:

b_1, b_2, b_n- są współczynnikami regresji wyliczonymi dla poszczególnych predyktorów w modelu

x_1, x_2, x_n - są wartościami predyktorów

Y - to zmienna objaśniana, zmienna zależna

a - to wyraz wolny

Graficzną interpretacją linii regresji dla dwóch predyktorów nie będzie już linia prosta lecz płaszczyzna w układzie trójwymiarowym.

MSE (z ang. mean square error)

- metryka określająca średnią kwadratową pomiędzy wartościami przewidywanymi a rzeczywistymi w zbiorze danych. Im niższe MSE, tym lepiej model pasuje do zbioru danych. Wzór:

$$MSE = \sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)}{n}$$

gdzie:

$\hat{y}_i$ - jest przewidywaną wartością dla i-tej obserwacji

y_i - jest obserwowaną wartością dla i-tej obserwacji

n- liczebność próbek

RMSE (z ang. root mean square error)

jest jedną z najczęściej stosowanych miar oceny jakości przewidywań. Pokazuje ona, jak bardzo prognozy odbiegają od zmierzonych wartości prawdziwych przy użyciu odległości euklidesowej.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)}{n}}$$

gdzie:

$\hat{y}_i$ - jest przewidywaną wartością dla i-tej obserwacji

y_i - jest obserwowaną wartością dla i-tej obserwacji

n- liczebność próbek

Jak widać RMSE to pierwiastek z MSE.

2. Model regresji liniowej

Niech dany będzie zbiór danych zaobserwowanych

$$(y_i, y_{i1}, \ldots, y_{ip})_{i=1}^{n}.$$

Model regresji liniowej zakłada, że istnieje liniowa (afiniczna) relacja pomiędzy zmienną zależną y_i, a wektorem p x 1 regresorów x_i . Zależność ta jest modelowana przez uwzględnienie składnika losowego (błędu) i , który jest zmienną losową. Dokładniej, model ten jest postaci:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i = x_i^T \beta_{ip} + \epsilon_i$$

dzie: ^{T} oznacza transpozycję, tj. x_i^T jest iloczynem skalarnym wektorów x_i oraz Powyższe n n równań można zapisać w sposób macierzowy:

$y = X\beta + \epsilon$

gdzie:

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}, x = \begin{bmatrix} x_1^T \\ x_2^T \\ . \\ . \\ . \\ x_n^T \end{bmatrix} = \begin{bmatrix} 1\ x_{11} \ . \ . \ . \ x_{1p} \\ 1\ x_{21} \ . \ . \ . \ x_{2p} \\ . \ . \ . \ . \ . \ . \\ . \ . \ . \ . \ . \ . \\ . \ . \ . \ . \ . \ . \\ 1\ x_{n1} \ . \ . \ . \ x_{np} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ . \\ . \\ \beta_p \end{bmatrix}, \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ . \\ . \\ \epsilon_p \end{bmatrix}$$

,

Najczęściej wykorzystuje się do tego celu klasyczną metodę najmniejszych kwadratów i jej pochodne. Metoda ta jest najstarsza i najłatwiejsza do zastosowania, choć posiada wady (np. niewielką odporność na elementy odstające), które udało się usunąć w innych, mniej rozpowszechnionych metodach. Są to odporne metody statystyczne, do których należy regresja medianowa i algorytmy z regularyzacją.

3.Kwartet Anscombe'a :

Niedostateczność prostych algorytmów w ogólnym przypadku pokazuje m.in. kwartet Anscombe'a – specjalnie przygotowany zestaw czterech zbiorów danych, które mają niemal tożsame wskaźniki statystyczne (średnią i wariancję w kierunku X i Y, współczynnik korelacji oraz prostą regresji) mimo znacząco różnego charakteru danych.

4. Metoda najmniejszych kwadratów

Metoda ta ma na celu wyznaczenie linii regresji, linii trendu dla zebranych danych. Stosowana jest ona zarówno do oszacowania zależności liniowej jak również nieliniowej. Nazwa „najmniejsze kwadraty" oznacza, że końcowe rozwiązanie tą metodą minimalizuje sumę kwadratów błędów przy rozwiązywaniu każdego z równań.

Przypuśćmy, że dane są punkty $(x_1, y_1), (x_2, y_2), (x_n, y_n)$. Chcemy znaleźć prostą $y = ax + b$, której wykres najlepiej w sensie najmniejszych kwadratów przybliża dane punkty. Oznacza to, że chcemy znaleźć minimum, a właściwie wartość najmniejszą funkcji

$$f(a, b) = \sum_{i=1}^{n} (ax_i + b - y_i)^2.$$

5. Algorytm regresji liniowej

Opis algorytmu:

Jest to jeden z najpopularniejszych i najprostszych algorytmów. Zakłada on istnienie zależności liniowej pomiędzy zmienną modelowaną a predyktorami. W najprostszym przypadku regresji liniowej przedstawia ona jedną zmienną modelowaną i jeden predyktor. Zależność pomiędzy nimi jest modelowana na dwuwymiarowym układzie współrzędnych za pomocą prostej o odpowiednim nachyleniu. Odzwierciedla ona trend i zmienność

danych. Rozwiązaniem problemu prognozowania z użyciem regresji liniowej będą odpowiednio dobrane parametry . To one definiują jak bardzo prosta jest dopasowana do danych. Celem jest tu minimalizowanie sumy pionowych odległości wszystkich obserwacji od prostej. Wykorzystuje się do tego m.in. metodę estymacji najmniejszych kwadratów. Regresja liniowa jest prosta w zrozumieniu i użyciu. Zapewnia szybkie uczenie, a czas wykonywania algorytmu jest krótki. Ma jednak jedną dużą wadę o której należy pamiętać: jest bardzo wrażliwa na wartości odstające. Jedna odstająca wartość może nam znacznie zaburzyć proces optymalizacji współczynników modelu. Dane wejściowe powinny być zatem dobrze wyczyszczone zarówno z wartości odstających, brakujących oraz duplikatów.

6. Przebieg projektu

Przyjmiemy zbiór danych "Housing Dataset", który zawiera informacje o różnych domach w Bostonie. Te dane były częścią repozytorium uczenia maszynowego UCI. Możemy również uzyskać dostęp do tych danych z biblioteki scikit-learn. W tym zbiorze danych znajduje się 506 próbek i 13 cech. Celem jest przewidzenie wartości cen domu przy użyciu podanych cech.

Najpierw zaimportujemy wymagane biblioteki.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline
```

Następnie załadujemy dane z biblioteki scikit-learn.

In [2]:
```
from sklearn.datasets import load_boston
housing_dataset = load_boston()
```

Aby lepiej zrozumieć dane które znajdują się w naszym zbiorze danych wyświetlamy klucze.

In [3]:
```
print(housing_dataset.keys())
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

```
data: zawiera informacje o domach
target: ceny domów
feature_names: nazwy cech
DESCR: opis zbioru danych
```

Aby dowiedzieć się więcej o zbiorze danych którym będziemy się posługiwać użyjemy ['DESCR']

In [4]:
```
print(housing_dataset['DESCR'])
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**
```

```
    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14)
is usually the target.

    :Attribute Information (in order):
        - 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      nitric 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 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
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black people by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon Uni
versity.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address reg
ression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sour
ces of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings
on the Tenth International Conference of Machine Learning, 236-243, University of Massachu
setts, Amherst. Morgan Kaufmann.
```

Ceny domu wskazane są przez atrybut MEDV (mediana cen domów) są naszą zmienną docelową, natomiast pozostałe atrybuty posłużą nam do przewidywania cen domów.

W następnym kroku przy użyciu funkcji pd.dataframe oraz head() wyświetlimy dane w postaci tabelki co pozwoli nam na lepszą wizualizację danych i zapoznanie się z nimi.

In [5]:
```
housing = pd.DataFrame(housing_dataset.data, columns=housing_dataset.feature_names)
housing.head()
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

Jak można zauważyć nasz docelowy atrybut MEDV nie znajduję się w tabelce. Dlatego w następnym kroku dodajemy nową kolumnę MEDV.

In [6]:
```
housing['MEDV'] = housing_dataset.target
```

Wstępne przetwarzanie danych

Po załadowaniu danych możemy sprawdzić czy w naszej bazie nie brakuje wartości przy pomocy funkcji isnull() która zwróci nam ilosc brakujących danych dla każdej cechy.

In [7]:
```
housing.isnull().sum()
```

Out[7]:
```
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
MEDV       0
dtype: int64
```

Jak widać baza jest kompletna.

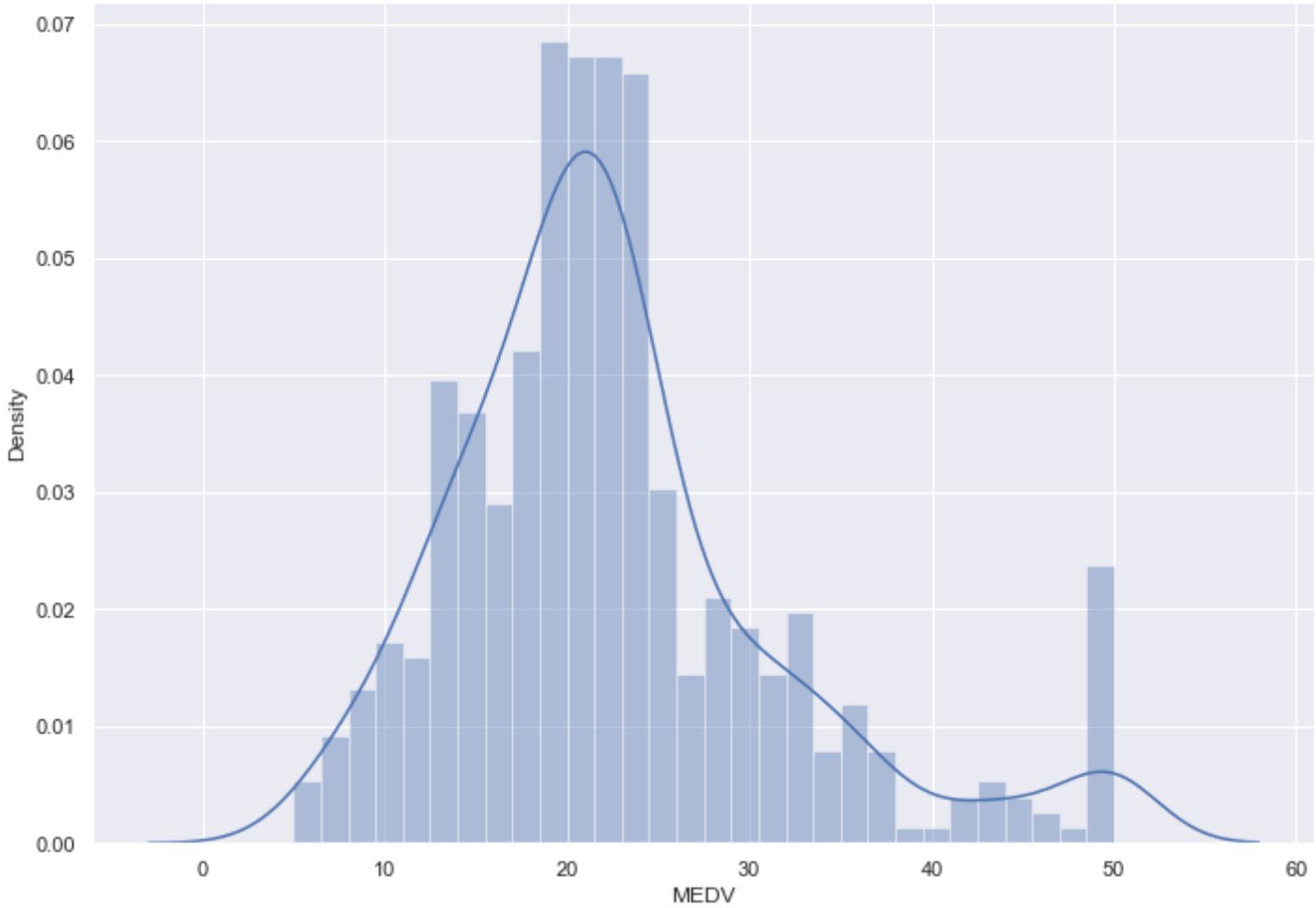
Exploracyjna analiza danych

Eksploracyjna analiza danych jest ważnym krokiem przed rozpoczęciem treningu modelu. Dlatego użyjemy wizualizacji danych, aby lepiej zrozumieć związek zmiennej docelowej z innymi cechami.

Wykreślimy rozkład zmiennej docelowej MEDV. Wykorzystamy do tego funkcję distplot z biblioteki seaborn.

In [8]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(housing['MEDV'], bins=30)
plt.show()
```

```
C:\Users\Mikolaj\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Jak widać wartości MEDV mają rozkład normalny z niewielką liczbą wartości odstających od reszty.

Następnie tworzymy macierz korelacji, która mierzy liniowe zależności między zmiennymi. Macierz korelacji można utworzyć za pomocą funkcji corr z biblioteki pandas dataframe. Do wykreślenia macierzy korelacji użyjemy funkcji heatmap z biblioteki seaborn

In [9]:
```
correlation_matrix = housing.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

Out[9]:
```
<AxesSubplot:>
```

Obserwacje:

- Aby dopasować model regresji liniowej, wybieramy te cechy, które mają wysoką korelację z naszą zmienną MEDV. obserwując macierz korelacji, możemy zauważyć, że RM ma silną dodatnią korelację z MEDV (0,7), natomiast LSTAT ma wysoką ujemną korelację z MEDV (-0,74).
- Przy wyborze cech do modelu regresji liniowej należy sprawdzić, czy nie występuje współliniowość. Cechy RAD, TAX mają korelację na poziomie 0,91, i są silnie skorelowane. Dlatego nie powinniśmy wybierać obu tych cech razem do trenowania modelu. Sprawdź to, aby uzyskać wyjaśnienie. To samo dotyczy cech DIS i AGE, których korelacja wynosi -0,75.

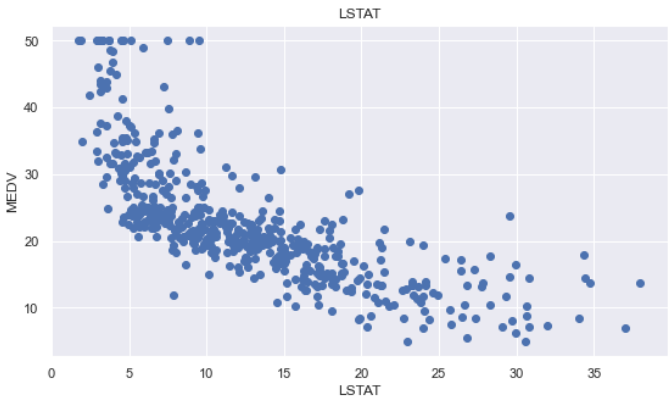
Po analizie macierzy wybieramy RM i LSTAT jako nasze cechy. Użyjemy scatter plot aby zbadać ich zachowanie względem MEDV.

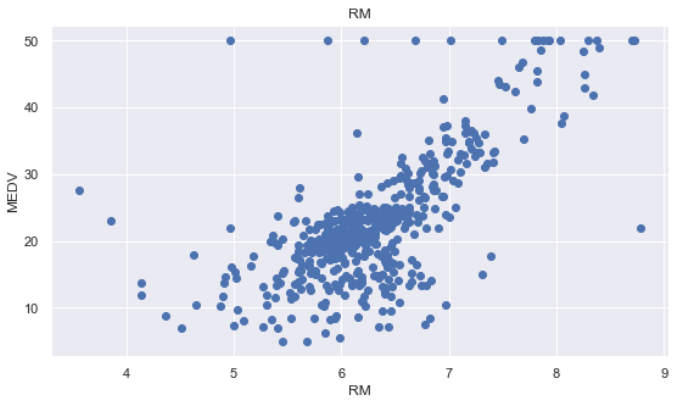
In [10]:

```
plt.figure(figsize=(20, 5))

features = ['LSTAT', 'RM']
target = housing['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = housing[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```





Obserwacje:

- Wraz ze wzrostem liniowym wartości RM ceny rosną. Istnieje niewielka ilość wartości odstających. a dane ograniczone są liczbą 50.
- Ze wzrostem LSTAT ceny wydają się maleć jednak nie przypomina to przebiegu liniowego.

Przygotowanie danych do trenowania modelu.

Kolumny LSTAT i RM łączymy za pomocą np.c_ udostępnianego przez bibliotekę numpy.

In [11]:
```
X = pd.DataFrame(np.c_[housing['LSTAT'], housing['RM']], columns = ['LSTAT','RM'])
Y = housing['MEDV']
```

Podział danych na zbiory treningowe i testowe

Następnie dzielimy dane na zbiory treningowe i testowe. Trenujemy model na 80% próbek, a testujemy pozostałe 20%. Robimy to, aby ocenić wydajność modelu na niewidzialnych danych. Do podziału danych używamy funkcji train_test_split udostępnianej przez bibliotekę scikit-learn. Na koniec wypisujemy rozmiary zbioru treningowego i testowego, aby sprawdzić, czy podział został przeprowadzony poprawnie.

In [12]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

Trenowanie i testowanie modelu

Używamy funkcji LinearRegression do trenowania naszego modelu na zbiorach treningowych i testowych. Imporujemy także biblioteki potrzebne do ewaluacji modelu.

In [13]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

Out[13]:
```
LinearRegression()
```

Ocena modelu używając RMSE i R2-score.

Ocena modelu dla zbioru treningowego

In [14]:
```
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)

print("The model performance for training set")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for training set
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701
```

Ocena modelu dla zbioru testowego

In [15]:
```
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for testing set
RMSE is 5.13740078470291
R2 score is 0.6628996975186954
```

Ocena modelu za pomocą walidacji krzyżowej (cross-validation)

Walidacja k-krotna (ang. k-fold cross-validation)

Polega na podzieleniu zestawu danych na k podzestawów i następnie kolejno braniu jednego z nich jako zestaw danych testowych, używając reszty jako zestaw danych treningowych. Oceną dokładnośći modelu jest średnia arytmetyczna wszystkich k testów.

In [16]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(LinearRegression(), housing_dataset['data'], housing_dataset['tar
print(scores)
```

```
[ 0.60217169  0.60398145  0.35873597 -1.10867706]
```

Wykres rozrzutu przedstawiający rozbierzności między wartościami prawdziwymi a przewidzianymi.

In [17]:
```
plt.scatter(Y_test, y_test_predict)
plt.xlabel("Price: in $1000's")
```

```
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```

Jak można zauważyć wyniki naszej predekcji pomimo starania się aby wybrać optymalne wartości do uczenia mija się z rzeczywistymi wartościami, jednak jest on bliski.

Bibliografia

Wykłady

wikipedia.org

naukowiec.org

In [ ]: