Oppimistehtävä 2

Tekijä: Niko Heikkilä

Käytän tässä oppimistehtävässä samaa datasettiä kuin ensimmäisessä. Valitsin käsiteltäväksi datasetiksi netistä löytyvän Bostonin talojen hinnat csv tiedostossa. ([lähde](https://www.kaggle.com/datasets/vikrishnan/boston-house-prices/)) Käyttötarkoituksena on koneoppimisen avulla ennustaa MEDV (talojen mediaani) arvot. Käytin tähän työhön lineaarista regressiota käyttäen Sickit-Learn kirjastoa.

* Datan tarkoitukset:  
  CRIM: Per capita rikollisuusaste kaupungissa
* ZN: Tämä ominaisuus kuvastaa asuintonttien osuutta, jotka on kaavoitettu yli 25 000 neliöjalan kokoisille alueille. Se ilmaisee määrän maata kussakin kaupungissa, joka on varattu suurille asuintonteille.
* INDUS: INDUS edustaa ei-vähittäiskaupan liiketoiminta-alueiden osuutta kaupungissa. Se kuvaa maan määrää, joka on omistettu ei-vähittäiskaupan liiketoiminnalle, kuten valmistukselle tai teollisille toimille, kussakin kaupungissa.
* CHAS: CHAS on dummy-muuttuja, joka kuvastaa, rajoittaako kaupunki Charles-joen. Se saa arvon 1, jos kaupunki rajoittuu jokeen, ja 0 muuten.
* NOX: NOX edustaa typpioksidien pitoisuutta (ilman saastumista) osina kymmenestä miljoonasta. Se mittaa ilman saastumisen tasoa kussakin kaupungissa.
* RM: RM tarkoittaa keskimääräistä huoneiden määrää asuntoa kohti. Se ilmaisee keskimääräisen huoneiden määrän taloissa tai asunnoissa kussakin kaupungissa.
* AGE: AGE edustaa omistusasuntojen osuutta, jotka on rakennettu ennen vuotta 1940. Se ilmaisee asuinrakennusten iän kussakin kaupungissa, korkeammat arvot osoittavat suuremman osuuden vanhoista kodeista.
* DIS: DIS tarkoittaa painotettuja etäisyyksiä viiteen Bostonin työkeskukseen. Se kuvaa keskimääräistä etäisyyttä kustakin kaupungista viiteen suureen työkeskukseen Bostonissa, pienemmät arvot osoittavat lähempää sijaintia.
* RAD: RAD on indeksi pääsystä säteittäisille moottoriteille. Se mittaa kunkin kaupungin pääsyä säteittäisille moottoriteille, korkeammat arvot osoittavat parempaa pääsyä.
* TAX: TAX edustaa kiinteistöveroprosenttia täydestä arvosta 10 000 dollaria kohti. Se ilmaisee kiinteistöveroprosentin kunkin kaupungin osalta, korkeammat arvot osoittavat korkeampia verokantoja.
* PTRATIO: PTRATIO edustaa oppilas-opettaja -suhdetta kaupungin kouluissa. Se kuvastaa keskimääräistä oppilaiden määrää opettajaa kohti kaupungin julkisissa kouluissa.
* B: Bk on osuus mustista väestöstä kaupungissa.
* LSTAT: Prosenttiosuus väestön alempiarvoisista kaupungeissa.
* MEDV: MEDV edustaa omistusasuntojen keskiarvoista arvoa 1000 dollareissa. Se ilmaisee omistusasuntojen keskihintaa kussakin kaupungissa, mikä on yleisesti käytetty mittari asuntomarkkinoiden arvosta.

Scatter plot MEDV arvojen ennustuksista:

Kuva, joka sisältää kohteen teksti, kuvakaappaus, diagrammi, viiva

Kuvaus luotu automaattisesti

Tässä näemme lineaarisen regression tekemät ennustukset suhteessa oikeisiin arvoihin.

Oma arviointi:

Tämä työ onnistui helpommin kuin ensimmäinen nyt kun on hieman enemmän kokemusta ja tietoa aiheesta. Saman datasetin käyttö helpotti työtä myös osittain. Kehityskohtana olisi kyky tehdä monimutkaisempia töitä kuin tämä.

Tähtään 1 arvosanaan.

Ensimmäiset 50 otosta datasta:

CRIM ZN INDUS 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 4.09 1 296.0 15.3 396.9 4.98 24.0

1 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242.0 17.8 396.9 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 18.7 396.9 5.33 36.2

5 0.02985 0.0 2.18 0 0.458 6.43 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 15.2 395.6 12.43 22.9

7 0.14455 12.5 7.87 0 0.524 6.172 96.1 5.9505 5 311.0 15.2 396.9 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 17.1 18.9

10 0.22489 12.5 7.87 0 0.524 6.377 94.3 6.3467 5 311.0 15.2 392.52 20.45 15.0

11 0.11747 12.5 7.87 0 0.524 6.009 82.9 6.2267 5 311.0 15.2 396.9 13.27 18.9

12 0.09378 12.5 7.87 0 0.524 5.889 39.0 5.4509 5 311.0 15.2 390.5 15.71 21.7

13 0.62976 0.0 8.14 0 0.538 5.949 61.8 4.7075 4 307.0 21.0 396.9 8.26 20.4

14 0.63796 0.0 8.14 0 0.538 6.096 84.5 4.4619 4 307.0 21.0 380.02 10.26 18.2

15 0.62739 0.0 8.14 0 0.538 5.834 56.5 4.4986 4 307.0 21.0 395.62 8.47 19.9

16 1.05393 0.0 8.14 0 0.538 5.935 29.3 4.4986 4 307.0 21.0 386.85 6.58 23.1

17 0.7842 0.0 8.14 0 0.538 5.99 81.7 4.2579 4 307.0 21.0 386.75 14.67 17.5

18 0.80271 0.0 8.14 0 0.538 5.456 36.6 3.7965 4 307.0 21.0 288.99 11.69 20.2

19 0.7258 0.0 8.14 0 0.538 5.727 69.5 3.7965 4 307.0 21.0 390.95 11.28 18.2

20 1.25179 0.0 8.14 0 0.538 5.57 98.1 3.7979 4 307.0 21.0 376.57 21.02 13.6

21 0.85204 0.0 8.14 0 0.538 5.965 89.2 4.0123 4 307.0 21.0 392.53 13.83 19.6

22 1.23247 0.0 8.14 0 0.538 6.142 91.7 3.9769 4 307.0 21.0 396.9 18.72 15.2

23 0.98843 0.0 8.14 0 0.538 5.813 100.0 4.0952 4 307.0 21.0 394.54 19.88 14.5

24 0.75026 0.0 8.14 0 0.538 5.924 94.1 4.3996 4 307.0 21.0 394.33 16.3 15.6

25 0.84054 0.0 8.14 0 0.538 5.599 85.7 4.4546 4 307.0 21.0 303.42 16.51 13.9

26 0.67191 0.0 8.14 0 0.538 5.813 90.3 4.682 4 307.0 21.0 376.88 14.81 16.6

27 0.95577 0.0 8.14 0 0.538 6.047 88.8 4.4534 4 307.0 21.0 306.38 17.28 14.8

28 0.77299 0.0 8.14 0 0.538 6.495 94.4 4.4547 4 307.0 21.0 387.94 12.8 18.4

29 1.00245 0.0 8.14 0 0.538 6.674 87.3 4.239 4 307.0 21.0 380.23 11.98 21.0

30 1.13081 0.0 8.14 0 0.538 5.713 94.1 4.233 4 307.0 21.0 360.17 22.6 12.7

31 1.35472 0.0 8.14 0 0.538 6.072 100.0 4.175 4 307.0 21.0 376.73 13.04 14.5

32 1.38799 0.0 8.14 0 0.538 5.95 82.0 3.99 4 307.0 21.0 232.6 27.71 13.2

33 1.15172 0.0 8.14 0 0.538 5.701 95.0 3.7872 4 307.0 21.0 358.77 18.35 13.1

34 1.61282 0.0 8.14 0 0.538 6.096 96.9 3.7598 4 307.0 21.0 248.31 20.34 13.5

35 0.06417 0.0 5.96 0 0.499 5.933 68.2 3.3603 5 279.0 19.2 396.9 9.68 18.9

36 0.09744 0.0 5.96 0 0.499 5.841 61.4 3.3779 5 279.0 19.2 377.56 11.41 20.0

37 0.08014 0.0 5.96 0 0.499 5.85 41.5 3.9342 5 279.0 19.2 396.9 8.77 21.0

38 0.17505 0.0 5.96 0 0.499 5.966 30.2 3.8473 5 279.0 19.2 393.43 10.13 24.7

39 0.02763 75.0 2.95 0 0.428 6.595 21.8 5.4011 3 252.0 18.3 395.63 4.32 30.8

40 0.03359 75.0 2.95 0 0.428 7.024 15.8 5.4011 3 252.0 18.3 395.62 1.98 34.9

41 0.12744 0.0 6.91 0 0.448 6.77 2.9 5.7209 3 233.0 17.9 385.41 4.84 26.6

42 0.1415 0.0 6.91 0 0.448 6.169 6.6 5.7209 3 233.0 17.9 383.37 5.81 25.3

43 0.15936 0.0 6.91 0 0.448 6.211 6.5 5.7209 3 233.0 17.9 394.46 7.44 24.7

44 0.12269 0.0 6.91 0 0.448 6.069 40.0 5.7209 3 233.0 17.9 389.39 9.55 21.2

45 0.17142 0.0 6.91 0 0.448 5.682 33.8 5.1004 3 233.0 17.9 396.9 10.21 19.3

46 0.18836 0.0 6.91 0 0.448 5.786 33.3 5.1004 3 233.0 17.9 396.9 14.15 20.0

47 0.22927 0.0 6.91 0 0.448 6.03 85.5 5.6894 3 233.0 17.9 392.74 18.8 16.6

48 0.25387 0.0 6.91 0 0.448 5.399 95.3 5.87 3 233.0 17.9 396.9 30.81 14.4

49 0.21977 0.0 6.91 0 0.448 5.602 62.0 6.0877 3 233.0 17.9 396.9 16.2 19.4

50 0.08873 21.0 5.64 0 0.439 5.963 45.7 6.8147 4 243.0 16.8 395.56 13.45 19.7

Koodi:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Load data

column\_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

data = pd.read\_csv('housing.csv', header=None, delimiter=r'\s+', names=column\_names)

missing\_values = data.isnull().sum()

data = data.dropna()

# Select features and target

X = data.drop('MEDV', axis=1)

y = data['MEDV']

# Split data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and fit a linear regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

# Visualize predictions vs. actual values

plt.scatter(y\_test, y\_pred)

plt.xlabel('Actual Values')

plt.ylabel('Predictions')

plt.title('Linear Regression Predictions vs. Actual Values')

plt.show()