Множествена линейна регресия и полиномна регресия с Python

В темата ще бъдат разгледани следните основни въпроси:

- Множествена линейна регресия с Python
  - Импортиране на пакети и класове и предоставяне на данни
  - Създаване на модел и напасване
  - Получаване на резултати
  - Прогнозиране на отговора
- Полиномна регресия с Python
  - Импортиране на пакети и класове
  - Предоставяне на данни и трансформиране на данни
  - Създаване на модел и напасване
  - Получаване на резултати
  - Прогнозиране на отговора

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Множествена линейна регресия и полиномна регресия с Python

1. Множествена линейна регресия с Python

Множествената линейна регресия може да се приложи като се следват същите стъпки, реализиращи проста регресия. Основната разлика е в това, че масивът x има две или повече колони.

Стъпки 1 и 2: Импортиране на пакети и класове и предоставяне на данни

Първо се импортират numpy и sklearn.linear_model.LinearRegression и се предоставят известните входове и изходи (фиг. 7.1.а).

Непосредственото въвеждане на масивите е лесен начин за дефиниране на входа x и изхода y. Накрая могат да се отпечатат x и y, за да се видят как изглеждат сега (фиг. 7.1.б)

При множествената линейна регресия x е двумерен масив с поне две колони, докато y обикновено е едномерен масив. Примерът от фиг. 7.1 е прост пример за множествена линейна регресия и входът x има точно две колони.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

```
import numpy as np
from sklearn.linear_model import LinearRegression

x = [ [0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15],
[55, 34], [60, 35] ]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)

print(f"Input x: {x}")
print(f"Output y: {y}")
```

а)

```
Input x: ([[ 0,  1],
       [ 5,  1],
       [15,  2],
       [25,  5],
       [35, 11],
       [45, 15],
       [55, 34],
       [60, 35]])
Output y: ([ 4,  5, 20, 14, 32, 22, 38, 43]
```

б)

Фиг. 7.1. Импортиране на пакета numpy и класа LinearRegression и предоставяне на данни

а. Код на Python **б.** Изход от програмата

Стъпка 3: Създаване на модел и напасване

Следващата стъпка е създаване на регресионния модел като екземпляр на LinearRegression и неговото напасне с .fit() (фиг. 7.2)

```
model = LinearRegression().fit(x, y)
```

Фиг. 7.2. Създаване на регресионния модел и неговото напасване

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Резултатът от този код е моделът на променливата model, отнасящ се до обект от тип LinearRegression. Той представлява регресионния модел, който най-добре съответства на подадените данни.

Стъпка 4: Получаване на резултати

Основните резултати на регресионния модел могат да се получат по същия начин, както в случая на проста линейна регресия (фиг. 7.3)

```
r_sq = model.score(x, y)
print(f"Coefficient of determination: {r_sq}")
print(f"Intercept: {model.intercept_}")
print(f"Coefficients: {model.coef_}")
```

а)

```
Coefficient of determination: 0.8615939258756776
Intercept: 5.52257927519819
Coefficients: [0.44706965 0.25502548]
```

б)

Фиг. 7.3. Получаване на резултатите на регресионния модел

а. Код на Python **б.** Изход от програмата

Стойността на R^2 се получава с помощта на .score() и стойностите на оценителите на коефициентите на регресия с .intercept_ и .coef_. Отново .intercept_ съдържа отклонението b_0, докато в този случай .coef_ е масив, съдържащ b_1 и b_2.

В този пример пресечната точка е приблизително 5,52 и това е стойността на прогнозирания отговор, когато $x_1 = x_2 = 0$. Увеличаването на

x_1 с 1 води до увеличение на прогнозирания отговор с 0,45. По същия начин, когато x_2 нараства с 1, отговорът нараства с 0,26.

Стъпка 5: Прогнозиране на отговора

Прогнозите работят по същия начин, както в случая на проста линейна регресия (фиг. 7.4).

```
y_pred = model.predict(x)
print(f"Predicted response:\n{y_pred}")
```

а)

```
Predicted response:
[ 5.77760476 8.012953 12.73867497 17.9744479 23.97529728
29.4660957 38.78227633 41.27265006]
```

б)

Фиг. 7.4. Прогнозиране на отговора на регресионния модел

а. Код на Python **б.** Изход от програмата

Предсказаният отговор освен с .predict(), може да се получи чрез непосредствено прилагана на формула (???) на регресионния модел (фиг. 7.5).

Предвидите изходните стойности се получават, като се умножи всяка колона на входа с подходящото тегло, резултатите се сумират и накрая, към сумата се добави пресечната точка.

```
y_pred = model.intercept_ + np.sum(model.coef_ * x, axis=1)
print(f"Predicted response:\n{y_pred}")
```

а)

```
Predicted response:
[ 5.77760476 8.012953 12.73867497 17.9744479 23.97529728
29.4660957 38.78227633 41.27265006]
```

б)

Фиг. 7.5. Прогнозиране на отговора чрез изчисление

а. Код на Python **б.** Изход от програмата

Вече създаденият регресионен модел може да се приложи към нови данни (фиг. 7.6).

```
x_new = np.arange(10).reshape((-1, 2))
print(f"x_new:\n{x_new}")
y_new = model.predict(x_new)
print(f"y_new:\n{y_new}")
```

а)

```
x_new:
([[0, 1],
  [2, 3],
  [4, 5],
  [6, 7],
  [8, 9]])
y_new:
[ 5.77760476, 7.18179502, 8.58598528, 9.99017554, 11.3943658 ]
```

б)

Фиг. 7.6. Прилагане на регресионния модел към нови данни

а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника", финансиран от Оперативна програма „Наука и образование за интелигентен растеж", съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

2. Полиномна регресия с Python

Прилагането на полиномна регресия с scikit-learn е много подобно на реализацията на линейната регресия. Има само една допълнителна стъпка: трябва да се трансформира масива от входове, за да се включат нелинейни елементи като x^2.

Стъпка 1: Импортиране на пакети и класове

В допълнение към numpy и sklearn.linear_model.LinearRegression, трябва също да се импортира и класа PolynomialFeatures от sklearn.preprocessing (фиг. 7.7).

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

Фиг. 7.7. Импортиране на пакети и класове за полиномна регресия

Стъпка 2а: Предоставяне на данни

Тази стъпка дефинира входа и изхода и е същата като в случая на линейна регресия (фиг. 7.8).

```
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([15, 11, 2, 8, 25, 32])
```

Фиг. 7.8. Предоставяне на данните за полиномна регресия

Кодът от фиг. 7.8 предоставя входа и изхода в подходящ формат. Входът трябва да бъде двумерен масив и затова се използва .reshape().

Стъпка 2b: Трансформиране на входните данни

Това е новата стъпка, която трябва да се приложи за полиномна регресия. В тази стъпка трябва да се включи x^2 и може би други елементи като допълнителни функции, когато се прилага полиномна регресия. Поради тази причина входният масив x трябва да се трансформира, за да съдържа всякакви допълнителни колони със стойностите на x^2 и евентуално други функции.

Входният масив може да се трансформира по няколко начина. Първият начин използва метода insert() от numpy. Друг много удобен за тази цел е класът PolynomialFeatures. Създаването на екземпляр на този клас е показано на фиг. 7.9.

```
transformer = PolynomialFeatures(degree=2, include_bias=False)
```

Фиг. 7.9. Създаване на екземпляр на класа PolynomialFeatures

Създаденият променлив преобразовател transformer е екземпляр на класа PolynomialFeatures, който може да се използва, за да се трансформира входа x.

На конструктора PolynomialFeatures могат да се предоставят няколко незадължителни параметъра:

- степента **degree** е цяло число (2 по подразбиране), което представлява степента на полиномната регресионна функция.
- **interaction_only** е булева променлива Boolean (False по подразбиране), която решава дали да включва само функции за

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

взаимодействие, т.е. само произведения на различни входове до степен **degree** (True) или всички функции (False).

- **include_bias** е булева променлива Boolean (True по подразбиране), която решава дали да включи отклонението, характеристика, при която всички степени на полинома са 0, т.е. това е колона от 1-ци (True) или не (False).

Примерът от фиг. 7.9 използва стойностите по подразбиране на всички параметри с изключение на include_bias. Ако е необходимо да се експериментира със степента на функцията, може да бъде от полза да се предаде и този аргумент на конструктора.

Преди да се приложи преобразуването, transformer трябва да се напасне с .fit() (фиг. 7.10).

```
transformer.fit(x)
```

Фиг. 7.10. Напасване на преобразувателя transformer

След това преобразователят transformer е готов за създаване на нов, модифициран входен масив. За тази цел се прилага метода .transform() (фиг. 7.11).

```
x_ = transformer.transform(x)
```

Фиг. 7.11. Създаване на нов модифициран масив x_

Това е трансформацията на входния масив с .transform(). Той приема входния масив като аргумент и връща модифицирания масив.

Може също да се използва метода .fit_transform(), за да се заменят трите предишни израза от фиг. 7.9, фиг. 7.10 и фиг. 7.11 само с един (фиг. 7.12).

```
x_= PolynomialFeatures(degree=2, include_bias=False).fit_transform(x)
```

Фиг. 7.12. Създаване на нов модифициран масив x_ с един оператор

С .fit_transform() се напасва и трансформира входния масив чрез един оператор. Този метод ефективно преобразува входния масив, като извършва същите дейности като .fit() и .transform(), извикани в този ред. Той също така връща модифицирания масив. Модифицираният входен масив може да се изведе с кода на фиг. 7.13.

```
print(f"x_:\n{x_}")
```

а)

```
x_:
[[   5.,   25.],
 [  15.,  225.],
 [  25.,  625.],
 [  35., 1225.],
 [  45., 2025.],
 [  55., 3025.]])
```

б)

Фиг. 7.13. Отпечатване на модифицирания входен масив

а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Регресионният модел се създава и напасва отново с метода .fit(), на който вход е модифицираният входен масив x_, а не оригиналният x (фиг. 7.14)

```
model = LinearRegression().fit(x_, y)
```

Фиг. 7.14. Създаване на регресионен модел с модифицирания входен масив x_

Стъпка 4: Получаване на резултати

Характеристиките на модела могат да се получат по същия начин, както в случая на линейна регресия (фиг. 7.15).

```
r_sq = model.score(x_, y)
print(f"Coefficient of determination: {r_sq}")
print(f"Intercept: {model.intercept_}")
print(f"Coefficients: {model.coef_}")
```

а)

```
Coefficient of determination: 0.8908516262498563
Intercept: 21.372321428571436
Coefficients: [-1.32357143  0.02839286]
```

б)

Фиг. 7.13. Отпечатване характеристиките на модела

а. Код на Python **б.** Изход от програмата

Много подобни резултати могат да се получат с различни аргументи за трансформация и регресия (фиг. 7.16).

```
x_= PolynomialFeatures(degree=2, include_bias= True).fit_transform(x)
```

Фиг. 7.16. Създаване на нов модифициран масив x_ с един оператор

За разлика от фиг. 7.12, ако PolynomialFeatures се извика с параметъра по подразбиране include_bias=True или ако просто се пропусне този параметър, тогава новият входен масив x_ ще съдържа една допълнителната най-лява колона, съдържаща само 1 стойност. Тази колона съответства на параметъра intercept. В този случай модифицираният входен масив ще изглежда във вида от фиг. 7.17.

```
print(f"x_:\n{x_}")
```

а)

```
x_:
[[1.000e+00, 5.000e+00, 2.500e+01],
 [1.000e+00, 1.500e+01, 2.250e+02],
 [1.000e+00, 2.500e+01, 6.250e+02],
 [1.000e+00, 3.500e+01, 1.225e+03],
 [1.000e+00, 4.500e+01, 2.025e+03],
 [1.000e+00, 5.500e+01, 3.025e+03]]
```

б)

Фиг. 7.13. Отпечатване на модифицирания входен масив при include_bias=True

а. Код на Python **б.** Изход от програмата

Първата колона на x_ съдържа единици, втората има стойностите на x, докато третата съдържа квадратите на x.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Параметърът intercept вече е включен в най-лявата колона от единици и не е необходимо да се включва отново, когато се създава екземпляра на LinearRegression. По този начин може да се предостави fit_intercept=False (фиг. 7.18).

```
model = LinearRegression(fit_intercept=False).fit(x_, y)
```

Фиг. 7.18. Създаване на регресионен модел с модифицирания входен масив x_ и fit_intercept=False

Създаденият модел на линейна регресия съответства на новия входен масив x_. Следователно x_ трябва да бъде подаден като първи аргумент вместо x. Приложеният подход от фиг. 7.17 и фиг. 7.18 дава следните резултати, които са подобни на предишния случай (фиг. 7.19).

```
r_sq = model.score(x_, y)
print(f"Coefficient of determination: {r_sq}")
print(f"Intercept: {model.intercept_}")
print(f"Coefficients: {model.coef_}")
```

а)

```
Coefficient of determination: 0.8908516262498564
Intercept: 0.0
Coefficients: [21.37232143 -1.32357143  0.02839286]
```

б)

Фиг. 7.19. Отпечатване характеристиките на модела

а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Вижда се, че сега .intercept_ е нула, но .coef_ всъщност съдържа b_0 като първи елемент. Всичко друго е същото.

Стъпка 5: Прогнозиране на отговора

За да се получи предсказания отговор, се използва отново метода .predict(), но аргументът трябва да бъде модифицираният вход x_ вместо стария x (фиг. 7.20).

```
y_pred = model.predict(x_)
print(f"Predicted response:\n{y_pred}")
```

а)

```
Predicted response:
[15.46428571  7.90714286  6.02857143  9.82857143 19.30714286
34.46428571]
```

б)

Фиг. 7.20. Предсказване и отпечатване на отговора

а. Код на Python **б.** Изход от програмата

Както може да се види от фиг. 7.20, методът .predict() работи почти по същия начин, както в случая на линейна регресия. Той просто изисква модифицирания вход x_ вместо оригинала x.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника", финансиран от Оперативна програма „Наука и образование за интелигентен растеж", съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.