Проста линейна регресия с Python

В темата ще бъдат разгледани следните основни въпроси:

- Пакети на Python за линейна регресия
- Проста линейна регресия с scikit-learn
  - Импортиране на пакети и класове
  - Предоставяне на данни
  - Създаване на модел и напасване с данните
  - Получаване на оценъчни 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.

Проста линейна регресия с Python

Линейната регресия е важна част от машинното обучение и науката за обработка данни, която се използва в разпознаването на изображения, програмните решения във финансовия и енергийния сектор, разработването на автономни превозни средства, медицината, социалните мрежи и др.

Линейната регресия е една от основните техники за машинно обучение, статистически и научни изследвания.

Теорията на линейната регресия бе разгледана в тема 4. Тук ще бъдат дадени примери за нейната реализация в Python.

1. Пакети на Python за линейна регресия

За да се реализира линейна регресия в Python, трябва да се приложат правилните пакети и техните функции и класове.

NumPy е основен научен пакет на Python, който позволява прилагането на високопроизводителни операции върху едномерни и многомерни масиви. Той също така предлага много математически операции и е с отворен код.

Пакетът **scikit-learn** е широко използвана библиотека на Python за машинно обучение, изградена върху NumPy и някои други пакети. Той предоставя средства за предварителна обработка на данни, намаляване на размерността, прилагане на регресия, класифициране, групиране и др. Подобно на NumPy, scikit-learn също е с отворен код.

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.

Повече за линейните модели и по-задълбочена представа за това как работи пакета scikit-learn може да се научи на страницата Linear Models [https://scikit-learn.org/stable/modules/linear_model.html] на уебсайта scikit-learn [https://scikit-learn.org/stable/].

При прилагане на линейна регресия и нужда от функционалност извън обхвата на scikit-learn, е необходимо да се използва statsmodels. Това е мощен пакет на Python за оценка на статистически модели, извършване на тестове и др. Освен това е с отворен код.

Повече информация за statsmodels може да се намери на официалния му уебсайт [https://www.statsmodels.org/stable/index.html].

Ако тези пакети не са инсталирани на вашия компютър може да ги инсталирате (фиг. 6.1).

```
pip3 install numpy scikit-learn statsmodels
```

Фиг. 6.1. Инсталиране на пакетите NumPy, scikit-learn и statsmodels

Това ще инсталира NumPy, scikit-learn, statsmodels и всичко от което те зависят

2. Проста линейна регресия с scikit-learn

За реализиране на проста линейна регресия се прилагат пет основни стъпки:

1. Импортират се пакетите и класовете, които са необходими за проста линейна регресия.

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. Предоставят се данни за обработка и евентуално те се трансформират по подходящ начин.

3. Създава се регресионен модел и се съгласува със съществуващите данни.

4. Проверяват се резултатите от напасването на модела, за да се разбере дали моделът е задоволителен.

5. Моделът се прилага за прогнози.

Тези стъпки са повече или по-малко общи за повечето регресионни подходи и реализации. В останалата част от лекцията ще се научи как да се изпълняват тези стъпки за няколко различни сценария.

Стъпка 1: Импортиране на пакети и класове

Първата стъпка е да се импортират пакета numpy и класа LinearRegression от sklearn.linear_model (фиг. 6.2)

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

Фиг. 6.2. Импортиране на пакета numpy и класа LinearRegression

Това са всички функционалности, от които се нужда прилагането на линейната регресия.

Основният тип данни на NumPy е типът масив, наречен numpy.ndarray. Останалата част от тази лекция използва термина **масив** за обозначаване на екземпляри от типа numpy.ndarray.

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.

Ще се използва класа sklearn.linear_model.LinearRegression, за да се извършва линейна и полиномна регресия и да се правят съответни прогнози.

Стъпка 2: Предоставяне на данни

Втората стъпка е дефиниране на данни, с които да се работи. Входовете (регресори, x) и изходите (отговори, y) трябва да бъдат масиви или подобни обекти. Най-простият начин за предоставяне на данни за регресия е тяхното директно дефиниране (фиг. 6.3)

```
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])
```

Фиг. 6.3. Данни регресори, x и отговори, y за линейна регресия

Кодът от фиг. 6.3 дефинира два масива: входен, x, и изходен, y. Методът .reshape() се извиква за входния масив x, защото този масив трябва да е двуизмерен или по-точно трябва да има една колона и толкова редове, колкото е необходимо. Точно това указва аргументът (-1, 1) на .reshape().

Масивите x и y могат да се разгледат по следния начин (фиг. 6.4).

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.

```
print(f"x:\n {x}")
print(f"y:\n {y}")
```

а)

```
x:
[[ 5],
 [15],
 [25],
 [35],
 [45],
 [55]]
y:
[ 5, 20, 14, 32, 22, 38]
```

б)

Фиг. 6.4. Изглед на входния масив x и изходен y

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

От фиг. 6.4 се вижда, че x има две измерения и x.shape е (6, 1), докато y има едно измерение, а y.shape е (6).

Стъпка 3: Създаване на модел и напасване с данните

Следващата стъпка е да се създаде линеен регресионен модел и да се напасне, като се използват съществуващите данни.

Създаването на екземпляр на класа LinearRegression, който ще представлява регресионния модел е показано на (фиг. 6.5).

```
model = LinearRegression()
```

Фиг. 6.5. Създаване на екземпляр на класа LinearRegression

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.

Операторът от фиг. 6.5 създава променливата model като екземпляр на класа LinearRegression. Допълнително могат да се предоставят няколко незадължителни параметъра на LinearRegression:

- **fit_intercept** е булев тип (Boolean), който, ако е True, позволява да се изчисли пресечната точка b_0 или, ако е False, я счита за равна на нула. По подразбиране стойността на параметъра е True.
- **normalize** е булев тип (Boolean),, който, ако е True, позволява да се нормализират входните данни. По подразбиране е False, в който случай не се нормализират входните данни.
- **copy_X** е Boolean, който решава дали да копира (True) или презапише входните променливи (False). По подразбиране е True.
- **n_jobs** е или цяло число, или None. Той представлява броя на заданията, използвани при паралелно изчисление. По подразбиране е None, което обикновено означава едно задание. -1 означава да се използват всички налични процесори.

Създаденият модел от фиг. 6.5 използва стойностите по подразбиране на всички параметри.

За да се използва модела, първо трябва да се извика метода .fit() на модела (фиг. 6.6).

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.

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

Фиг. 6.6. Изчисляване на оптималните стойности на теглата b_0 и b_1 на модела на линейната регресия

С метода .fit() се изчисляват оптималните стойности на теглата b_0 и b_1, като се използват съществуващите входове x и изходи y, като аргументи. Методът .fit() обработва модела и връща self, което е самият модел на променливата. Ето защо кодът от фиг. 6.5 и 6.6 може да се замени с по-кратък код от фиг. 6.7.

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

Фиг. 6.7. Създаване на модела на линейната регресия и изчисляване на оптималните стойности на теглата b_0 и b_1 на модела

Стъпка 4: Получаване на оценъчни резултати

След като се създаде модела, могат да се получат оценъчни резултати, за да се провери дали моделът работи задоволително и той да се интерпретира.

Може да се изчисли коефициента на определяне R^2, с метода .score(), извикан от модела (фиг. 6.8).

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.

```
r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
```

а)

```
coefficient of determination: 0.7158756137479542
```

б)

Фиг. 6.8. Изчисляване на коефициента на определяне R^2

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

Когато се прилага метода .score(), аргументите също са предикторът x и отговорът y, а върнатата стойност е R^2.

Атрибутите на model са .intercept_, който представлява коефициента b_0, и .coef_, който представлява b_1 (фиг. 6.9). В scikit-learn, по конвенция, долната черта в края показва, че даден атрибут е оценен. В този пример .intercept_ и .coef_ са приблизителни стойности.

```
print(f"intercept: {model.intercept_}")
print(f"slope: {model.coef_}")
```

а)

```
intercept: 5.633333333333329
slope: [0.54]
```

б)

Фиг. 6.9. Извеждане на атрибутите 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.

От изхода на фиг. 6.9 се вижда, че атрибутът .intercept_ (b_0) е скаларно число, докато .coef_ (b_1) е масив. Стойността на b_0 е приблизително 5,63. Това илюстрира, че моделът предвижда отговора 5,63, когато x е нула. Стойността b_1 = 0,54 означава, че прогнозираният отговор нараства с 0,54, когато x се увеличи с единица.

Ако изходът y се предостави на модела като двуизмерен масив, се получава подобен резултат, но в този случай .intercept_ е едномерен масив с единичен елемент b_0, а .coef_ е двуизмерен масив с единичен елемент b_1. Кодът и изхода в този случай е показан на фиг. 6.10.

```
new_model = LinearRegression().fit(x, y.reshape((-1, 1)))
print(f"intercept: {new_model.intercept_}")
print(f"slope: { new_model.coef_}")
```

а)

```
intercept: [5.63333333]
slope: [[0.54]]
```

б)

Фиг. 6.10. Извеждане на атрибутите на модела new_model

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

Стъпка 5: Прогнозиране на отговора

След като се създаде задоволителен модел, той може да се използва за прогнози със съществуващи или нови данни. За да се получи предвидения отговор, се използва метода .predict() (фиг. 6.11).

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.

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

а)

```
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333
35.33333333]
```

б)

Фиг. 6.11. Прогнозиране и извеждане на отговора на модела model

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

Когато се прилага метода .predict(), се предава регресора x като аргумент и се получава съответния прогнозиран отговор y_pred.

Почти идентичен начин за прогнозиране на отговора може да се получи чрез пресмятане на израза от фиг. 6.12.

```
y_pred = model.intercept_ + model.coef_ * x
print(f"predicted response:\n{y_pred}")
```

а)

```
predicted response:
[[ 8.33333333]
 [13.73333333]
 [19.13333333]
 [24.53333333]
 [29.93333333]
 [35.33333333]]
```

б)

Фиг. 6.12. Прогнозиране и извеждане на отговора на модела model чрез изчисляване на израз **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.

В този случай всеки елемент от x се умножава с model.coef_ и се добавя model.intercept_ към произведението.

Резултатът тук се различава от предишния пример само по размера на предсказания отговор. Предсказаният отговор сега е двуизмерен масив, докато в предишния пример (фиг. 6.11) имаше едно измерение.

Ако се намалят броя на измеренията на x до едно, тогава двата подхода от фиг. 6.11 и 6.12 ще дават един и същ резултат. Това може да се направи като се замени x с x.reshape(-1), x.flatten() или x.ravel(), и след това се умножи с model.coef_.

На практика регресионните модели често се прилагат за прогнози. Това означава, че може да се използват създадените модели, за да изчислят изходите въз основа на нови входове (фиг. 6.13).

```
x_new = np.arange(5).reshape((-1, 1))
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]]
y_new:
[5.63333333, 6.17333333, 6.71333333, 7.25333333, 7.79333333]
```

б)

Фиг. 6.13. Прогнозиране и извеждане на отговора на модела model чрез използване на нови входове **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.

Тук методът .predict() се прилага към новия регресор x_new и дава отговор y_new. Този пример използва метода arange() от numpy за генериране на масив с елементи от 0 до 5, но без 5 - тоест 0, 1, 2, 3 и 4.

Повече информация за използването на LinearRegression може да се намери на официалната страница с документация [https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.htm]

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.