Анализ данных

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

ЕГЭ в Португалии

.

Иваново-2024

План

ЕГЭ в Португалии

Имеется набор данных об учащихся, собранный в 2006 году в одной из школ Португалии. Данные представлены в неудобном для машинного обучения виде, в них имеется ошибки. Ваша задача — привести их к надлежащему виду, очистить и обучить на них простую модель.
Данные состоят из четырех файлов:

- data.csv — основная таблица с информацией о учащихся
- attendance.csv — таблица посещений занятий по этому предмету
- school_support.txt - список учащихся, которым оказывается финансовая поддержка
- scores.csv — список финальных оценок по одному из предметов (20-балльная шкала переведенная в проценты)

data2.csv

```
age,Medu,Fedu,traveltime,studytime,famrel,free...
16,4,4,1,2,5,4,4.0,1.0,2.0,5,1,1,0,1,1,1,0
17,4,4,1,1,5,3,4.0,1.0,2.0,5,0,1,0,1,1,1,0
16,1,1,2,1,4,5,5.0,2.0,4.0,5,1,0,1,1,1,1,0
18,1,2,2,1,3,4,4.0,2.0,4.0,4,1,1,0,1,0,-1,0
17,2,1,2,2,4,2,5.0,1.0,2.0,5,0,0,0,1,1,1,0
17,2,4,1,2,4,3,2.0,1.0,1.0,5,0,1,1,1,1,1,0
15,1,1,3,2,4,2,1.0,1.0,2.0,2,0,0,0,1,1,1,0
15,3,3,1,4,4,3,3.0,1.0,1.0,4,0,1,0,0,1,1,0
17,1,1,1,3,4,3,2.0,1.0,2.0,4,0,1,1,1,1,1,0
1987,3,2,1,1,5,3,4.0,2.0,2.0,5,1,0,0,1,0,1,0
18,1,1,2,3,5,3,2.0,1.0,,4,0,1,1,1,1,1,0
...
```

data2.csv структура

- age — возраст
- Medu — уровень образования матери (по некоторой условной шкале)
- Fedu — уровень образования отца (по некоторой условной шкале)
- traveltime — время в пути до школы (1: < 15 мин., 2: от 15 до 30 мин., 3: от 30 мин. to 1 ч. или 4 - > 1 ч.)
- studytime — время, затрачиваемое на занятия вне школы (1:< 2 ч., 2: от 2 до 5 ч., 3:от 5 до 10 ч. или 4:> 10 ч.)
- famrel — насколько хорошие отношения в семье у учащегося (по некоторой условной шкале)
- freetime — количество свободного времени вне школы (по некоторой условной шкале)
- goout — время, затрачиваемое на общение с друзьями (по некоторой условной шкале)
- Dalc — количество употребления алкоголя в учебные дни (по некоторой условной шкале)

data2.csv структура

- Walc — количество употребления алкоголя в неучебные дни (по некоторой условной шкале)
- health — уровень здоровья (по некоторой условной шкале)
- sex_M — пол: мужской (1) или женский (0)
- address_U — живет ли учащийся в городе (1) или в пригороде (0)
- famsize_LE3 — размер семьи: не больше 3 человек (1) или больше (0)
- Pstatus_T — живут ли родители вместе (1) или отдельно (0)
- nursery — посещал ли учащийся детский сад
- plans_university — планирует ли учащийся поступать в университет (-1 или 1)
- past_failures — количество неудовлетворительных оценок по другим предметам ранее (от 0 до 4)

scores.csv

```
70.0
85.0
55.0
80.0
70.0
50.0
50.0
50.0
0.0
0.35
0.5
...
```

Простые решения

- Прочитать файлы data2.csv, scores.csv.
- Не исправляя ошибок с помощью линейной регрессии найти погрешность
- Исправив ошибки с помощью линейной регрессии найти погрешность. Сравнить.

По-простому

```
np.set_printoptions(precision=4, linewidth = 120, suppress=
def lin_reg(y, x):
    x = np.hstack((np.ones(len(x)).reshape((-1,1)), x))
    A = x.transpose().dot(x)
    b = x.transpose().dot(y)
    return np.linalg.solve(A,b)

Y = np.loadtxt('scores.csv')
X = np.genfromtxt('data2.csv', skip_header=1,
                  delimiter=',')
print('Y, X.shape=', Y.shape, X.shape )
> Y, X.shape= (649,) (649, 18)
print(np.count_nonzero(np.isnan(X)))
> 17
```

По-простому

```
N = len(Y)
X[np.isnan(X)] = 0
w = lin_reg1(X,Y)
err  = w[0] + X.dot(w[1:]) - Y
sigma = np.sqrt( np.sum(err*err)/N )
print(f'sigma={sigma:6.3f}')
> sigma=14.362
```

Правим Y<1 и возраст 1987

```
...
Y[ (0<Y) & (Y<5)] *=100

X0 = X[:, 0]
X0[X0>30] *= -1
X0[X0<0] += 2005
...
print(f'sigma={sigma:6.3f}')
> sigma=13.533 (было 14.362)
```

Выбросим тех, кто не сдавал

```
...
idx = ~(Y==0)
Y = Y[idx]
X = X[idx]
...
print(f'sigma={sigma:6.3f}')
> sigma=10.994 (было 13.533)
```

attendance.csv

```
0;1;2;3;4;5;6;7;8;9;10;11;12; ... ;25;26;27;28;29;30;31
+;+;+;+;+;+;+;;;+;;+;+;;+;+;+;+;+;+;+;+;+;+;+;+;+;;+;+;;+;+
+;;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;;+;+;+;+;+;+;+;+;+;+;+;+
+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+
+;+;;;+;+;+;+;;;+;;+;+;;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;;;+
+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+
+;+;;+;+;+;+;+;;;+;+;+;+;+;;;+;;+;+;+;;;+;+;+;+;+;;+;+;+
+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+;+
+;+;;+;;+;+;+;+;+;;;+;+;+;+;+;;;;+;+;+;+;;;+;+;+;+;+;+
```

Добавим посещаемость

```
...
att = np.genfromtxt(r't:\attendance.csv', skip_header=1,
                    delimiter=';', dtype=str)
att[att=='+'] = '1'
att[att==''] = '0'
att = att.astype(int)
X = np.hstack((X, att))
...
print(f'sigma={sigma:6.3f}')
> sigma=10.696 (было 10.994)
```

school_support.txt

```
575
56
481
547
559
346
10
193
597
562
616
210
...
```

Поддержка школы

```
...
supp = np.loadtxt(r't:\school_support.txt').astype(int)
xsupp = np.zeros(N)
for s1 in supp:
   xsupp[s1-1] = 1
X = np.hstack((X, xsupp.reshape((-1,1))))
> X.shape= (634, 51)
...
print(f'sigma={sigma:6.3f}')
> sigma=10.684 (было 10.696)
```

Решение задачи о ЕГЭ

- читаем data.csv — основная таблица с информацией
- исправляем ошибки
- читаем attendance.csv — таблица посещений
- исправляем ошибки
- читаем school_support.txt - список финансовой поддержки
- исправляем ошибки
- читаем scores.csv — оценки
- исправляем ошибки
- Объединяем данные в одну матрицу
- С помощью линейной регрессии найти погрешность.

Сокращение размерности

```
def optimal_basis(V):# оптимальный базис для строк матрицы
    '''
    В строках (my) матрицы V лежат вектора (длины mx).
    Возвращает пару (w,B), где
    B - оптимальный базис для этих векторов,
        то есть матрицу mx*mx, в СТОЛБЦАХ которой лежат эти
    w - собственные значения
    '''
    my, mx = V.shape
    X2 = np.zeros((mx, mx))  # sum(x_i^2)
    for v1 in V:    # по строкам матрицы V
        X2 += np.outer(v1,v1)
    X2 /= my
    w, B = np.linalg.eigh(X2)
    return w[::-1], B[:, ::-1]
```

Сокращение размерности

```
eval, basis = optimal_basis(X)
print(np.sqrt(eval), '=values')
```

```
[19.5523  2.7475  1.6009  1.4765  1.4294  1.1774  0.9394
  0.8862  0.8767  0.7748  0.6794  0.6432  0.5788  0.508
  0.479   0.4456  0.4329  0.3955  0.3644  0.3563  0.3511
  0.3433  0.3332  0.3275  0.3268  0.3212  0.3171  0.308
  0.3062  0.3015  0.2989  0.2966  0.2905  0.2872  0.2816
  0.2774  0.2723  0.2716  0.2658  0.2624  0.2579  0.2574
  0.2514  0.243   0.2405  0.2369  0.2298  0.2242  0.2145
  0.2091  0.1922] =values
```

Задание. Проверить, как влияет сокращение размерности на точность предсказания?

Метод ближайших соседей

Основная идея: близкие объекты выдают похожие ответы.
Найти K ближайших объектов к интересующему объекту x в обучающей выборке.
Если $K = 1$, то $y(x)$ полагаем равному
y(от найденного соседа).
Если $K > 1$, то $y(x)$ полагаем равному средневзвешенному значению y от всех найденных соседей.
Параметры:

- число соседей K
- функция близости $r(x, y)$

Метод ближайшего соседа

Достоинства:

- Нужна только функция близости
- Простая логика
- Не требует обучения
-

Недостатки

- Медленная классификаци.
- Много памяти
- Точность ухудшается с ростом размерности

Метрика

Чаще всего используется обычная еклидова метрика (L_2):

$$r(x, z) = \sqrt{\sum (x_i - z_i)^2}$$

Можно использовать метрику L_1:

$$r(x, z) = \sum |x_i - z_i|$$

Столбцы в обучающей матрице могут сильно различаться по смыслу и по величине. Поэтому их в начале имеет смысл нормализовать некоторым образом.

Нормализация на numpy

Из каждого столбца матрицы X вычесть его среднее значение:

```
X -= X.mean(axis=0)
```

Из каждого столбца матрицы X вычесть его минимальное значение:

```
X -= X.min(axis=0)
```

Каждый столбец матрицы X поделить на его ср.кв.отклонение:

```
X /= X.std(axis=0)
```

Каждый столбец матрицы X поделить на его максимальное значение:

```
X /= X.min(axis=0)
```

Несколько соседей

Если взять несколько соседей, ответ надо брать взвешенно:

r :	r_0	r_1	r_2
y :	y_0	y_1	y_2
weights :	$1/r_0$	$1/r_1$	$1/r_2$

$$y = \frac{y_0/r_0 + y_1/r_1 + y_2/r_2}{1/r_0 + 1/r_1 + 1/r_2}$$

Метод ближайшего соседа

```
from sklearn.neighbors import NearestNeighbors
nbrs = NearestNeighbors(n_neighbors=2,
                        algorithm='ball_tree').fit(X)
distances, indices = nbrs.kneighbors(X)
print(distances[:4], '=distances\n')
print(indices[:4], '=indices\n')
```

```
[[0.     3.1623]
 [0.     2.4495]
 [0.     2.6458]
 [0.     3.6056]] =distances
```

```
[[  0 278]
 [  1  69]
 [  2 591]
 [  3  63]] =indices
```

Метод ближайшего соседа

```
from sklearn.neighbors import NearestNeighbors
nbrs = NearestNeighbors(n_neighbors=2,
                        algorithm='ball_tree').fit(X)
distances, indices = nbrs.kneighbors(X)
print(distances[:4], '=distances\n')
print(indices[:4], '=indices\n')
```

```
[[0.     3.1623]
 [0.     2.4495]
 [0.     2.6458]
 [0.     3.6056]] =distances
```

```
[[  0 278]
 [  1  69]
 [  2 591]
 [  3  63]] =indices
```

Метод ближайшего соседа

```
indices1 = indices[:,1]
Y1 = Y[indices1]
print(Y[:10], '=Y')
print(Y1[:10], '=Y1')

[65. 60. 40. 50. 55. 50. 75. 50. 40. 75.] =Y
[55. 50. 45. 60. 55. 70. 60. 50. 55. 55.] =Y1

sigma=17.234
```

Если изменить веса:

```
X = X*w[1:]

sigma=15.538
```