Задача 4 (Регрессор на плоскости)

Стельмах Татьяна

Январь 2021

Содержание

1 Введение

Задача: Построить регрессор на плоских данных, выдвинуть предположения о модели - линейная, полиномиальная, экспоненциальная, продемонстрировать работу программы на примерах (в результате) и сделать краткие выводы.

2 Описание метода

При решении задачи будем использовать библиотеку `Sklearn` для построения линейной, экспоненциальной и полиномиальной регрессионных моделей. Также с помощью библиотеки `numpy` будем выяснять, какая модель подходит лучше для некоторого набора данных.

3 Описание данных

Данные будем генерировать самостоятельно, для разнообразия в конце привёдем результат работы на каких-нибудь более приближенных к реальности данных (например, данные о зарплате и стаже работника), то есть в первом случае данные генерируются в теле самой программы, а во втором мы получаем csv-файл.

4 Описание постановки эксперимента

Библиотеки, которые будут использованы для решения задачи:

```
from math import sqrt, exp, fabs, sin
import numpy as np
import csv
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import random
```

Для начала мы генерируем наши данные, в качестве примера разместим здесь генерацию одного из тестов:

```
if test == 2:
    for j in range(4):
        for i in range(1,20):
            y.append(np.arctan(5+i) + 2*i + random.randint(-4,27))
            x.append(i)
```

Снизу привёдено описание того, как строились модели.

1. Опишем построение линейной регрессионной модели:

   ```
   X = np.array(x).reshape(-1,1)          # создаём двумерный массив для обработки его моделью
   model1 = LinearRegression().fit(X, Y) # обучаем модель
   y_pred1 = model1.predict(X).tolist()  # предсказываем Y по X
   # изображаем на графике полученную зависимость
   scatter1 = plt.plot(x, y_pred1, c = 'red', label = u'Линейная регрессия')
   ```

2. Опишем построение экспоненциальной модели:

   ```
   i = 0
   y_0 = y
   x_0 = x
   for y0 in y:         # избегаем 0
       if y0 == 0:
           x_0.pop(i)
           y_0.pop(i)
           i-=1
       i+=1
   expo = np.polyfit(x_0, np.log(y_0), 1, w=np.sqrt(y_0))
   #  находит коэффициенты полинома p(x) степени n, который аппроксимирует функцию y(x) в смысле метода наименьших квадратов
   y_pred3 = []
   x.sort()             # чтобы построить функцию
   for x1 in x:
       y_pred3.append(exp(expo[1]) * exp(expo[0] * x1))
   scatter1 = plt.plot(x, y_pred3, c = 'yellow', label = u'Экспоненциальная регрессия')
   ```

   Какой же хитростью воспользовались? Мы ищем коэффициенты A и B в случае $y = A{\cdot}\mathrm{e}^{B\cdot x}$, возмём же натуральный логарифм от обеих частей, тогда logy= logA + B $\cdot x$. Получается, что нам достаточно найти коэффициенты полинома первой степени, а затем восставновить y.

3. Опишем построение полиномиальной модели:

   ```
   poly_reg = PolynomialFeatures(degree=4)
   x_poly = poly_reg.fit_transform(X)
   model2 = LinearRegression()
   model2.fit(x_poly,Y)
   ```

```
y_pred2 = model2.predict(x_poly).tolist()
x.sort()
X = np.array(x).reshape(-1,1)
scatter1 = plt.plot(x, model2.predict(poly_reg.fit_transform(X)), c = 'green', label = u'Полиномиальная регрессия')
```

Теперь, когда предсказанные значения и модели имеются, мы можем сказать, какая регрессионная модель больше подходит для нашей ситуации. Для этого вычислим среднеквадратичное (стандартное) отклонение расстояний от данных точек до предсказанных (рассматриваем парами), используя `numpy.std`, он же, в свою очередь, высчитывает его по следующей формуле:

$S = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (a_i - \bar{a})^2}$, где $\bar{a}$ - среднее арифмитическое чисел $a_1, a_2, ..., a_n$

Взглянем на код:

```
dists1 = []   # расстояния до точек, предсказанных линейной регрессией
dists2 = []   # полиномиальной регрессией
dists3 = []   # экспоненциальной регрессией
for j in range(len(x)):
    dists1.append(fabs(y_pred1[j] - y[j]))   # вычисляет дробный модуль
    dists2.append(fabs(y_pred2[j] - y[j]))
    dists3.append(fabs(y_pred3[j] - y[j]))
dists = [round(np.std(dists1), 5), round(np.std(dists2), 5),   round(np.std(dists3), 5)]  # округляет до 5-ти знаков
min_dist = float('inf')
i = 0
indexs = []
for dist in dists:
    if dist < min_dist:
        min_dist = dist      # нашли минимальное среднеквадратичное отклонение среди них
for dist in dists:
    if dist > min_dist - 0.001 and dist < min_dist + 0.001:
        indexs.append(i)
    i+=1
```

Далее строим графики и просим программу вывести результат, то есть какая же модель подходит для той или иной ситуации.

5 Результаты

Посмотрим на результаты (внизу картинки указана выбранная программой модель)

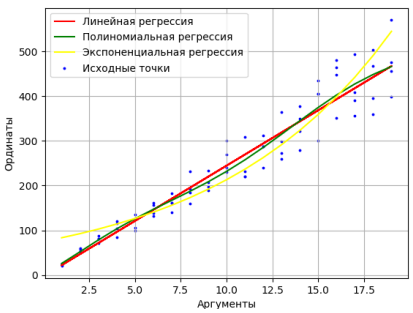

Рисунок 1. Линейная

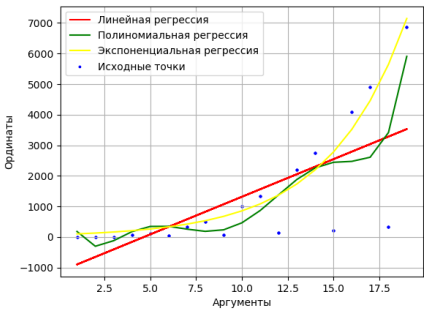

Рисунок 2. Полиномиальная

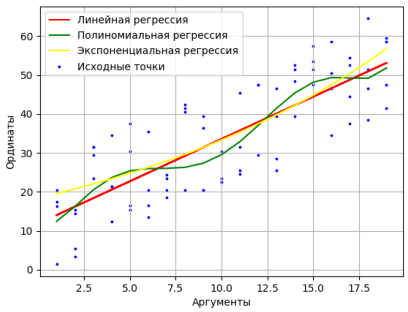

Рисунок 3. Линейная

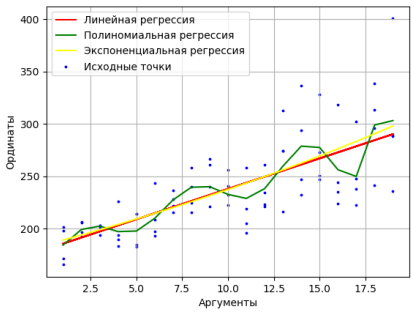

Рисунок 4. Полиномиальная

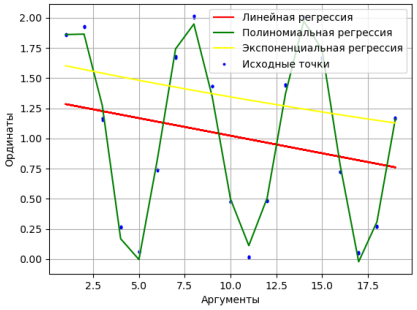

Рисунок 5. Полиномиальная

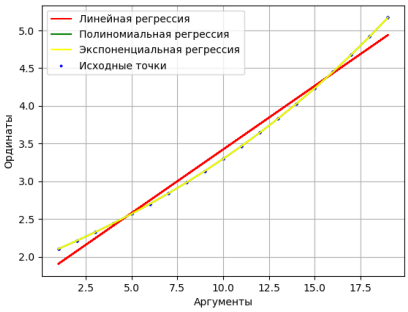

Рисунок 6. Экспотенциальная

Прокомментируем некоторые получившиеся результаты:

1. Тест пятый: Если бы мы работали с реальными данными, то подобный результат скорее всего означал бы тот факт, что модель переобучилась и в реальности модель линейной регрессии была бы куда более привлекательной.
2. Тест шестой: На самом деле программа сочла подходящими 2 модели: экспоненциальную и полиномиальную, что довольно-таки похоже на правду, так как данные генерировались с помощью экспоненты.
3. В остальных тестах результаты близки к реальности.

Теперь посмотрим на работу программы на данных более приближенных к жизни, и даже прикинем заисимость заработной платы от стажа работника (Рисунок 7) Полиномиальная модель, кстати, здесь является наиболее подходящей. Но почему? На самом деле если бы мы использовали линейную регрессию, например, то мы бы предсказали человеку с большим опытом довольно-таки огромные суммы в качестве заработной платы, но в реальной жизни все не так сказочно и полиномиальная регрессионная модель в самом деле будет более удачной, более того, она с меньшими потерями смогла предугадать заработную плату для работника с опытом работы 10 лет.

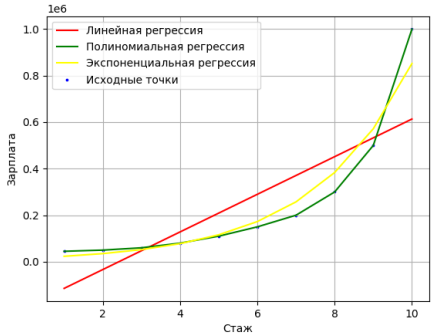

Рисунок 7. Salary(Level)

6 Краткий вывод

Мы построили несколько регрессионных моделей (линейную, полиномиальную и экспоненциальную) и на нашем наборе данных смогли найти наиболее подходящую модель, далее изобразили получившийся результат. В ходе работы выяснили, что полиномиальная регрессионная модель является более эффективной, однако необходимо здраво оценивать результаты, анализировать ситуацию и избегать переобучения.