Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

«Линейная регрессия»

по дисциплине «Системы искусственного интеллекта»

Выполнили:

Студент группы P3332

Чмурова М.В.

Преподаватель:

Бессмертный Игорь Александрович

Санкт-Петербург

2024

Оглавление

Введение 3

Описание метода линейной регрессии 3

Псевдокод метода 3

Выполнение 5

Получение и визуализация статистики по датасету. 5

Визуализация статистики по датасету: 8

Предварительная обработка данных 9

Разделение данных на обучающий и тестовый наборы данных 12

Реализация линейной регрессии с использованием метода наименьших квадратов 13

Построение 3 моделей с различными наборами признаков 14

Оценка производительности с помощью коэффициента детерминации 15

Сравнение результатов 16

Бонус. Введение синтетического признака 17

Заключение 18

Введение

Цель работы: изучить метод линейной регрессии, освоить процесс работы с данными, включая их предварительную обработку, исследование, разделение на обучающий и тестовый наборы, а также построение и оценку модели линейной регрессии на основе метода наименьших квадратов.

Описание метода линейной регрессии

Линейная регрессия — метод машинного обучения, используется для предсказания значения зависимой переменной на основе одной или нескольких независимых переменных.

Цель метода — найти линейную зависимость между целевой переменной и признаками, которая минимизирует разницу между фактическими значениями и предсказанными моделью значениями. Линейная регрессия основывается на методе наименьших квадратов, который позволяет подобрать такие коэффициенты уравнения регрессии, при которых сумма квадратов отклонений между предсказанными и фактическими значениями будет минимальной.

Псевдокод метода

Регрессионная модель представляется в матричном виде следующим образом:

$$y = X\beta$$

Где:

y – целевая переменная (вектор)

X - матрица признаков

β - вектор коэффициентов, который мы ищем.

Чтобы найти коэффициенты β, минимизируется сумма квадратов разностей между фактическими значениями y и предсказанными значениями

$$\hat{y} = X\beta.$$

Коэффициенты β находятся по следующей формуле:

$$\beta = (X^T X)^{-1} X^T y$$

Где:

X^T – транспонированная матрица признаков

$(X^T X)^{-1}$ – обратная матрица для произведения

y – вектор целевых значений

Следующими шагами необходимо:

- Построить модель с полученными коэффициентами
- Оценить качество модели (Сделать предсказания для тестовой выборки и рассчитать коэффициент детерминации (R^2), чтобы оценить точность модели:

$$R^2 = 1 - \frac{\sum(y_{true} - y_{pred})^2}{\sum(y_{true} - \bar{y})^2}$$

Где:

y_{true} – истинные значения целевой переменной

y_{pred} – предсказанные значения

$\bar{y}$ – среднее значение

$\sum(y_{true} - y_{pred})^2$ –квадрат остатков (SSR)

$\sum(y_{true} - \bar{y})^2$ – общее значение (SST)

- Вывести значения коэффициентов и оценку качества модели (R^2)

Выполнение

Получение и визуализация статистики по датасету.

Программа для получения значений:

```
import numpy as np
import pandas as pd

data_frame = pd.read_csv('Student_Performance.csv')

pd.set_option('display.max_columns', None)
print(data_frame.describe())
```

Вывод программы:

```
          Hours Studied  Previous Scores   Sleep Hours  \
count   10000.000000     10000.000000  10000.000000
mean        4.992900        69.445700      6.530600
std         2.589309        17.343152      1.695863
min         1.000000        40.000000      4.000000
25%         3.000000        54.000000      5.000000
50%         5.000000        69.000000      7.000000
75%         7.000000        85.000000      8.000000
max         9.000000        99.000000      9.000000

       Sample Question Papers Practiced  Performance Index
count                      10000.000000       10000.000000
mean                           4.583300          55.224800
std                            2.867348          19.212558
min                            0.000000          10.000000
25%                            2.000000          40.000000
50%                            5.000000          55.000000
75%                            7.000000          71.000000
max                            9.000000         100.000000
```

- **Количество**:

count = 10 000 – количество элементов в датасете

- **Среднее значение:**

mean = 4.993 – часов в среднем проведено за учебой

mean = 69.446 – средняя оценка на тестах

mean = 6.531 – среднее количество часов сна

mean = 4.583 – количество работ, решенное студентами для практики

mean = 55.225 – средний performance студента на курсе

$$mean = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- **Стандартное отклонение**:

std = 2.589 – отклонение по времени, потраченному на учебу

std = 17.343 – отклонение по результатам теста

std = 1.696 – отклонение по количеству сна

std = 2.867 – отклонение по количеству работ, решенных для практики

std = 19.213 – отклонение по performance студентов

$$std = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

- **Минимум:**

min = 1.0 – минимальное кол-во проведенное за учебой студентом

min = 40.0 – минимальная оценка студентом за тест

min = 4.0 – минимальное кол-во часов сна

min = 0.0 – минимальное кол-во решенных работ для практики

min = 10.0 – минимальный performance студента

- **Максимум:**

max = 9.0 - максимальное кол-во проведенное за учебой студентом

max = 99.0 – максимальная оценка студентом за тест

max = 9.0 – максимальное кол-во часов сна

max = 9.0 – максимальное кол-во решенных работ для практики

max = 100.0 – максимальный performance студента

- **Квантиль (25%):**

3.0 – 25% студентов учились <= 3 часа

54.0 – 25% студентов получили оценку <= 54 на тестах

5.0 – 25% студентов спали <= 5 часов

2.0 – 25% студентов решили <= 2 работы

40.0 - перформанс 25% студентов был < 40

- **Квантиль (50%):**

5.0 – половина студентов учились <= 5 часа

69.0 – половина студентов получили оценку <= 69 на тестах

7.0 – половина студентов спали <= 7 часов

5.0 – половина студентов решили <= 2 работы

55.0 - перформанс половины студентов был < 55

- **Квантиль (75%):**

7.0 – 75% студентов учились <= 7 часов.

85.0 – 75% студентов получили оценку <= 85 на тестах

8.0 – 75% студентов спали <= 8 часов

7.0 – 75% студентов решили <= 7 работ

71.0 - перформанс половины студентов был < 71

Визуализация статистики по датасету:

Программа, используемая для визуализации:

```
import matplotlib.pyplot as plt

data_frame.hist(figsize=(15, 10))
plt.suptitle("Гистограммы признаков")
plt.show()
```

Результат работы программы:

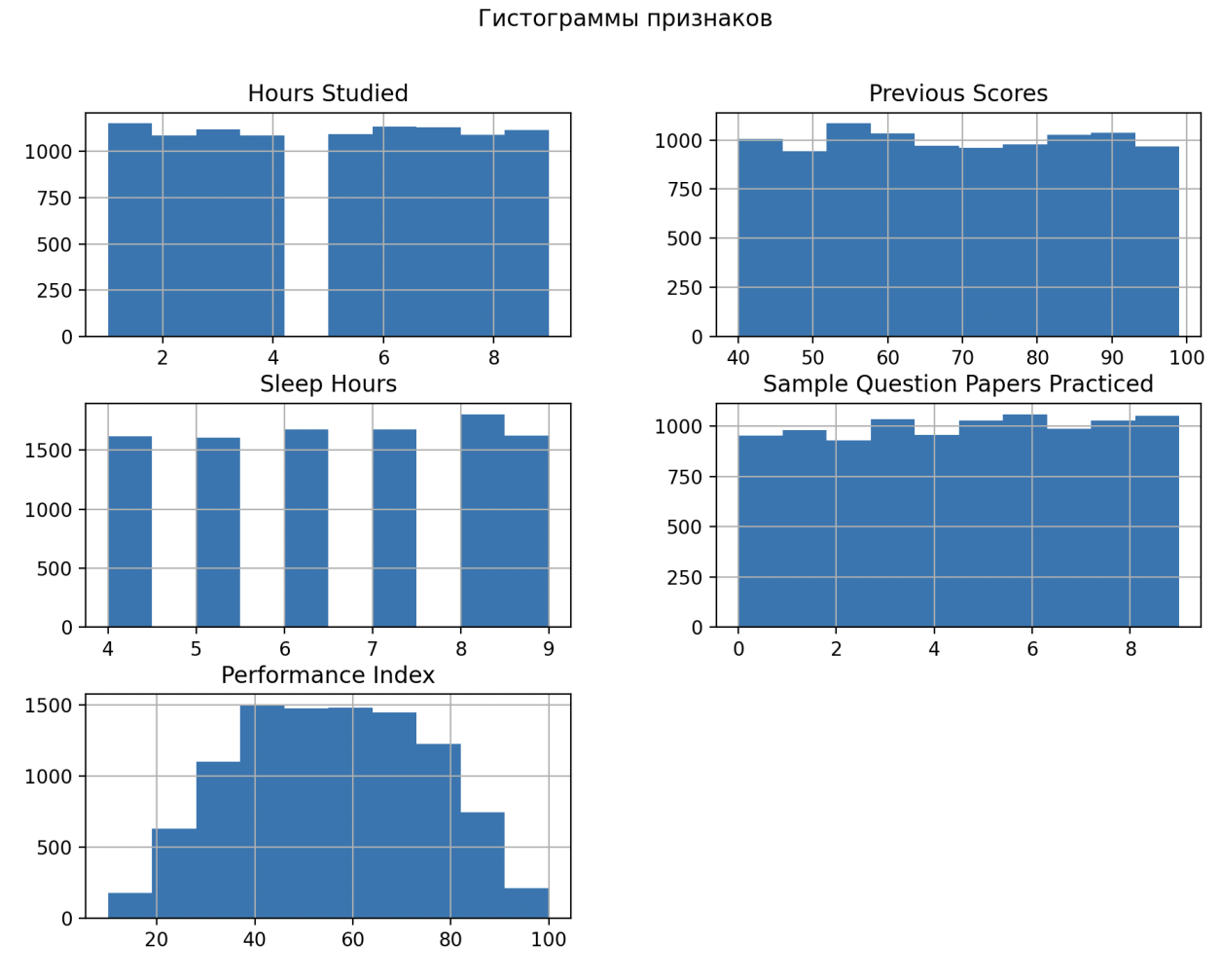

Рисунок 1. Визуализация признаков в форме гистограммы

Предварительная обработка данных

- **Обработка отсутствующих значений:**

Программа, используемая для обработки:

```
print(data_frame.isnull().sum())
```

Результат работы программы:

```
Hours Studied                       0
Previous Scores                     0
Extracurricular Activities          0
Sleep Hours                         0
Sample Question Papers Practiced    0
Performance Index                   0
dtype: int64
```

Таким образом, имеющийся датасет не имеет пропусков – следовательно их не нужно заполнять средним значением, медианой, часто встречающимся значением.

- **Кодирование категориальных признаков** - процесс преобразования категориальных данных в числовые значения, которые можно использовать в моделях машинного обучения

Воспользуемся таким методом кодирования категориальных признаков, как Label Encoding (Метки). Датасет студентов имеет 1 категориальных признак - Extracurricular Activities, который разделяет всех студентов на две группы – Yes / No. С помощью метода кодирования заменяется Yes на 1, а No на 0

Программа, используемая для кодирования:

```
data_frame['Extracurricular Activities'] =
data_frame['Extracurricular Activities'].map({'Yes': 1, 'No':
0})
print(data_frame.head())
```

Результат работы программы для первых 5 значений:

```
   Hours Studied  Previous Scores  Extracurricular Activities  Sleep
Hours  \
0              7               99                           1      9
1              4               82                           0      4
2              8               51                           1      7
3              5               52                           1      5
4              7               75                           0      8

   Sample Question Papers Practiced  Performance Index
0                                 1               91.0
1                                 2               65.0
2                                 2               45.0
3                                 2               36.0
4                                 5               66.0
```

- **<u>Нормализация данных</u>** – преобразование числовых данных к единому масштабу необходимое из-за чувствительности линейной регрессии к масштабу признаков

Проводится нормализация данных с помощью min-max нормализации – приведению всех значений к диапазону от 0 до 1 с использованием формулы:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Где:

x – исходное значение признака

x_{min}, x_{max} – мин. и макс. значение признака

Программа, используемая для нормализации:

```
def min_max_normalize(column):
    return (column - np.min(column)) / (np.max(column) -
np.min(column))

columns_to_scale = ['Hours Studied', 'Previous Scores', 'Sleep
Hours', 'Sample Question Papers Practiced']

for col in columns_to_scale:
    data_frame[col] = min_max_normalize(data_frame[col])

print(data_frame.head())
```

Результат работы программы для первых 5 значений:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	\
0	0.750	1.000000	Yes	1.0	
1	0.375	0.711864	No	0.0	
2	0.875	0.186441	Yes	0.6	
3	0.500	0.203390	Yes	0.2	
4	0.750	0.593220	No	0.8	

	Sample Question Papers Practiced	Performance Index
0	0.111111	91.0
1	0.222222	65.0
2	0.222222	45.0
3	0.222222	36.0
4	0.555556	66.0

Разделение данных на обучающий и тестовый наборы данных

Программа для разделения данных:

```
# Деление данных на признаки (X) и целевую переменную (y)
X = data_frame.drop('Performance Index', axis=1).values
y = data_frame['Performance Index'].values

# Определение размера обучающей выборки (80% от общего
количества данных)
train_size = int(0.8 * len(X))

# Перемешивание индексов для случайного разделения
indices = np.random.permutation(len(X))

# Разделение данных на обучающую и тестовую выборки
train_indices = indices[:train_size]
test_indices = indices[train_size:]

X_train, X_test = X[train_indices], X[test_indices]
y_train, y_test = y[train_indices], y[test_indices]

# Проверка размеров полученных наборов
print(f'Размер обучающей выборки: {X_train.shape}')
print(f'Размер тестовой выборки: {X_test.shape}')
```

Результат работы программы:

```
Размер обучающей выборки: (8000, 5)
Размер тестовой выборки: (2000, 5)
```

Таким образом по признакам X будет предсказано целевая переменная Y. Обучающая выборка (анализ зависимостей между признаками (input) и целевой переменной (output), настройка своих коэффициентов) имеет размер 80% от тестовой. Тестовая выборка используется для проверки предсказывания моделью значения на новых данных

Реализация линейной регрессии с использованием метода наименьших квадратов

Программа для реализации линейной регрессии:

```
X_train_with_bias = np.c_[np.ones(X_train.shape[0]), X_train]
beta = np.linalg.inv(X_train_with_bias.T @ X_train_with_bias) @ X_train_with_bias.T @ y_train
y_train_pred = X_train_with_bias @ beta

print("Коэффициенты модели:", beta)
print("Пример предсказанных значений:", y_train_pred[:5])

X_test_with_bias = np.c_[np.ones(X_test.shape[0]), X_test]
y_test_pred = X_test_with_bias @ beta

print("Пример предсказанных значений для теста:", y_test_pred[:5])
```

Результат работы программы:

```
Коэффициенты модели: [11.36582465 22.85477371 60.149184
0.60956703  2.44995122  1.74606898]
Пример предсказанных значений: [26.05195268 35.93836042
64.70966204 41.5675998  51.72744308]
Пример предсказанных значений для теста: [35.75673938
84.49682028 28.62476998 78.83289161 76.7815496 ]
```

Шаги реализации:

1. Добавляется столбец единиц в начало X_train. Этот столбец отвечает за свободный член (смещение) в уравнении линейной регрессии.
2. Нахождение коэффициентов beta для линейной регрессии с помощью метода наименьших квадратов.
3. Вычисление предсказанных значения y_train_pred для обучающей выборки, используя найденные коэффициенты beta.
4. Вывод рассчитанных коэффициенты и несколько первых предсказанных значений для обучающей выборки и для тестовой выборки.

Таким образом, были найдены коэффициенты линейной регрессии. Получена линейная функция для описания зависимости между предикатами и зависимой переменной.

Построение 3 моделей с различными наборами признаков

Программа для построения 3 моделей:

1. Модель №1 – 2 набора признаков (Hours Studied, Sample Question Papers Practiced)
2. Модель №2 – 3 набора признаков (Hours Studied, Previous Scores, Sleep Hours)
3. Модель №3 – весь набор признаков

```
# Предсказываем значения коэффциентов для модели
def build_and_evaluate_model(X, y):
    X_with_bias = np.c_[np.ones(X.shape[0]), X]
    beta = np.linalg.inv(X_with_bias.T @ X_with_bias) @ 
X_with_bias.T @ y
    y_pred = X_with_bias @ beta
    return beta, y_pred

X_model_1 = X_train
beta_1, y_train_pred_1 = build_and_evaluate_model(X_model_1, 
y_train)
X_model_2 = X_train[:, [0, 1, 3]]
beta_2, y_train_pred_2 = build_and_evaluate_model(X_model_2, 
y_train)
X_model_3 = X_train[:, [0, 4]]
beta_3, y_train_pred_3 = build_and_evaluate_model(X_model_3, 
y_train)

print("Коэффициенты для 2 признаков:", beta_3)
print("Коэффициенты для 3 признаков:", beta_2)
print("Коэффициенты для всех признаков:", beta_1)
```

Результат работы программы:

```
Коэффициенты для 2 признаков: [43.25712987 21.72239361
2.16142495]
Коэффициенты для 3 признаков: [12.59919542 22.89231931
60.08380758  2.42642551]
Коэффициенты для всех признаков: [11.41521007 22.8424306
60.06120481  0.59770266  2.4259516   1.8121935 ]
```

Оценка производительности с помощью коэффициента детерминации

Программа для оценки производительности, используя коэффициент детерминации:

```
# Функция для вычисления коэффициента детерминации
def r_squared(y_true, y_pred):
    ss_res = np.sum((y_true - y_pred) ** 2)
    ss_tot = np.sum((y_true - np.mean(y_true)) ** 2)
    return 1 - (ss_res / ss_tot)

r2_model_1 = r_squared(y_train, y_train_pred_1)
r2_model_2 = r_squared(y_train, y_train_pred_2)
r2_model_3 = r_squared(y_train, y_train_pred_3)

print("R^2 для 2 признаков:", r2_model_1)
print("R^2 для 3 признаков:", r2_model_2)
print("R^2 для всех признаков:", r2_model_3)
```

Результат работы программы:

```
R^2 для 2 признаков: 0.14478323477322164
R^2 для 3 признаков: 0.9876150719078294
R^2 для всех признаков: 0.9886928717991958
```

Сравнение результатов

1. Использование модели со всеми признаками:

Полученный коэффициент детерминации: R = 0.9887) — наилучший результат, почти идеальная точность.

Важные признаки: Previous Scores (60.08), Hours Studied (22.81).

Подходит для максимальной точности, если доступны все данные.

2. Использование модели с 3 признаками:

Полученный коэффициент детерминации: R = 0.9875 — точность близкая к модели со всеми признаками

Важные признаки: Previous Scores (60.09), Hours Studied (22.83).

Вывод: более экономная модель с высокой точностью. Использовать, если нужно меньше признаков.

3. Использование модели с 2 признаками:

Полученный коэффициент детерминации: R = 0.1412 — очень низкая точность.

Важные признаки: Hours Studied (22.11), Sample Question Papers Practiced (1.83).

Вывод: не рекомендуется, так как двух признаков недостаточно для точных предсказаний.

Бонус. Введение синтетического признака

Синтетический признак – новый признак, созданный на основе существующих данных

Вводится признак, который показывает баланс между тем, сколько студент занимает, сколько отдыхает и участвует во внеучебных активностях - Relaxation Index:

$$RI = \frac{SleepHours \cdot (1 + ExtracurricularActivities)}{HoursStudied + 1}$$

Индекс будет выше для студентов, которые уделяют больше времени сну и занятиям внеучебной активности для поддержания высокого уровня расслабления и общего хорошего самочувствия

Программа для введения синтетического признака:

```
# Создаём новый признак в датафрейме
data_frame['Relaxation Index'] = data_frame['Sleep Hours'] * (1 + data_frame['Extracurricular Activities']) / (data_frame['Hours Studied'] + 1)
X_train_with_relaxation = np.c_[X_train, data_frame['Relaxation Index'].iloc[:X_train.shape[0]]]
beta_relaxation, y_train_pred_relaxation = build_and_evaluate_model(X_train_with_relaxation, y_train)
r2_relaxation = r_squared(y_train, y_train_pred_relaxation)
print("R^2 для синтетического признака:", r2_relaxation)
```

Результат работы программы:

```
R^2 для синтетического признака: 0.988845483737806
```

Таким образом, был получен хороший синтетический признак с высоким коэффициентом детерминации

Заключение

В ходе данной лабораторной работы была исследована линейная регрессия и ее применение для предсказания целевой переменной в выбранном датасете. Проведен анализ и предварительная обработка данных, включая нормализацию и кодирование категориальных признаков, а также реализована линейная регрессия методом наименьших квадратов. Построены и оценены три модели с разными наборами признаков; результаты оценивались с помощью коэффициента детерминации R^2.

Для улучшения модели также введен синтетический признак, что повысило качество предсказаний. Полученные результаты показали, что правильно выбранные признаки и новые синтетические переменные способны улучшить точность линейной регрессии и повысить ее обобщающую способность.