|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| доцент, к.т.н., доцент | / |  | / |  | / | В. В. Мышко |
| (должность, учёная степень, звание) |  | (подпись) |  | (дата защиты) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

«Однофакторный регрессионный анализ»

ПО КУРСУ: «ОБРАБОТКА ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТУ ВЫПОЛНИЛ СТУДЕНТ: | 4134К | / | Самарин Д. В. |
|  | (номер группы) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | / |  | / | 18.02.2025 |
|  |  | (подпись студента) |  | (дата отчета) |

Цель

Целью лабораторной работы является применение методов однофакторного регрессионного анализа для построения квадратичной и линейной регрессий по заданным экспериментальным данным и оценка их адекватности.

Задание на лабораторную работу

На основе заданного массива данных:

• построить уравнение регрессии в виде алгебраического полинома второй степени;

• проверить адекватность уравнения регрессии;

• проверить значимость коэффициентов регрессии.

Расчеты произвести в скалярной и матричной форме. Порядок выполнения задания:

1. Составить систему нормальных уравнений, используя массив экспериментальных данных (таблица 4.1);

2. Найти оценки коэффициентов регрессии посредством решения системы нормальных уравнений;

3. При расчетах в матричной форме составить матричное уравнение с вектором неизвестных оценок коэффициентов регрессии и найти его решение;

4. Проверить адекватность построенного уравнения регрессии экспериментальным данным по критерию Фишера при уровне значимости α = 0,01;

5. Проверить значимость коэффициентов регрессии по критерию Стьюдента при таком же уровне значимости;

6. Повторно проверить адекватность уравнения регрессии после исключения незначимых коэффициентов.

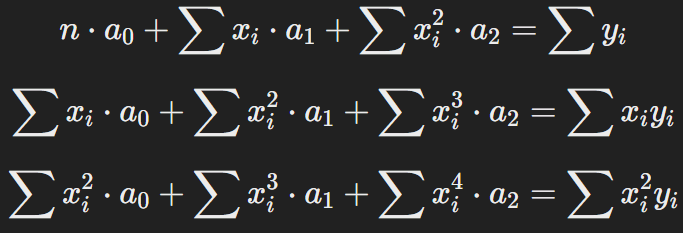
Ход работы

**Вариант 96**

****

Решение:

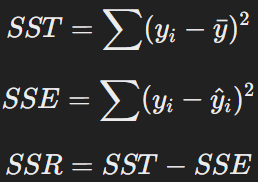
**Нормальные уравнения для квадратичной модели:**



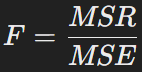
**Матричный метод для нахождения коэффициентов:**



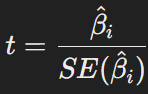
**Сумма квадратов ошибок (SSE), сумма квадратов регрессии (SSR) и общая сумма квадратов (SST):**



**F-статистика**:



**t-статистика для коэффициентов регрессии:**

****

**Критическое значение t при уровне значимости α=0.01:**

****

**Результаты выполнения работы**

В ходе работы был реализован алгоритм.

С работоспособностью алгоритма можно ознакомиться по ссылке ниже:

<https://ehufh87ji5utt5wwbwqdg4.streamlit.app/>

Даны экспериментальные данные:

X = [−1,0,1,2,3], y = [15,12,6,1,−2]

Число наблюдений: n=5n = 5n=5

Построение квадратичной модели:

Для квадратичной модели y=a0+a1x+a2x^2 были рассчитаны коэффициенты методом нормальных уравнений и методом наименьших квадратов.

Коэффициенты (скалярный и матричный метод):

a0​=10.8286, a1​=−4.6429, a2​=0.0714

Оценка адекватности модели

* Общая сумма квадратов (SST): 205.2000205.2000205.2000
* Сумма квадратов ошибок (SSE): 2.62862.62862.6286
* Сумма квадратов регрессии (SSR): 202.5714202.5714202.5714

F-статистика для квадратичной модели: F=77.0652, p-value: 0.0128.

Проверка значимости коэффициентов:

Коэффициенты a0​ и a1​ значимы с p-value меньше 0.05.

Коэффициент a2 оказался незначимым, так как его t-статистика: 0.2331 меньше критического значения tкрит = 9.9248.

Построение линейной модели

После исключения коэффициента a2​ был построен линейный регрессионный анализ y=b0+b1x.

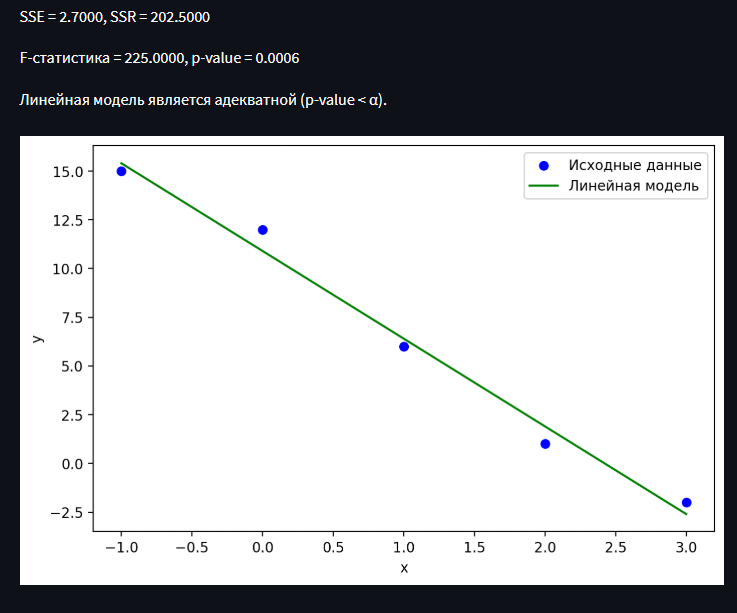
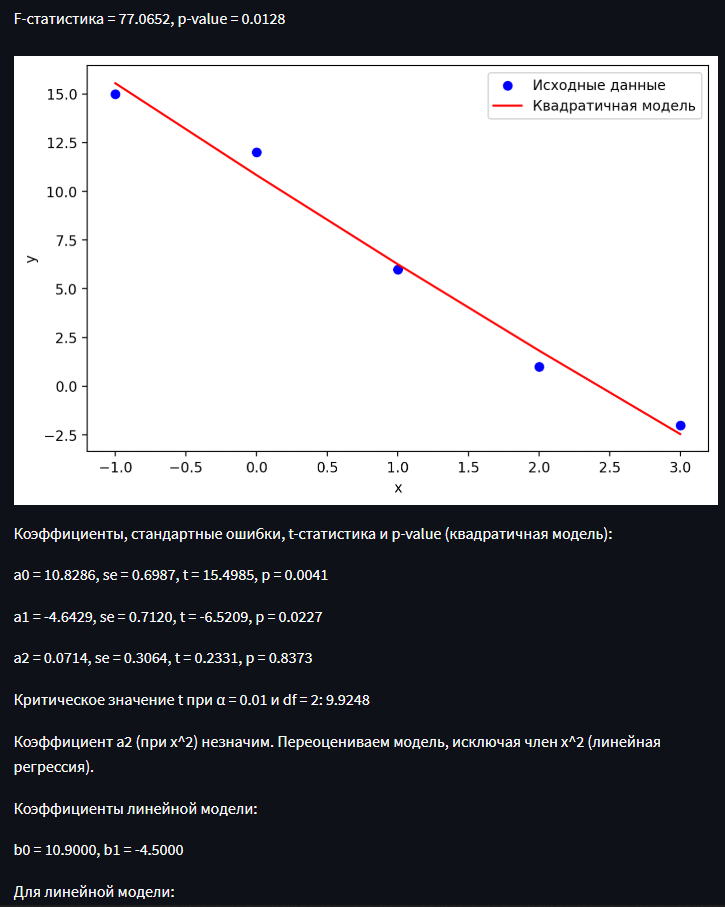
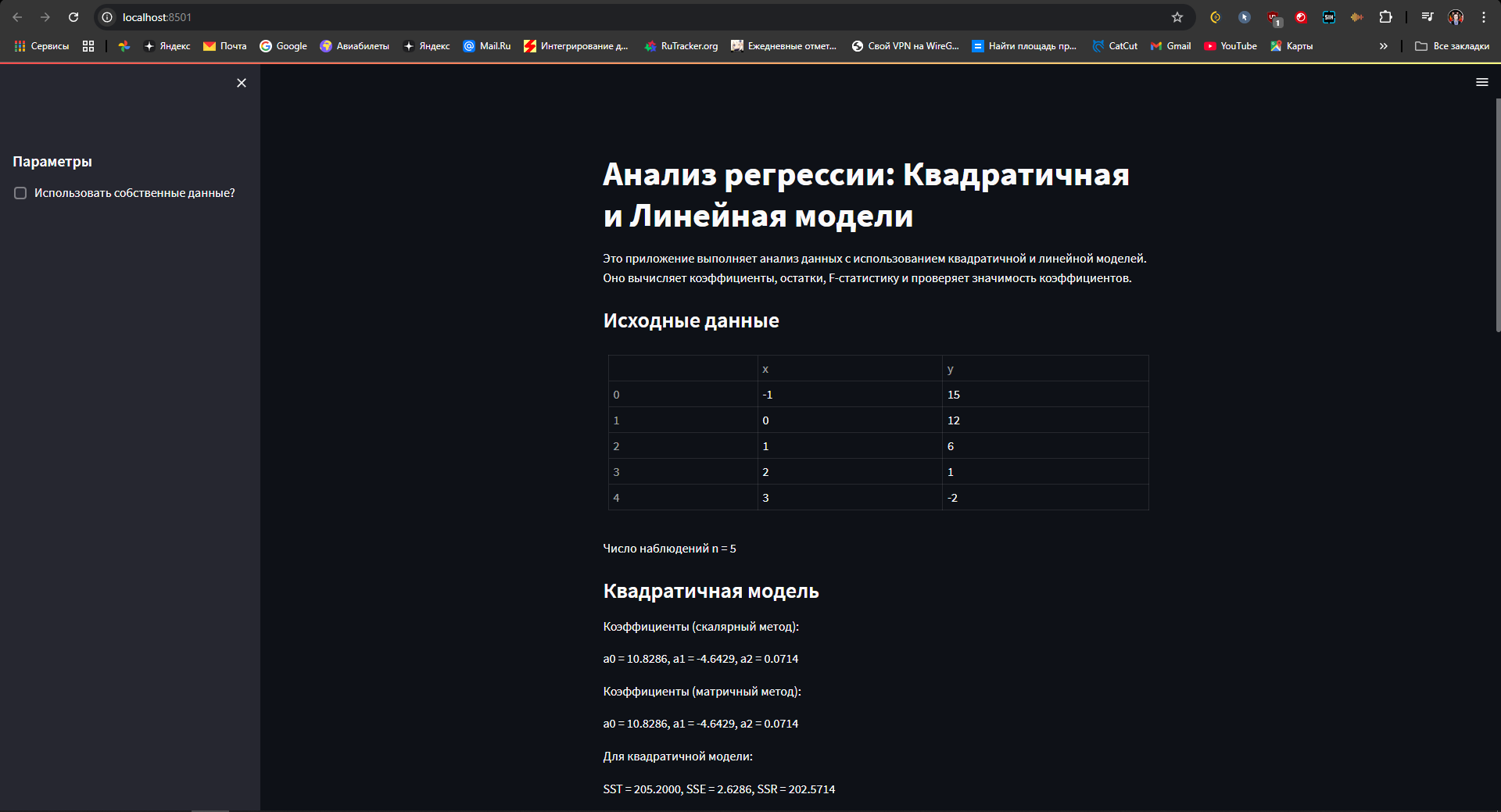
Коэффициенты линейной модели:

b0​=10.9000, b1​=−4.5000

Оценка адекватности линейной модели:

* Сумма квадратов ошибок (SSE): 2.7000
* Сумма квадратов регрессии (SSR): 202.5000
* F-статистика для линейной модели: F=225.0000, p-value: 0.0006

Модель является адекватной, так как p-value меньше уровня значимости α=0.01.

Результат работы программы:

**Анализ полученных результатов:**

* Квадратичная модель показала хорошие результаты, однако коэффициент при x^2 оказался незначимым, что подтверждает необходимость исключения этого члена из модели.
* Линейная модель с коэффициентами b0=10.9000 и b1=−4.5000 является адекватной, так как p-value для F-статистики составило 0.0006, что ниже порогового значения 0.01.

Линейная модель лучше описывает данные, чем квадратичная, что подтверждается результатами F-статистики и t-статистики.

**Вывод:**

При проведении анализа квадратичной модели y=a0+a1x+a2x^2 было установлено, что коэффициенты a0​ и a1​ статистически значимы, так как их p-value меньше 0.05, в то время как коэффициент a2​ оказался незначимым с p-value = 0.8373, что указывает на его отсутствие влияния на зависимую переменную y. Это позволило исключить член x^2 из модели и перейти к линейной модели y=b0+b1x. Линейная модель продемонстрировала высокую статистическую значимость, так как коэффициенты b0​ и b1 имеют p-value ниже 0.01, а F-статистика модели составила 225.0000, что подтверждает её адекватность.

Таким образом несмотря на то, что квадратичная модель в целом была статистически значимой, линейная модель оказалась более подходящей для данной задачи, так как её простота и значимость коэффициентов делают её лучшим выбором для дальнейшего анализа и предсказаний.

Листинг кода:

|  |
| --- |
| import streamlit as st  import numpy as np  import scipy.stats as st\_stats  import matplotlib.pyplot as plt  # Заголовок приложения  st.title("Анализ регрессии: Квадратичная и Линейная модели")  st.write("""  Это приложение выполняет анализ данных с использованием квадратичной и линейной моделей.  Оно вычисляет коэффициенты, остатки, F-статистику и проверяет значимость коэффициентов.  """)  # Исходные данные  x = np.array([-1, 0, 1, 2, 3])  y = np.array([15, 12, 6, 1, -2])  # Пользовательский ввод  st.sidebar.header("Параметры")  custom\_data = st.sidebar.checkbox("Использовать собственные данные?")  if custom\_data:      x\_input = st.sidebar.text\_input("Введите значения x (через запятую)", "-1,0,1,2,3")      y\_input = st.sidebar.text\_input("Введите значения y (через запятую)", "15,12,6,1,-2")      try:          x = np.array([float(i) for i in x\_input.split(",")])          y = np.array([float(i) for i in y\_input.split(",")])      except ValueError:          st.error("Ошибка ввода данных. Пожалуйста, введите числа через запятую.")          st.stop()  n = len(x)  # Отображение исходных данных  st.subheader("Исходные данные")  data = {"x": x, "y": y}  st.table(data)  st.write(f"Число наблюдений n = {n}")  # ===============================  # 1. Квадратичная модель: y = a0 + a1\*x + a2\*x^2  # ===============================  st.subheader("Квадратичная модель")  X\_quad = np.column\_stack((np.ones(n), x, x\*\*2))  # Скалярный метод  sum\_x = np.sum(x)  sum\_x2 = np.sum(x\*\*2)  sum\_x3 = np.sum(x\*\*3)  sum\_x4 = np.sum(x\*\*4)  sum\_y = np.sum(y)  sum\_xy = np.sum(x \* y)  sum\_x2y = np.sum(x\*\*2 \* y)  A = np.array([[n, sum\_x, sum\_x2],                [sum\_x, sum\_x2, sum\_x3],                [sum\_x2, sum\_x3, sum\_x4]])  b\_vec = np.array([sum\_y, sum\_xy, sum\_x2y])  beta\_scalar = np.linalg.solve(A, b\_vec)  st.write("Коэффициенты (скалярный метод):")  st.write(f"a0 = {beta\_scalar[0]:.4f}, a1 = {beta\_scalar[1]:.4f}, a2 = {beta\_scalar[2]:.4f}")  # Матричный метод  beta\_matrix = np.linalg.inv(X\_quad.T @ X\_quad) @ (X\_quad.T @ y)  st.write("Коэффициенты (матричный метод):")  st.write(f"a0 = {beta\_matrix[0]:.4f}, a1 = {beta\_matrix[1]:.4f}, a2 = {beta\_matrix[2]:.4f}")  # Предсказанные значения и остатки  y\_pred\_quad = X\_quad @ beta\_matrix  residuals\_quad = y - y\_pred\_quad  y\_bar = np.mean(y)  SST = np.sum((y - y\_bar)\*\*2)  SSE = np.sum(residuals\_quad\*\*2)  SSR = SST - SSE  st.write("Для квадратичной модели:")  st.write(f"SST = {SST:.4f}, SSE = {SSE:.4f}, SSR = {SSR:.4f}")  # F-статистика  p\_quad = 3  df\_reg = p\_quad - 1  df\_res = n - p\_quad  MSR = SSR / df\_reg  MSE = SSE / df\_res  F\_stat = MSR / MSE  p\_value\_F = 1 - st\_stats.f.cdf(F\_stat, df\_reg, df\_res)  st.write(f"F-статистика = {F\_stat:.4f}, p-value = {p\_value\_F:.4f}")  # График квадратичной модели  plt.figure(figsize=(8, 5))  plt.scatter(x, y, color="blue", label="Исходные данные")  plt.plot(x, y\_pred\_quad, color="red", label="Квадратичная модель")  plt.xlabel("x")  plt.ylabel("y")  plt.legend()  st.pyplot(plt)  # Проверка значимости коэффициентов  XtX\_inv = np.linalg.inv(X\_quad.T @ X\_quad)  var\_beta = MSE \* XtX\_inv  se\_beta = np.sqrt(np.diag(var\_beta))  t\_stats = beta\_matrix / se\_beta  p\_values\_t = 2 \* (1 - st\_stats.t.cdf(np.abs(t\_stats), df\_res))  st.write("Коэффициенты, стандартные ошибки, t-статистика и p-value (квадратичная модель):")  for i, (coef, se, t\_val, p\_val) in enumerate(zip(beta\_matrix, se\_beta, t\_stats, p\_values\_t)):      st.write(f"a{i} = {coef:.4f}, se = {se:.4f}, t = {t\_val:.4f}, p = {p\_val:.4f}")  alpha = 0.01  t\_crit = st\_stats.t.ppf(1 - alpha/2, df\_res)  st.write(f"Критическое значение t при α = {alpha:.2f} и df = {df\_res}: {t\_crit:.4f}")  # Переоценка модели, если a2 незначим  if np.abs(t\_stats[2]) < t\_crit:      st.write("Коэффициент a2 (при x^2) незначим. Переоцениваем модель, исключая член x^2 (линейная регрессия).")        # ===============================      # 2. Линейная модель: y = b0 + b1\*x      # ===============================      X\_lin = np.column\_stack((np.ones(n), x))      beta\_lin = np.linalg.inv(X\_lin.T @ X\_lin) @ (X\_lin.T @ y)      st.write("Коэффициенты линейной модели:")      st.write(f"b0 = {beta\_lin[0]:.4f}, b1 = {beta\_lin[1]:.4f}")        # Предсказания и остатки для линейной модели      y\_pred\_lin = X\_lin @ beta\_lin      residuals\_lin = y - y\_pred\_lin      SSE\_lin = np.sum(residuals\_lin\*\*2)      SSR\_lin = SST - SSE\_lin        # F-статистика для линейной модели      p\_lin = 2      df\_reg\_lin = p\_lin - 1      df\_res\_lin = n - p\_lin      MSR\_lin = SSR\_lin / df\_reg\_lin      MSE\_lin = SSE\_lin / df\_res\_lin      F\_stat\_lin = MSR\_lin / MSE\_lin      p\_value\_F\_lin = 1 - st\_stats.f.cdf(F\_stat\_lin, df\_reg\_lin, df\_res\_lin)        st.write("Для линейной модели:")      st.write(f"SSE = {SSE\_lin:.4f}, SSR = {SSR\_lin:.4f}")      st.write(f"F-статистика = {F\_stat\_lin:.4f}, p-value = {p\_value\_F\_lin:.4f}")      if p\_value\_F\_lin < alpha:          st.write("Линейная модель является адекватной (p-value < α).")      else:          st.write("Линейная модель не является адекватной (p-value ≥ α).")        # График линейной модели      plt.figure(figsize=(8, 5))      plt.scatter(x, y, color="blue", label="Исходные данные")      plt.plot(x, y\_pred\_lin, color="green", label="Линейная модель")      plt.xlabel("x")      plt.ylabel("y")      plt.legend()      st.pyplot(plt)  else:      st.write("Все коэффициенты квадратичной модели значимы, повторная оценка не требуется.") |