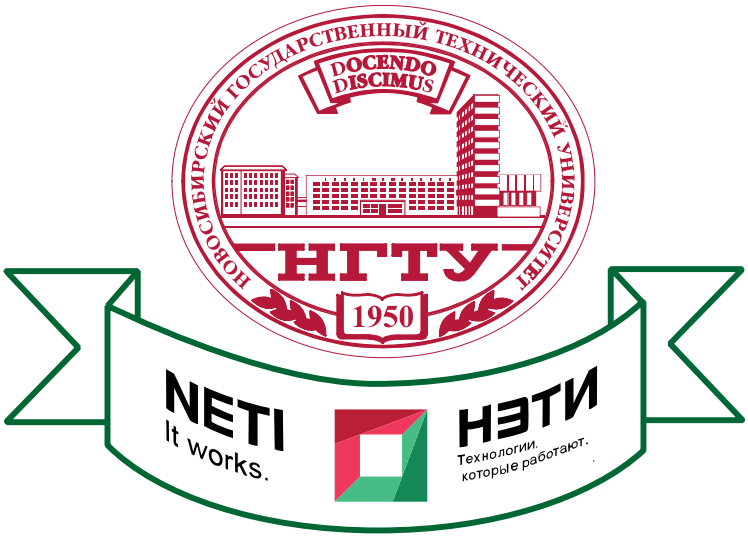
Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

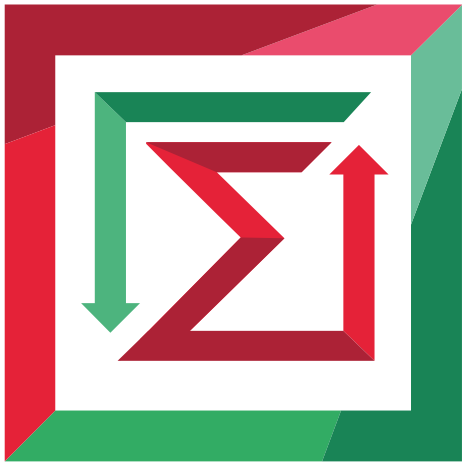
«Новосибирский государственный технический университет»



Кафедра теоретической и прикладной информатики

Лабораторная работа № 5

по дисциплине «Статистические методы анализа данных»



Факультет:	ПМИ
Группа:	ПМИ-02
Вариант:	6
Студент:	Сидоров Даниил,
	Дюков Богдан
Преподаватель:	Попов Александр Александрович.

Новосибирск

2026

1. Постановка задачи

1. В соответствии с вариантом задания сгенерировать экспериментальные данные, в которых в явном виде присутствует эффект мультиколлинеарности.

2. Рассчитать ряд показателей, характеризующих эффект мультиколлинеарности. Определить факторы, ответственные за возникновение эффекта мультиколлинеарности.

3. Построить ридж-оценки параметров при различных значениях параметра регуляризации. Выбрать оптимальное значение параметра регуляризации. Построить графики изменения квадрата евклидовой нормы оценок параметров и остаточной суммы квадратов от параметра регуляризации.

4. Провести оценивание модели регрессии по методу главных компонентов. Перейти к описанию в исходном пространстве факторов. Сравнить решение с ридж-оцениванием по смещению оценок и точности предсказания отклика.

   Регрессия на 6 факторах. Эффект мультиколлинеарности создают 2 фактора. Имеется разброс в масштабах факторов.

2. Ход работы

Генерация экспериментальных данных

$\eta(\mathrm{x}, \theta) = \theta^{\mathrm{T}} f(x_1, x_2, x_3, x_4, x_5, x_6)$;

$f(x_1, x_2, x_3, x_4, x_5, x_6) = (1, x_1, x_2, x_3, x_4, x_5, x_6)$;

$\theta = (1, 1, 1{,}1, 1, 1, 1)^{\mathrm{T}}$;

$x_1 \in [-1, 1], x_2 \in [-7, 7], x_3 \in [-3, 3],$

$x_4 \in [-2, 2], x_5 \in [-10, 10], x_6 = \frac{x_5}{3} + N(0{,}0.01)$;

$\varepsilon_{\mathrm{i}} \sim N(0, {\sigma_i}^2)$;

${\sigma_i}^2 = 5\ \%$ от мощности сигнала;

Количество экспериментов $n = 1200$;

Показатели, характеризующие мультиколлинеарность

1. Определитель информационной матрицы.

   $$|X^T X| = \prod_{i=1}^{m} \lambda_i = 2.58853e + 20;$$

   Определитель информационной матрицы может быть большим числом, даже если присутствует мультиколлинеарность. Это связано с тем, что определитель матрицы зависит не только от наличия линейной зависимости между столбцами, но и от масштаба данных.

2. Минимальное собственное число информационной матрицы.

   $$\lambda_{min}(X^T X) = 1.01799e - 01;$$

   Число близко к нулю, это может указывать на наличие мультиколлинеарности.

   Найдем также максимальное собственное число информационной матрицы:

   $$\lambda_{max}(X^T X) = 4.49935e + 04;$$

3. Мера обусловленности матрицы по Нейману-Голдстрейну.

   $$\frac{\lambda_{max}(X^T X)}{\lambda_{min}(X^T X)} = 4.41982e + 05;$$

   Это число значительно больше 1, что может указывать на наличие мультиколлинеарности.

4. Максимальная парная сопряженность. Построим матрицу сопряженности:

   $$R = \begin{bmatrix} 1 & \dots & r_{1m} \\ r_{21} & \dots & r_{2m} \\ r_{m1} & \dots & 1 \end{bmatrix},$$

   где $r_{ij} = cos(x_i, x_j)$;

Косинус угла между векторами можно интерпретировать как косинусное сходство между векторами. Построенная матрица сопряженности:

1.0	0.03236739	0.03910345	-0.02993151	-0.01425576	-0.01426835
0.03236739	1.0	-0.02614126	-0.00416902	-0.02918098	-0.02920188
0.03910345	-0.02614126	1.0	-0.00364204	0.02464101	0.0247731
-0.02993151	-0.00416902	-0.00364204	1.0	0.00759167	0.00758656
-0.01425576	-0.02918098	0.02464101	0.00759167	1.0	0.99998741
-0.01426835	-0.02920188	0.0247731	0.00758656	0.99998741	1.0

В качестве показателя мультиколлинеарности выступает величина:

$$\max_{i,j}|r_{ij}| = 0.999987, i \neq j;$$

5. Максимальная сопряженность. В качество меры мультиколлинеарности можно взять $\max_i|R_i|$. Вычислить R_i можно следующим способом:

$$R^2{}_t = 1 - \frac{1}{R_{ii}^{-1}}, i = 1..m,$$

где $R_{ii}^{-1} -$ диагональный элемент матрицы, обратный к сопряженной.

```
Вектор R_2:  [0.00373335, 0.00260749, 0.00358045, 0.00096474, 0.99997484, 0.99997484]
```

Максимальная сопряженность: 0.99997. За возникновение эффекта мультиколлинеарности отвечают факторы: x_5, x_6.

Построение ридж-оценок параметров

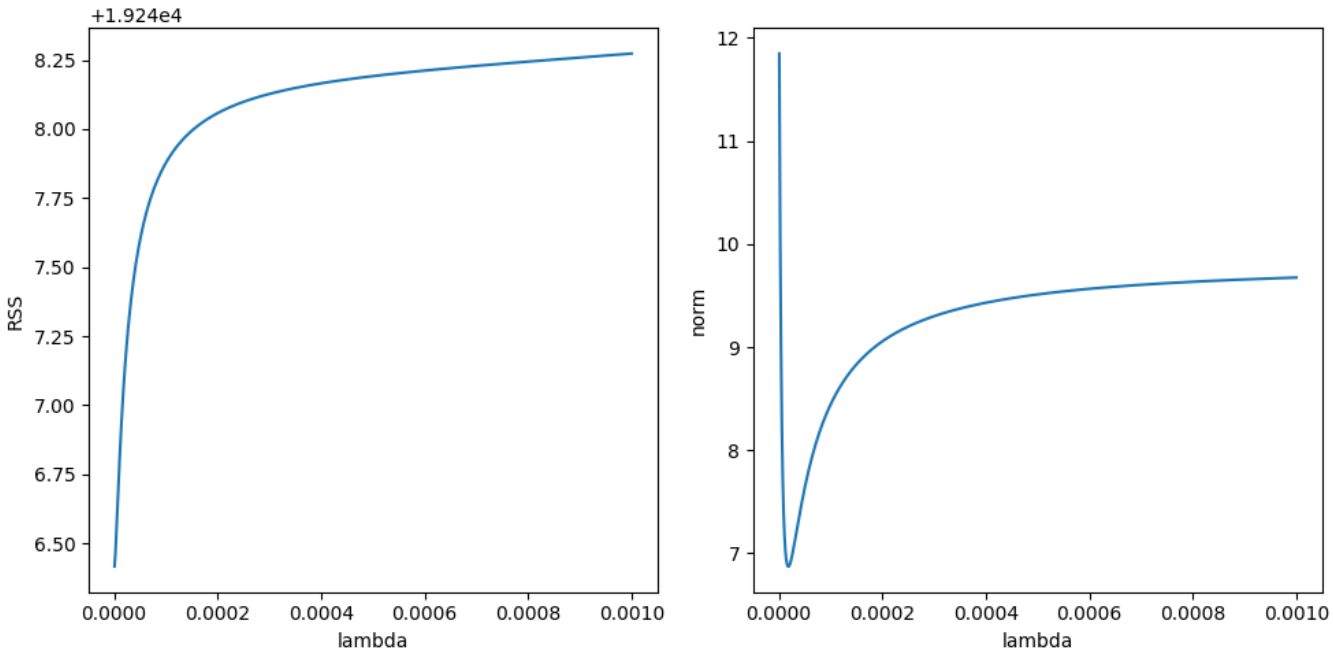
Для оценок параметров модели в условиях мультиколлинеарности используется следующая формула:

$$\overline{\theta} = (X^T X + \Lambda)^{-1} X^T y,$$

где матрицу Λ можно задать как диагональную матрицу с элементами

$\Lambda_{ii} = \lambda(X^T X)_{ii}, \lambda \geq 0$, где λ - параметр регуляризации.

Параметр регуляризации выбираем как компромисс между неизбежным увеличением остаточной суммы квадратов и желаемым уменьшением евклидовой нормы оценок параметров.



Исходя из графиков имеем оптимальный $\lambda = 3.6e - 05$ и соответствующую ему ридж-оценку:

$\bar{\theta} = (1.06503, 1.0865, 1.04844, 1.04739, 0.81965, 1.03506, 1.00323)^{\mathrm{T}}$;
$\mathrm{RSS} = 1.92474\mathrm{e} + 04$;
$\text{Евклидова норма оценок параметров} = \ 2.69457.$

Метод главных компонент

Перейдем к центрированным переменным:

$x_{it}^{*} = x_{it} - \overline{x_i}$,

где $\overline{x_i} = \sum_{k=1}^{n} \frac{x_{kj}}{n}$;

$y_t{}^{*} = y_t - \overline{y}$;

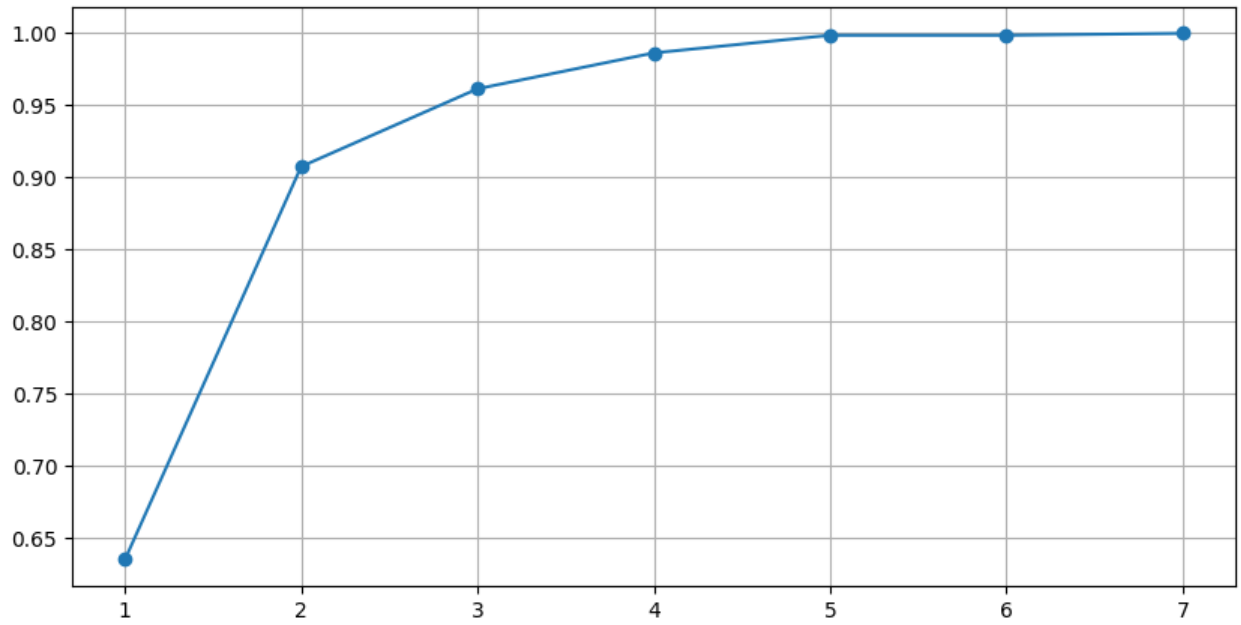
Построим матрицу ковариаций $X^{*T}X^{*}$, найдем для неё собственные значения и собственные вектора. Выразим главные компоненты в матричном виде:

$Z = X^{*}V$,

где V –матрица, состоящая из всех собственных векторов матрицы $X^{*T}X^{*}$.

Исключим из матриц Z и V столбцы, соответствующие собственным значениям с незначительными вкладами. Для этого построим график зависимости критерия информативности от номера собственного значения.

Критерий информативности определяется как отношение кумулятивной суммы собственных значений к общей сумме собственных значений.



Собственные значения ковариационной матрицы:

0	36871.311838023255
1	15819.886481170113
2	3128.75752240784
3	1444.895622187281
4	710.5175139972913
5	-1.094629583416711e-12
6	77.65787543480947

Видим, что незначительный вклад дает собственное значения λ_6, соответствующие ему столбцы и исключаем. Обозначим полученные матрицы как Z_{new} и V_{new}. Найдем вектор $\hat{b} = ({Z_{new}}^T Z_{new})^{-1} {Z_{new}}^T y^*$, с помощью него найдем искомые оценки параметров: $\hat{\theta} = V_{new}\hat{b}$.

$\hat{\theta} = (0.42838, 1.08314, 1.04834, 1.04853, 0.8187, 4.26787, -8.69496)^{\mathrm{T}}$
$\mathrm{RSS} = 1.97357\mathrm{e} + 04;$
$\text{Евклидова норма оценок параметров} = 9.90163.$

Метод главных компонент оказался хуже метода ридж-оценок как по смещению оценок, так и по точности предсказания отклика.

3. Код программы

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(10)

# Генерация комбинаций факторов
def generate_random(n):
    x1_list = [np.random.uniform(x1_interval[0], x1_interval[1]) for _ in range(n)]
    x2_list = [np.random.uniform(x2_interval[0], x2_interval[1]) for _ in range(n)]
    x3_list = [np.random.uniform(x3_interval[0], x3_interval[1]) for _ in range(n)]
    x4_list = [np.random.uniform(x4_interval[0], x4_interval[1]) for _ in range(n)]
    x5_list = [np.random.uniform(x5_interval[0], x5_interval[1]) for _ in range(n)]
    x6_list = np.array(x5_list) / 3 + np.random.normal(0, 0.01, size=n)

    return map(np.array, [x1_list, x2_list, x3_list, x4_list, x5_list, x6_list])

# Получение дисперсии шума
def get_sigma_squared(u):
    # Вычисление мощности сигнала
    omega_squared = np.dot(u - np.mean(u), u - np.mean(u)) / (len(u) - 1)
    # Доля от мощности сигнала
    rho = 0.05
    # Вычисление дисперсии шума
    return rho * omega_squared

# Получение ошибки
def get_noise(u, sigma_squared):
    return np.random.normal(0, sigma_squared, len(u))

n=1200

# Определение параметров
theta = np.array([1, 1, 1, 1, 1, 1, 1])
x1_interval = (-1, 1)
x2_interval = (-7, 7)
x3_interval = (-3, 3)
x4_interval = (-2, 2)
```

```
x5_interval = (-10, 10)

#---------- Задание №1. Генерация данных ----------#
# Генерация комбинаций факторов
x1, x2, x3, x4, x5, x6 = generate_random(n)
# Вычисление истинного отклика без шума
u = theta[0] + theta[1]*x1 + theta[2]*x2 + theta[3]*x3 \
    + theta[4]*x4 + theta[5]*x5 + theta[6]*x6
sigma2 = get_sigma_squared(u)
e = get_noise(u, sigma2)
y = u + e
X = np.column_stack((np.ones(len(x1)), x1, x2, x3, x4, x5, x6))

#---------- Задание №2. Расчет показателей ----------#
# Вычисление информационной матрицы и её определителя
info_matrix = np.dot(X.T, X)
det_info_matrix = np.linalg.det(info_matrix)

print("Определитель информационной матрицы: ", det_info_matrix)

# Вычисление собственных чисел информационной матрицы и сохранение max и min собственного числа
eigvals = np.linalg.eigvals(info_matrix)
min_eigval = min(eigvals)
max_eigval = max(eigvals)

print("Минимальное собственное число информационной матрицы: {:.5e}".format(min_eigval))
print("Максимальное собственное число информационной матрицы: {:.5e}".format(max_eigval))

# Вычисление меры обусловленности матрицы по Нейману-Голдстейну
neuman_goldstein_measure = max_eigval / min_eigval

print("Мера обусловленности матрицы по Нейману-Голдстейну: {:.5e}".format(neuman_goldstein_measure))

# Инициализация матрицы сопряженности
R = np.zeros((X.shape[1] - 1, X.shape[1] - 1))

# Вычисление матрицы сопряженности без первого столбца (свободный член)
# Косинус угла между векторами можно интерпретировать как косинусное сходство между векторами
for i in range(1, X.shape[1]):
    for j in range(i, X.shape[1]):
        R[i - 1, j - 1] = (np.dot(X[:, i], X[:, j]) / (np.linalg.norm(X[:, i]) * np.linalg.norm(X[:,
j]))).round(8)
        R[j - 1, i - 1] = R[i - 1, j - 1]  # Отражение верхнего треугольника в нижний
```

```
print("Матрица сопряженности")
print(R)

np.fill_diagonal(R, 0)

# Вычисление максимальной парной сопряженности
max_pairwise_conjugacy = np.max(np.abs(R))

print("Максимальная парная сопряженность: ", max_pairwise_conjugacy)

np.fill_diagonal(R, 1)

# Вычисление обратной матрицы к матрице сопряженности
R_inv = np.linalg.inv(R)

# Вычисление R_2 для каждого i и нахождение максимального значения
R_2 = [(1 - 1 / R_inv[i, i]).round(8) for i in range(0, R.shape[0])]
max_conjugacy = max(R_2)

print("Вектор R_2: ", R_2)
print("Максимальная сопряженность: ", max_conjugacy)

#---------- Задание №3. Ридж-оценки ----------#
# Инициализация списков для хранения евклидовых норм и остаточных сумм квадратов для графиков
euclidean_norms = []
rss_values = []
# Инициализация списка для хранения значений параметра регуляризации
lambdas = np.linspace(1e-06, 0.001, 600)

# Вычисление ридж-оценок для каждого значения параметра регуляризации
for lambda_ in lambdas:
    # Создание диагональной матрицы Lambda
    Lambda = np.diag(lambda_ * np.diag(info_matrix))
    # Вычисление ридж-оценки
    theta_ridge = np.linalg.inv(X.T @ X + Lambda) @ X.T @ y
    # Вычисление евклидовой нормы между оценкой и искомыми параметрами
    euclidean_norms.append(theta_ridge.T @ theta_ridge)
    # Вычисление остаточной суммы квадратов
    rss_values.append((y - X @ theta_ridge).T @ (y - X @ theta_ridge))

# Построение графиков параметра регуляризации от RSS и евклидовой нормы
```

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)  # первый график
plt.plot(lambdas, rss_values)
plt.xlabel('lambda')
plt.ylabel('RSS')
plt.subplot(1, 2, 2)  # второй график
plt.plot(lambdas, euclidean_norms)
plt.xlabel('lambda')
plt.ylabel('norm')
# Автоматическое выравнивание элементов на рисунке
plt.tight_layout()
plt.show()

# Создание диагональной матрицы при оптимальном lambda = 3.6e-05
optimal_diagonal_lambda = np.diag(3.6e-05 * np.diag(info_matrix))
# Получаем ридж-оценку, соответствующую оптимальному значению lambda
optimal_theta_ridge = np.linalg.inv(X.T @ X + optimal_diagonal_lambda) @ X.T @ y
# Вычисление евклидовой нормы между оценкой и искомыми параметрами
optimal_norm = np.sqrt(optimal_theta_ridge.T @ optimal_theta_ridge)
# Вычисление остаточной суммы квадратов
optimal_rss = (y - X @ optimal_theta_ridge).T @ (y - X @ optimal_theta_ridge)

print("Ридж-оценка при lambda примерно равном 3.6e-05: ", optimal_theta_ridge)
print("RSS и Евклидова норма оценок параметров при lambda примерно равном 3.6e-05:
{:.5e}".format(optimal_rss), optimal_norm)

#---------- Задание №4. Метод главных компонент ----------#
# Переход к центрированным оценкам
y_centered = y - np.mean(y)
X_centered = X - np.mean(X, axis=1, keepdims=True)
# Вычисление матрицы ковариации
cov_matrix = X_centered.T @ X_centered
# Вычисление собственных значений и собственных векторов
eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)
print("Собственные числа ковариационной матрицы: ", eigen_values)
# Матрица V состоит из всех собственных векторов
V = eigen_vectors
# Главные компоненты Z получаются умножением X* на V
Z = X_centered @ V

# Построение графика зависимости критерия информативности от номера собственного значения
# Вычисляем кумулятивную сумму собственных значений
cumulative_eigen_values = np.cumsum(eigen_values)
```

```
# Вычисление общей суммы собственных значений
total_eigen_values = np.sum(eigen_values)
# Вычисление критерия информативности
ratios = cumulative_eigen_values / total_eigen_values

plt.figure(figsize=(10, 5))
plt.plot(range(1, len(eigen_values) + 1), ratios, marker='o')
plt.grid(True)
plt.show()

# Исключение из Z и V столбцов соответствующих собственным значениям с незначительным вкладом
Z_new = np.delete(Z, -2, axis=1)
V_new = np.delete(V, -2, axis=1)

# Посчет b=(Z_R^T * Z_R)^(-1) * Z_R^(T) * y^*
b = np.linalg.inv(Z_new.T @ Z_new) @ Z_new.T @ y_centered
# Выражение оценок V_R * b
theta_main = V_new @ b
# Вычисление остаточной суммы квадратов
rss_main = (y - X @ theta_main).T @ (y - X @ theta_main)
# Вычисление евклидовой нормы между оценкой и искомыми параметрами
norm_main = np.sqrt(theta_main.T @ theta_main)

print("Оценки параметров по методу главных компонентов: ", theta_main)
print("RSS и Евклидова норма оценок параметров: {:.5e}".format(rss_main), norm_main))
```