Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Системы искусственного интеллекта»

Лабораторная работа №3

«Линейная регрессия»

Выполнил:

студент группы ИВТАПбд-41

Сокольский Р.С

Проверил:

преподаватель кафедры «ВТ»

Святов К.В.

Ульяновск, 2025

Оглавление

[1. Постановка задачи 3](#_Toc210251350)

[2. Реализация 4](#_Toc210251351)

[3. Тестирование 6](#_Toc210251352)

[4. Выводы 8](#_Toc210251353)

# **1. Постановка задачи**

1. Написать программу, которая разделяет исходную выборку на обучающую и тестовую (training set, test set). Использовать стандартные функции (train\_test\_split и др. нельзя).
2. С использованием библиотеки scikit-learn обучить модель линейной регрессии по обучающей выборке
3. Проверить точность модели по тестовой выборке
4. Построить модель с использованием полиномиальной функции. Построить графики зависимости точности на обучающей и тестовой выборке от степени полиномиальной функции.
5. Построить модель с использованием регуляризации. На основе экспериментов подобрать параметры для регуляризации. Построить графики зависимости точности модели на обучающей и тестовой выборках от коэффициента регуляризации.

Вариант 22: Combined Cycle Power Plant

# **2****. Реализация**

В начале происходит загрузка данных из Excel-файла, а также вывод информации о размере данных, первых пяти строк данных, о самих данных и статистическое описание:

**Листинг 1.** Загрузка данных и вывод информации

|  |
| --- |
| **file\_path = 'Folds5x2\_pp.xlsx'**  **all\_sheets = pd.read\_excel(file\_path, sheet\_name=None)**  **data = pd.concat(all\_sheets.values(), ignore\_index=True)**  **print("Размер данных:", data.shape)**  **print("\nПервые 5 строк данных:")**  **print(data.head())**  **print("\nИнформация о данных:")**  **print(data.info())**  **print("\nСтатистическое описание:")**  **print(data.describe())** |

Дальше идёт разделение на признаки и целевую переменную:

**Листинг 2.** Разделение на признаки и целевую переменную

|  |  |
| --- | --- |
| |  | | --- | | **X = data.iloc[:, :-1].values**  **y = data.iloc[:, -1].values**  **print(f"Форма массива признаков (X): {X.shape}")**  **print(f"Форма массива целевой переменной (y): {y.shape}")**  **print(f"\nНазвания признаков: {data.columns[:-1].tolist()}")**  **print(f"Целевая переменная: {data.columns[-1]}")** |   Вывод:  Форма массива признаков (X): (47840, 4)  Форма массива целевой переменной (y): (47840,)  Названия признаков: ['AT', 'V', 'AP', 'RH']  Целевая переменная: PE  После следует ручное разделение на обучающую и тестовую выборки (следуя по заданию): |

**Листинг 3.** Реализация ручного разделения данных

|  |  |
| --- | --- |
| |  | | --- | | **test\_size = 0.2**  **indices = np.arange(len(X))**  **np.random.seed(42)**  **np.random.shuffle(indices)**  **split\_index = int(len(X) \* (1 - test\_size))**  **X\_shuffled = X[indices]**  **y\_shuffled = y[indices]**  **X\_train = X\_shuffled[:split\_index]**  **X\_test = X\_shuffled[split\_index:]**  **y\_train = y\_shuffled[:split\_index]**  **y\_test = y\_shuffled[split\_index:]**  **print("Разделение завершено:")**  **print(f"Обучающая выборка: {X\_train.shape[0]} примеров")**  **print(f"Тестовая выборка: {X\_test.shape[0]} примеров")**  **print(f"\nРазмер X\_train: {X\_train.shape}")**  **print(f"Размер X\_test: {X\_test.shape}")** | |

Вывод:

Разделение завершено:

Обучающая выборка: 38272 примеров

Тестовая выборка: 9568 примеров

Размер X\_train: (38272, 4)

Размер X\_test: (9568, 4)

Потом идёт обучение линейной регрессии:

**Листинг 4.** Реализация обучения и оценки линейной регрессии

|  |  |
| --- | --- |
| |  | | --- | | **print("=== Линейная регрессия ===")**  **lr\_model = LinearRegression()**  **lr\_model.fit(X\_train, y\_train)**  **print("Коэффициенты модели:")**  **for i, col in enumerate(data.columns[:-1]):**  **print(f" {col}: {lr\_model.coef\_[i]:.4f}")**  **print(f"Свободный член (intercept): {lr\_model.intercept\_:.4f}")**  **y\_pred\_train = lr\_model.predict(X\_train)**  **y\_pred\_test = lr\_model.predict(X\_test)**  **print("\nКачество модели на обучающей выборке:")**  **print(f" R^2: {r2\_score(y\_train, y\_pred\_train):.4f}")**  **print(f" MSE: {mean\_squared\_error(y\_train, y\_pred\_train):.4f}")**  **print("Качество модели на тестовой выборке:")**  **print(f" R^2: {r2\_score(y\_test, y\_pred\_test):.4f}")**  **print(f" MSE: {mean\_squared\_error(y\_test, y\_pred\_test):.4f}")** | |

Вывод:

=== Линейная регрессия ===

Коэффициенты модели:

AT: -1.9728

V: -0.2352

AP: 0.0644

RH: -0.1577

Свободный член (intercept): 452.1806

Качество модели на обучающей выборке:

R^2: 0.9287

MSE: 20.7535

Качество модели на тестовой выборке:

R^2: 0.9287

MSE: 20.8256

Полиномиальная регрессия:

**Листинг 5.** Исследование полиномиальной регрессии для отдельных признаков

|  |  |
| --- | --- |
| |  | | --- | | **print("=== Полиномиальная регрессия для всех признаков ===")**  **feature\_names = ['Температура (AT)', 'Давление (V)', 'Атмосферное давление (AP)', 'Влажность (RH)']**  **feature\_columns = [0, 1, 2, 3]**  **degrees = [1, 4, 15]**  **plt.figure(figsize=(16, 12))**  **results = {}**  **for feature\_idx, feature\_name in zip(feature\_columns, feature\_names):**  **X\_train\_single = X\_train[:, feature\_idx].reshape(-1, 1)**  **X\_test\_single = X\_test[:, feature\_idx].reshape(-1, 1)**    **sort\_idx = np.argsort(X\_train\_single.ravel())**  **X\_train\_sorted = X\_train\_single[sort\_idx]**  **y\_train\_sorted = y\_train[sort\_idx]**    **plt.subplot(2, 2, feature\_idx + 1)**    **feature\_results = []**    **for degree in degrees:**  **poly = PolynomialFeatures(degree=degree, include\_bias=False)**  **X\_train\_poly = poly.fit\_transform(X\_train\_single)**  **X\_test\_poly = poly.transform(X\_test\_single)**    **poly\_model = LinearRegression()**  **poly\_model.fit(X\_train\_poly, y\_train)**    **y\_pred\_train = poly\_model.predict(X\_train\_poly)**  **y\_pred\_test = poly\_model.predict(X\_test\_poly)**    **mse\_train = mean\_squared\_error(y\_train, y\_pred\_train)**  **mse\_test = mean\_squared\_error(y\_test, y\_pred\_test)**  **feature\_results.append((mse\_train, mse\_test))**    **X\_plot = np.linspace(X\_train\_single.min(), X\_train\_single.max(), 300).reshape(-1, 1)**  **X\_plot\_poly = poly.transform(X\_plot)**  **y\_plot = poly\_model.predict(X\_plot\_poly)**    **plt.plot(X\_plot, y\_plot, linewidth=5, label=f'Степень {degree}')**    **results[feature\_name] = feature\_results**    **plt.scatter(X\_train\_sorted, y\_train\_sorted, color='navy', s=5, alpha=0.3, label="Данные")**    **plt.xlabel(feature\_name)**  **plt.ylabel("Выработка энергии (PE)")**  **plt.title(f"Зависимость энергии от {feature\_name.split(' (')[0]}")**  **plt.legend()**  **plt.grid(True, alpha=0.3)**  **plt.tight\_layout()**  **plt.show()** | |

Также выводится таблица результатов регрессии:

**Листинг 6.** Сводная таблица результатов полиномиальной регрессии

|  |  |
| --- | --- |
| |  | | --- | | **print("\n" + "="\*80)**  **print("СРАВНЕНИЕ MSE ДЛЯ ВСЕХ ПРИЗНАКОВ")**  **print("="\*80)**  **print(f"{'Признак':<25} {'Степень':<8} {'MSE train':<12} {'MSE test':<12} {'Разница':<10}")**  **print("-"\*80)**  **for feature\_name in feature\_names:**  **feature\_results = results[feature\_name]**  **for i, degree in enumerate(degrees):**  **mse\_train, mse\_test = feature\_results[i]**  **difference = mse\_test - mse\_train**    **print(f"{feature\_name if i == 0 else '':<25} {degree:<8} {mse\_train:<12.4f} {mse\_test:<12.4f} {difference:>+8.4f}")**  **print("-" \* 80)** | |

В конечном итоге, производится анализ зависимости точности от степени полинома:

**Листинг 7.** Исследование зависимости точности от сложности модели

|  |  |
| --- | --- |
| |  | | --- | | **print("\n=== Зависимость точности от степени полинома ===")**  **degrees\_range = range(1, 11)**  **train\_scores = []**  **test\_scores = []**  **scaler = StandardScaler()**  **X\_train\_scaled = scaler.fit\_transform(X\_train)**  **X\_test\_scaled = scaler.transform(X\_test)**  **for degree in degrees\_range:**  **poly = PolynomialFeatures(degree=degree, include\_bias=False)**  **X\_train\_poly = poly.fit\_transform(X\_train\_scaled)**  **X\_test\_poly = poly.transform(X\_test\_scaled)**    **poly\_model = LinearRegression()**  **poly\_model.fit(X\_train\_poly, y\_train)**    **y\_pred\_train = poly\_model.predict(X\_train\_poly)**  **y\_pred\_test = poly\_model.predict(X\_test\_poly)**    **train\_scores.append(r2\_score(y\_train, y\_pred\_train))**  **test\_scores.append(r2\_score(y\_test, y\_pred\_test))**  **plt.figure(figsize=(10, 6))**  **plt.plot(degrees\_range, train\_scores, 'bo-', label='Обучающая выборка', linewidth=2, markersize=6)**  **plt.plot(degrees\_range, test\_scores, 'ro-', label='Тестовая выборка', linewidth=2, markersize=6)**  **plt.xlabel('Степень полинома')**  **plt.ylabel('R-квадрат')**  **plt.title('Зависимость точности от степени полиномиальной функции')**  **plt.legend()**  **plt.grid(True, alpha=0.3)**  **plt.xticks(degrees\_range)**  **plt.show()**  **print("Анализ переобучения:")**  **for i, degree in enumerate(degrees\_range):**  **print(f"Степень {degree}: Train R^2 = {train\_scores[i]:.4f}, Test R^2 = {test\_scores[i]:.4f}")** | |

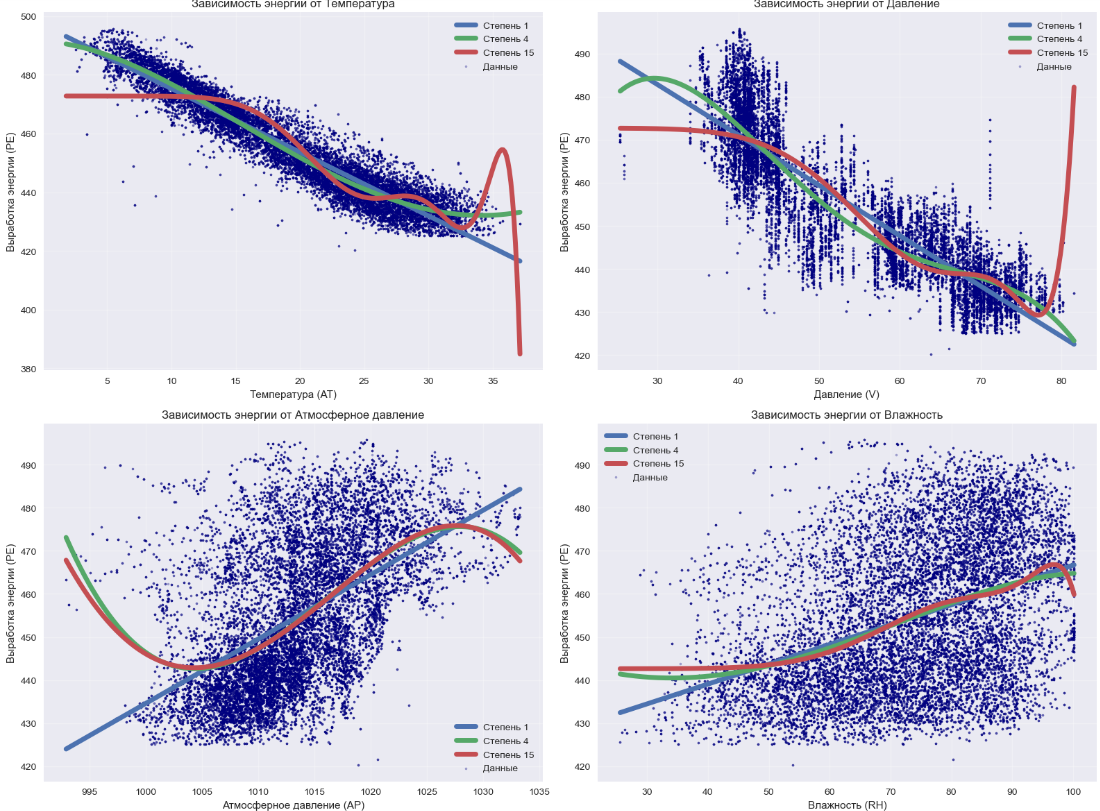
И наконец, исследование регуляризации:

**Листинг 9.** Исследование Ridge и Lasso регуляризации

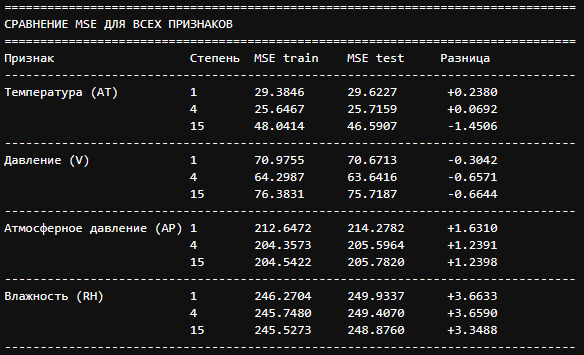
|  |  |
| --- | --- |
| |  | | --- | | **print("=== Ridge и Lasso регуляризация ===")**  **alphas = np.logspace(-3, 2, 50)**  **ridge\_train\_scores, ridge\_test\_scores = [], []**  **lasso\_train\_scores, lasso\_test\_scores = [], []**  **print("Подбор параметров регуляризации...")**  **for alpha in alphas:**  **ridge\_model = Ridge(alpha=alpha, random\_state=42)**  **ridge\_model.fit(X\_train\_scaled, y\_train)**  **y\_pred\_train\_ridge = ridge\_model.predict(X\_train\_scaled)**  **y\_pred\_test\_ridge = ridge\_model.predict(X\_test\_scaled)**  **ridge\_train\_scores.append(r2\_score(y\_train, y\_pred\_train\_ridge))**  **ridge\_test\_scores.append(r2\_score(y\_test, y\_pred\_test\_ridge))**    **lasso\_model = Lasso(alpha=alpha, random\_state=42, max\_iter=5000)**  **lasso\_model.fit(X\_train\_scaled, y\_train)**  **y\_pred\_train\_lasso = lasso\_model.predict(X\_train\_scaled)**  **y\_pred\_test\_lasso = lasso\_model.predict(X\_test\_scaled)**  **lasso\_train\_scores.append(r2\_score(y\_train, y\_pred\_train\_lasso))**  **lasso\_test\_scores.append(r2\_score(y\_test, y\_pred\_test\_lasso))**  **plt.figure(figsize=(12, 5))**  **plt.subplot(1, 2, 1)**  **plt.semilogx(alphas, ridge\_train\_scores, 'b-', label='Обучающая выборка', linewidth=2)**  **plt.semilogx(alphas, ridge\_test\_scores, 'r-', label='Тестовая выборка', linewidth=2)**  **plt.xlabel('Коэффициент регуляризации (alpha)')**  **plt.ylabel('R^2')**  **plt.title('Ridge-регуляризация')**  **plt.legend()**  **plt.grid(True, alpha=0.3)**  **plt.subplot(1, 2, 2)**  **plt.semilogx(alphas, lasso\_train\_scores, 'b-', label='Обучающая выборка', linewidth=2)**  **plt.semilogx(alphas, lasso\_test\_scores, 'r-', label='Тестовая выборка', linewidth=2)**  **plt.xlabel('Коэффициент регуляризации (alpha)')**  **plt.ylabel('R^2')**  **plt.title('Lasso-регуляризация')**  **plt.legend()**  **plt.grid(True, alpha=0.3)**  **plt.tight\_layout()**  **plt.show()**  **best\_alpha\_ridge = alphas[np.argmax(ridge\_test\_scores)]**  **best\_alpha\_lasso = alphas[np.argmax(lasso\_test\_scores)]**  **print("Результаты подбора параметров:")**  **print(f"Ridge - лучший alpha: {best\_alpha\_ridge:.4f}, R^2: {np.max(ridge\_test\_scores):.4f}")**  **print(f"Lasso - лучший alpha: {best\_alpha\_lasso:.4f}, R^2: {np.max(lasso\_test\_scores):.4f}")** | |

# **3. Тестирование**

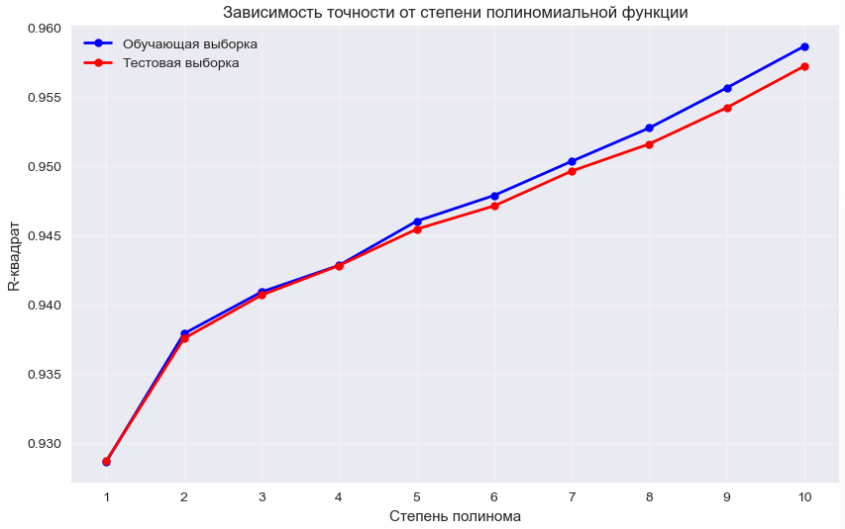
1. Полиномиальная регрессия для всех признаков.



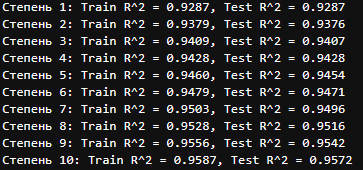
2. Сравнение MSE (среднеквадратической ошибки) для всех признаков.



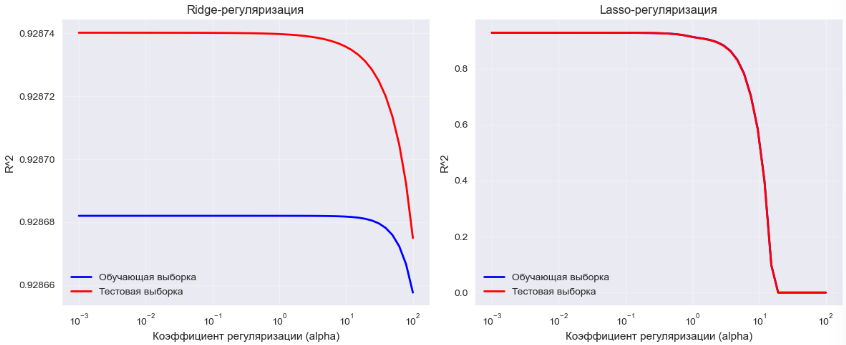
3. Зависимость точности от степени полинома.



4. Анализ переобучения.



5. Ridge и Lasso регуляризация.



6. Результаты подбора параметров.



# **4.** **Выводы**

В данной лабораторной работе реализованы методы линейной, полиномиальной и Ridge-регрессии для прогнозирования электропроизводства. Эксперименты показали, что повышение сложности модели снижает ошибку, однако, в каких-то случаях, ведёт к переобучению и требует регуляризации для предотвращения переобучения.