МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА –Российскийтехнологическийуниверситет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Дисциплина «Технологии извлечения знаний из больших данных»

Отчет
о проделанной практической работе

Выполнил студент 1 курса
Группы: ББМО-01-25
Мухаметшин Александр Ринатович

Москва
2025

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Построить по многорядному полиномиальному алгоритму метода группового учета аргументов модель предметной области, заданной ретроспективным паттерном. В качестве опорной функции использовать функцию: y = a0 + a1*x1 + a2*x2. В качестве обучающей выборки взять первые 20 значений паттерна, в качестве тестовой выборки, оставшиеся 5 паттернов исходной таблицы. Обучающую выборку поделить на две в соотношении: 60% и 40% (непосредственно обучающая выборка и проверочная выборка для отбора по МГУА). Провести сравнение значений исходной модели и модели, построенной по МГУА. Результат сравнения представить в таблице. Построить график значений исходной модели и модели, построенной по МГУА. Просчитать среднюю ошибку аппроксимации и сделать вывод о качестве обученной модели по методы МГУА.

ХОД РАБОТЫ

Часть 1. Реализация кода

На рисунке 1 импортируются необходимые библиотеки для работы с массивами, графиками, таблицами и комбинациями.

```
prac2.py > ...
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import pandas as pd
4   from itertools import combinations
5
```

Рисунок 1 – Импорт необходимых библиотек

Определяется класс MGUA с параметром для количества лучших моделей на уровне и пустыми атрибутами для хранения модели и селекций. Код представлен на рисунке 2.

```
6  class MGUA:
7      def __init__(self, max_models_per_level=5):
8          self.max_models_per_level = max_models_per_level
9          self.best_model = None
10         self.all_selections = []
11
```

Рисунок 2 – Создание класса

На рисунке 3 показан код метода, который вычисляет коэффициенты линейной регрессии методом наименьших квадратов, добавляя столбец единиц для свободного члена.

```
12     def least_squares(self, X, y):
13         # Добавляем столбец единиц для свободного члена a0
14         X_extended = np.c_[np.ones(X.shape[0]), X]
15
16         # Вычисляем коэффициенты по формуле: a = (X^T * X)^(-1) * X^T * y
17         coeffs, _, _, _ = np.linalg.lstsq(X_extended, y, rcond=None)
18
19         return coeffs
```

Рисунок 3 – Метод наименьших квадратов (МНК)

На рисунке 4 приведен код метода, который предсказывает значения y по формуле y = a0 + a1x1 + a2x2 с использованием матричного умножения.

```python
    def predict(self, X, coeffs):
        # y = a0 + a1*x1 + a2*x2
        X_extended = np.c_[np.ones(X.shape[0]), X]
        y_pred = X_extended @ coeffs  # Матричное умножение
        return y_pred
```

Рисунок 4 – Вычисление предсказанных значений

На рисунке 5 представлены методы вычислений среднеквадратичной ошибки (MSE) между истинными и предсказанными значениями и вычислений средней абсолютной процентной ошибки (MAPE) в процентах.

```python
    def calculate_mse(self, y_true, y_pred):
        # MSE = (1/n) * Σ(y_true - y_pred)²
        return np.mean((y_true - y_pred) ** 2)

    def calculate_mape(self, y_true, y_pred):
        # MAPE = (100/n) * Σ|(y_true - y_pred) / y_true|
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

Рисунок 5 – Методы вычисления ошибок

На рисунках 6-7 приведен основной метод, который обучает модель: генерирует комбинации признаков, строит модели, оценивает по MSE на проверочной выборке и отбирает лучшие.

```python
    def fit(self, X_train, y_train, X_valid, y_valid):
        n_features = X_train.shape[1]
        models = []

        print("ПЕРВЫЙ РЯД СЕЛЕКЦИИ")
        print()
        print(f"Количество признаков: {n_features}")
        print(f"Обучающая выборка: {X_train.shape[0]} наблюдений")
        print(f"Проверочная выборка: {X_valid.shape[0]} наблюдений")

        feature_combinations = list(combinations(range(n_features), 2))
        print(f"Количество попарных комбинаций: {len(feature_combinations)}")
        print()

        # Для каждой комбинации строим модель
        for idx, (i, j) in enumerate(feature_combinations, 1):
            X_train_pair = X_train[:, [i, j]]

            coeffs = self.least_squares(X_train_pair, y_train)

            # Оцениваем модель на проверочной выборке
            X_valid_pair = X_valid[:, [i, j]]
            y_pred = self.predict(X_valid_pair, coeffs)
            mse = self.calculate_mse(y_valid, y_pred)
```

Рисунок 6 – Метод обучения модели. Часть 1

```
            models.append({
                'features': (i, j),
                'feature_names': (f'x{i+1}', f'x{j+1}'),
                'coeffs': coeffs,
                'mse': mse
            })

            print(f"Модель {idx}: x{i+1}, x{j+1} -> MSE = {mse:.6f}")

        models.sort(key=lambda x: x['mse'])
        best_models = models[:self.max_models_per_level]

        print()
        print(f"Отобрано {len(best_models)} лучших моделей:")
        for idx, model in enumerate(best_models, 1):
            print(f"  {idx}. {model['feature_names']} - MSE: {model['mse']:.6f}")

        self.all_selections.append({
            'iteration': 1,
            'models': best_models,
            'best_mse': best_models[0]['mse']
        })

        self.best_model = best_models[0]

        print()
        print("ЛУЧШАЯ МОДЕЛЬ:")
        print(f"  Признаки: {self.best_model['feature_names']}")
        print(f"  Уравнение: y = {self.best_model['coeffs'][0]:.4f} + "
              f"{self.best_model['coeffs'][1]:.4f}*{self.best_model['feature_names'][0]} + "
              f"{self.best_model['coeffs'][2]:.4f}*{self.best_model['feature_names'][1]}")
        print(f"  MSE: {self.best_model['mse']:.6f}")
        print()
```

Рисунок 7 – Метод обучения модели. Часть 7

На рисунке 8 разработан метод, который предсказывает значения по лучшей модели, используя выбранные признаки.

```

    def predict_best(self, X):
        if self.best_model is None:
            raise ValueError("Модель не обучена! Вызовите метод fit().")

        i, j = self.best_model['features']
        X_pair = X[:, [i, j]]

        return self.predict(X_pair, self.best_model['coeffs'])
```

Рисунок 8 – Предсказание по лучшей найденной модели

На дальнейших рисунках будет показана основная программа, которая будет использовать определенные ранее класс и его методы.

На рисунке 9 показана загрузка данных.

```python
def main():
    # Исходные данные из таблицы 1
    # Столбцы: y, x1, x2, x3, x4
    data = np.array([
        [0.904, 75.5, 25.2, 3343, 77], [0.922, 78.5, 21.8, 3001, 78.2],
        [0.763, 78.4, 25.7, 3101, 68], [0.923, 77.7, 17.8, 3543, 77.2],
        [0.918, 84.4, 15.9, 3237, 77.2], [0.906, 75.9, 22.4, 3330, 77.2],
        [0.905, 76, 20.6, 3808, 75.7], [0.545, 67.5, 25.2, 2415, 62.6],
        [0.894, 78.2, 20.7, 3295, 78], [0.9, 78.1, 17.5, 3504, 78.2],
        [0.932, 78.6, 19.7, 30565, 79], [0.74, 84, 18.5, 3007, 67.6],
        [0.701, 59.2, 54.4, 2844, 69.8], [0.744, 90.2, 23, 2861, 68.4],
        [0.921, 72.8, 20.2, 3259, 77.9], [0.927, 67.7, 25.2, 3350, 78.1],
        [0.802, 82.6, 22.4, 3344, 72.5], [0.747, 74.4, 22.7, 2704, 66.6],
        [0.927, 83.3, 18.1, 3642, 76.7], [0.721, 83.7, 20.1, 2753, 68.8],
        [0.913, 73.8, 17.3, 2916, 76.8], [0.918, 79.2, 16.8, 3551, 78.1],
        [0.833, 71.5, 29.9, 3177, 73.9], [0.914, 75.3, 20.3, 3280, 78.6],
        [0.923, 79, 14.1, 3160, 78.5]
    ])

    # Разделяем данные на признаки (X) и целевую переменную (y)
    y_all = data[:, 0]  # Первый столбец - выходная величина y
    X_all = data[:, 1:]  # Остальные столбцы - признаки x1, x2, x3, x4

    print("ИСХОДНЫЕ ДАННЫЕ")
    print(f"Всего наблюдений: {len(data)}")
    print(f"Количество признаков: {X_all.shape[1]}")
    print()
```

Рисунок 9 – Загрузка данных

Далее данные делятся на обучающую, проверочную и тестовую выборки согласно заданию. Код показан на рисунке 10.

```python
    X_train_full = X_all[:20]
    y_train_full = y_all[:20]

    X_test = X_all[20:]
    y_test = y_all[20:]

    split_idx = int(len(X_train_full) * 0.6)

    X_train = X_train_full[:split_idx]
    y_train = y_train_full[:split_idx]

    X_valid = X_train_full[split_idx:]
    y_valid = y_train_full[split_idx:]

    print("РАЗДЕЛЕНИЕ ДАННЫХ:")
    print(f"  Обучающая выборка: {len(X_train)} наблюдений (60% от 20)")
    print(f"  Проверочная выборка: {len(X_valid)} наблюдений (40% от 20)")
    print(f"  Тестовая выборка: {len(X_test)} наблюдений")
    print()
```

Рисунок 10 – Разбиение данных на обучающую, проверочную и тестовую выборки

Создается экземпляр MGUA и вызывается метод fit для обучения. Вычисляются предсказания и метрики MSE/MAPE на тестовой выборке. Код представлен на рисунке 11.

```
151     mgua = MGUA(max_models_per_level=5)
152     mgua.fit(X_train, y_train, X_valid, y_valid)
153
154     print("ТЕСТИРОВАНИЕ МОДЕЛИ")
155
156     y_pred_all = mgua.predict_best(X_all)
157
158     y_pred_test = mgua.predict_best(X_test)
159
160     test_mse = mgua.calculate_mse(y_test, y_pred_test)
161     test_mape = mgua.calculate_mape(y_test, y_pred_test)
162
163     print(f"MSE на тестовой выборке: {test_mse:.6f}")
164     print(f"MAPE на тестовой выборке: {test_mape:.2f}%")
165     print()
```

Рисунок 11 – Обучение и тестирование модели

Создается и выводится таблица с исходными и предсказанными значениями, ошибками (рис. 12).

```
167 print("ТАБЛИЦА СРАВНЕНИЯ ЗНАЧЕНИЙ")
168
169 results_df = pd.DataFrame({
170     '№': range(1, len(y_all) + 1),
171     'Исходное Y': y_all,
172     'Предсказанное Y': y_pred_all,
173     'Ошибка': y_all - y_pred_all,
174     'Ошибка, %': np.abs((y_all - y_pred_all) / y_all) * 100
175 })
176
177 pd.set_option('display.max_rows', None)
178 pd.set_option('display.float_format', '{:.4f}'.format)
179 print(results_df.to_string(index=False))
180 print()
181 print("Примечание: строки 21-25 - тестовая выборка")
182 print()
```

Рисунок 12 – Создание таблицы сравнения значений

Далее определяется качество модели по MAPE и выводится описание (рис. 13).

```
    print("ВЫВОДЫ О КАЧЕСТВЕ МОДЕЛИ")

    if test_mape < 5:
        quality = "ОТЛИЧНОЕ"
        comment = "Модель имеет высокую точность предсказаний."
    elif test_mape < 10:
        quality = "ХОРОШЕЕ"
        comment = "Модель показывает хорошие результаты."
    elif test_mape < 20:
        quality = "УДОВЛЕТВОРИТЕЛЬНОЕ"
        comment = "Модель приемлема для использования, но есть резерв для улучшения."
    else:
        quality = "НИЗКОЕ"
        comment = "Модель требует доработки или использования других методов."

    print(f"Качество модели: {quality} (MAPE = {test_mape:.2f}%)")
    print(f"Комментарий: {comment}")
    print()
```

Рисунок 13 – Выводы о качестве модели

В итоге строятся два графика: общий и для тестовой выборки, с отображением и запускается программа (рис. 14).

```
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.plot(range(1, len(y_all) + 1), y_all, 'o-',
             label='Исходные данные', linewidth=2, markersize=6)
    plt.plot(range(1, len(y_all) + 1), y_pred_all, 's--',
             label='Модель МГУА', linewidth=2, markersize=6)
    plt.axvline(x=20.5, color='red', linestyle=':', linewidth=2,
                label='Граница тестовой выборки')
    plt.xlabel('№ наблюдения', fontsize=12)
    plt.ylabel('Y', fontsize=12)
    plt.title('Сравнение исходных и предсказанных значений', fontsize=14, fontweight='bold')
    plt.legend()
    plt.grid(True, alpha=0.3)

    plt.subplot(1, 2, 2)
    test_indices = range(21, 26)
    plt.plot(test_indices, y_test, 'o-',
             label='Исходные данные', linewidth=2, markersize=8)
    plt.plot(test_indices, y_pred_test, 's--',
             label='Модель МГУА', linewidth=2, markersize=8)
    plt.xlabel('№ наблюдения', fontsize=12)
    plt.ylabel('Y', fontsize=12)
    plt.title('Тестовая выборка (детально)', fontsize=14, fontweight='bold')
    plt.legend()
    plt.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

Рисунок 14 – Отображение графика и запуск программы

Полный код программы представлен на листинге А1.

Часть 2. Результат работы программы

На рисунке 15 выводится информация о размере набора данных и количестве признаков и описание размеров выборок после разделения.

```
ИСХОДНЫЕ ДАННЫЕ
Всего наблюдений: 25
Количество признаков: 4

РАЗДЕЛЕНИЕ ДАННЫХ:
  Обучающая выборка: 12 наблюдений (60% от 20)
  Проверочная выборка: 8 наблюдений (40% от 20)
  Тестовая выборка: 5 наблюдений
```

Рисунок 15 – Разбиение тестовых данных

На рисунке 16 выводится информация о селекции моделей: MSE для каждой комбинации, топ-5 и лучшая модель с уравнением.

```
ПЕРВЫЙ РЯД СЕЛЕКЦИИ

Количество признаков: 4
Обучающая выборка: 12 наблюдений
Проверочная выборка: 8 наблюдений
Количество попарных комбинаций: 6

Модель 1: x1, x2 -> MSE = 0.029164
Модель 2: x1, x3 -> MSE = 0.024246
Модель 3: x1, x4 -> MSE = 0.001729
Модель 4: x2, x3 -> MSE = 0.025294
Модель 5: x2, x4 -> MSE = 0.000655
Модель 6: x3, x4 -> MSE = 0.000980

Отобрано 5 лучших моделей:
  1. ('x2', 'x4') - MSE: 0.000655
  2. ('x3', 'x4') - MSE: 0.000980
  3. ('x1', 'x4') - MSE: 0.001729
  4. ('x1', 'x3') - MSE: 0.024246
  5. ('x2', 'x3') - MSE: 0.025294

ЛУЧШАЯ МОДЕЛЬ:
  Признаки: ('x2', 'x4')
  Уравнение: y = -0.6471 + -0.0012*x2 + 0.0204*x4
  MSE: 0.000655
```

Рисунок 16 – Селекция моделей и выбор лучшей

Далее выводятся метрики MSE и MAPE на тестовой выборке и таблица

с номерами наблюдений, исходными и предсказанными Y, ошибками и процентными ошибками; примечание о тестовой выборке. Вывод представлен на рисунке 17.

```
ТЕСТИРОВАНИЕ МОДЕЛИ
MSE на тестовой выборке: 0.000209
MAPE на тестовой выборке: 1.45%

ТАБЛИЦА СРАВНЕНИЯ ЗНАЧЕНИЙ
 №  Исходное Y  Предсказанное Y  Ошибка  Ошибка, %
 1      0.9040           0.8973  0.0067      0.7464
 2      0.9220           0.9257 -0.0037      0.3997
 3      0.7630           0.7128  0.0502      6.5822
 4      0.9230           0.9099  0.0131      1.4243
 5      0.9180           0.9120  0.0060      0.6492
 6      0.9060           0.9046  0.0014      0.1588
 7      0.9050           0.8760  0.0290      3.2064
 8      0.5450           0.6030 -0.0580     10.6447
 9      0.8940           0.9229 -0.0289      3.2286
10      0.9000           0.9306 -0.0306      3.4036
11      0.9320           0.9444 -0.0124      1.3356
12      0.7400           0.7129  0.0271      3.6636
13      0.7010           0.7165 -0.0155      2.2161
14      0.7440           0.7241  0.0199      2.6804
15      0.9210           0.9214 -0.0004      0.0430
16      0.9270           0.9197  0.0073      0.7843
17      0.8020           0.8085 -0.0065      0.8136
18      0.7470           0.6876  0.0594      7.9487
19      0.9270           0.8993  0.0277      2.9890
20      0.7210           0.7356 -0.0146      2.0205
21      0.9130           0.9023  0.0107      1.1768
22      0.9180           0.9294 -0.0114      1.2412
23      0.8330           0.8285  0.0045      0.5400
24      0.9140           0.9356 -0.0216      2.3615
25      0.9230           0.9407 -0.0177      1.9149

Примечание: строки 21-25 - тестовая выборка
```

Рисунок 17 – Тестирование модели и таблица сравнений значений

На рисунке 18 выводится оценка качества по MAPE и краткое описание модели.

```
ВЫВОДЫ О КАЧЕСТВЕ МОДЕЛИ
Качество модели: ОТЛИЧНОЕ (MAPE = 1.45%)
Комментарий: Модель имеет высокую точность предсказаний.
```

Рисунок 18 – Оценка качества модели

В конце работы программы отображается график с двумя подграфиками: общий и тестовая выборка. Итоговый график представлен на рисунке 19.

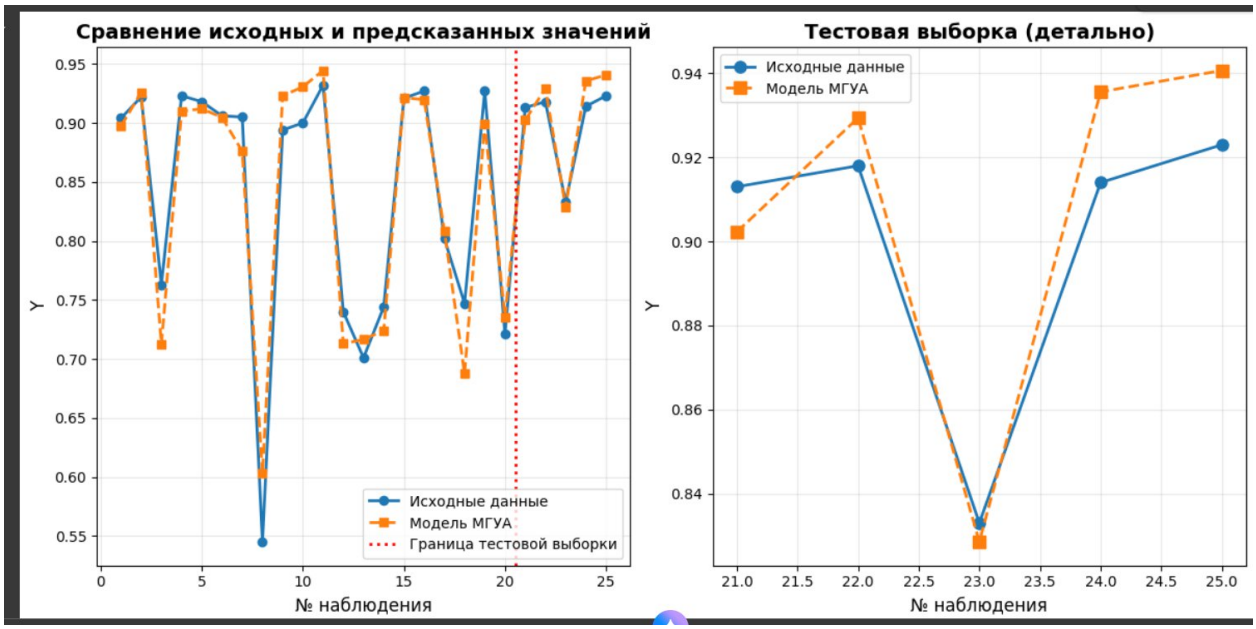

Рисунок 19 – График значений исходной модели и модели, построенной по МГУА

ВЫВОД

В результате выполнения практической работы был получен опыт построения прогностических моделей для ретроспективного паттерна данных с использованием метода группового учета аргументов (МГУА) в Python. Реализован многорядный полиномиальный алгоритм с опорной функцией y = a0 + a1xi + a2xj, коэффициенты которой найдены методом наименьших квадратов. Проведено разбиение выборки на обучающую (первые 20 наблюдений, разделенных в соотношении 60/40) и тестовую (оставшиеся 5), выполнен перебор попарных комбинаций признаков с отбором лучших по критерию MSE. Построена оптимальная модель на основе признаков x2 и x4 с уравнением y = -0.6471 - 0.0012x2 + 0.0204x4, оценена ее точность на тестовой выборке (MAPE = 1.45%), что подтвердило отличное качество аппроксимации. Кроме того, проведено сравнение исходных и предсказанных значений в таблице и на графике, что обеспечило понимание зависимостей в данных и эффективности метода МГУА.

ПРИЛОЖЕНИЕ А

Листинг А1 – Полный код программы

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from itertools import combinations

class MGUA:
    def __init__(self, max_models_per_level=5):
        self.max_models_per_level = max_models_per_level
        self.best_model = None
        self.all_selections = []

    def least_squares(self, X, y):
        # Добавляем столбец единиц для свободного члена a0
        X_extended = np.c_[np.ones(X.shape[0]), X]

        # Вычисляем коэффициенты по формуле: a = (X^T * X)^(-1) * X^T * y
        coeffs, _, _, _ = np.linalg.lstsq(X_extended, y, rcond=None)

        return coeffs

    def predict(self, X, coeffs):
        # y = a0 + a1*x1 + a2*x2
        X_extended = np.c_[np.ones(X.shape[0]), X]
        y_pred = X_extended @ coeffs  # Матричное умножение
        return y_pred

    def calculate_mse(self, y_true, y_pred):
        # MSE = (1/n) * Σ(y_true - y_pred)²
        return np.mean((y_true - y_pred) ** 2)

    def calculate_mape(self, y_true, y_pred):
        # MAPE = (100/n) * Σ||(y_true - y_pred) / y_true||
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

    def fit(self, X_train, y_train, X_valid, y_valid):
        n_features = X_train.shape[1]
        models = []

        print("ПЕРВЫЙ РЯД СЕЛЕКЦИИ")
        print()
        print(f"Количество признаков: {n_features}")
        print(f"Обучающая выборка: {X_train.shape[0]} наблюдений")
        print(f"Проверочная выборка: {X_valid.shape[0]} наблюдений")

        feature_combinations = list(combinations(range(n_features), 2))
        print(f"Количество попарных комбинаций: {len(feature_combinations)}")
        print()

        # Для каждой комбинации строим модель
        for idx, (i, j) in enumerate(feature_combinations, 1):
            X_train_pair = X_train[:, [i, j]]

            coeffs = self.least_squares(X_train_pair, y_train)

            # Оцениваем модель на проверочной выборке
            X_valid_pair = X_valid[:, [i, j]]
            y_pred = self.predict(X_valid_pair, coeffs)
            mse = self.calculate_mse(y_valid, y_pred)

            models.append({
```

```
                'features': (i, j),
                'feature_names': (f'x{i+1}', f'x{j+1}'),
                'coeffs': coeffs,
                'mse': mse
            })

            print(f"Модель {idx}: x{i+1}, x{j+1} -> MSE = {mse:.6f}")

        models.sort(key=lambda x: x['mse'])
        best_models = models[:self.max_models_per_level]

        print()
        print(f"Отобрано {len(best_models)} лучших моделей:")
        for idx, model in enumerate(best_models, 1):
            print(f"         {idx}.     {model['feature_names']}     -     MSE:
{model['mse']:.6f}")

        self.all_selections.append({
            'iteration': 1,
            'models': best_models,
            'best_mse': best_models[0]['mse']
        })

        self.best_model = best_models[0]

        print()
        print("ЛУЧШАЯ МОДЕЛЬ:")
        print(f"  Признаки: {self.best_model['feature_names']}")
        print(f"  Уравнение: y = {self.best_model['coeffs'][0]:.4f} + "

f"{self.best_model['coeffs'][1]:.4f}*{self.best_model['feature_names'][0]}   +
"

f"{self.best_model['coeffs'][2]:.4f}*{self.best_model['feature_names'][1]}")
        print(f"  MSE: {self.best_model['mse']:.6f}")
        print()

    def predict_best(self, X):
        if self.best_model is None:
            raise ValueError("Модель не обучена! Вызовите метод fit().")

        i, j = self.best_model['features']
        X_pair = X[:, [i, j]]

        return self.predict(X_pair, self.best_model['coeffs'])

def main():
    # Исходные данные из таблицы 1
    # Столбцы: y, x1, x2, x3, x4
    data = np.array([
        [0.904, 75.5, 25.2, 3343, 77], [0.922, 78.5, 21.8, 3001, 78.2],
        [0.763, 78.4, 25.7, 3101, 68], [0.923, 77.7, 17.8, 3543, 77.2],
        [0.918, 84.4, 15.9, 3237, 77.2], [0.906, 75.9, 22.4, 3330, 77.2],
        [0.905, 76, 20.6, 3808, 75.7], [0.545, 67.5, 25.2, 2415, 62.6],
        [0.894, 78.2, 20.7, 3295, 78], [0.9, 78.1, 17.5, 3504, 78.2],
        [0.932, 78.6, 19.7, 30565, 79], [0.74, 84, 18.5, 3007, 67.6],
        [0.701, 59.2, 54.4, 2844, 69.8], [0.744, 90.2, 23, 2861, 68.4],
        [0.921, 72.8, 20.2, 3259, 77.9], [0.927, 67.7, 25.2, 3350, 78.1],
        [0.802, 82.6, 22.4, 3344, 72.5], [0.747, 74.4, 22.7, 2704, 66.6],
        [0.927, 83.3, 18.1, 3642, 76.7], [0.721, 83.7, 20.1, 2753, 68.8],
        [0.913, 73.8, 17.3, 2916, 76.8], [0.918, 79.2, 16.8, 3551, 78.1],
        [0.833, 71.5, 29.9, 3177, 73.9], [0.914, 75.3, 20.3, 3280, 78.6],
        [0.923, 79, 14.1, 3160, 78.5]
```

```
])

# Разделяем данные на признаки (X) и целевую переменную (y)
y_all = data[:, 0]  # Первый столбец - выходная величина y
X_all = data[:, 1:]  # Остальные столбцы - признаки x1, x2, x3, x4

print("ИСХОДНЫЕ ДАННЫЕ")
print(f"Всего наблюдений: {len(data)}")
print(f"Количество признаков: {X_all.shape[1]}")
print()

X_train_full = X_all[:20]
y_train_full = y_all[:20]

X_test = X_all[20:]
y_test = y_all[20:]

split_idx = int(len(X_train_full) * 0.6)

X_train = X_train_full[:split_idx]
y_train = y_train_full[:split_idx]

X_valid = X_train_full[split_idx:]
y_valid = y_train_full[split_idx:]

print("РАЗДЕЛЕНИЕ ДАННЫХ:")
print(f"  Обучающая выборка: {len(X_train)} наблюдений (60% от 20)")
print(f"  Проверочная выборка: {len(X_valid)} наблюдений (40% от 20)")
print(f"  Тестовая выборка: {len(X_test)} наблюдений")
print()

mgua = MGUA(max_models_per_level=5)
mgua.fit(X_train, y_train, X_valid, y_valid)

print("ТЕСТИРОВАНИЕ МОДЕЛИ")

y_pred_all = mgua.predict_best(X_all)

y_pred_test = mgua.predict_best(X_test)

test_mse = mgua.calculate_mse(y_test, y_pred_test)
test_mape = mgua.calculate_mape(y_test, y_pred_test)

print(f"MSE на тестовой выборке: {test_mse:.6f}")
print(f"MAPE на тестовой выборке: {test_mape:.2f}%")
print()

print("ТАБЛИЦА СРАВНЕНИЯ ЗНАЧЕНИЙ")

results_df = pd.DataFrame({
    '№': range(1, len(y_all) + 1),
    'Исходное Y': y_all,
    'Предсказанное Y': y_pred_all,
    'Ошибка': y_all - y_pred_all,
    'Ошибка, %': np.abs((y_all - y_pred_all) / y_all) * 100
})

pd.set_option('display.max_rows', None)
pd.set_option('display.float_format', '{:.4f}'.format)
print(results_df.to_string(index=False))
print()
print("Примечание: строки 21-25 - тестовая выборка")
print()
```

```
    print("ВЫВОДЫ О КАЧЕСТВЕ МОДЕЛИ")

    if test_mape < 5:
        quality = "ОТЛИЧНОЕ"
        comment = "Модель имеет высокую точность предсказаний."
    elif test_mape < 10:
        quality = "ХОРОШЕЕ"
        comment = "Модель показывает хорошие результаты."
    elif test_mape < 20:
        quality = "УДОВЛЕТВОРИТЕЛЬНОЕ"
        comment = "Модель приемлема для использования, но есть резерв для
улучшения."
    else:
        quality = "НИЗКОЕ"
        comment = "Модель требует доработки или использования других
методов."

    print(f"Качество модели: {quality} (MAPE = {test_mape:.2f}%)")
    print(f"Комментарий: {comment}")
    print()

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.plot(range(1, len(y_all) + 1), y_all, 'o-',
             label='Исходные данные', linewidth=2, markersize=6)
    plt.plot(range(1, len(y_all) + 1), y_pred_all, 's--',
             label='Модель МГУА', linewidth=2, markersize=6)
    plt.axvline(x=20.5, color='red', linestyle=':', linewidth=2,
                label='Граница тестовой выборки')
    plt.xlabel('№ наблюдения', fontsize=12)
    plt.ylabel('Y', fontsize=12)
    plt.title('Сравнение исходных и предсказанных значений', fontsize=14,
fontweight='bold')
    plt.legend()
    plt.grid(True, alpha=0.3)

    plt.subplot(1, 2, 2)
    test_indices = range(21, 26)
    plt.plot(test_indices, y_test, 'o-',
             label='Исходные данные', linewidth=2, markersize=8)
    plt.plot(test_indices, y_pred_test, 's--',
             label='Модель МГУА', linewidth=2, markersize=8)
    plt.xlabel('№ наблюдения', fontsize=12)
    plt.ylabel('Y', fontsize=12)
    plt.title('Тестовая выборка (детально)', fontsize=14, fontweight='bold')
    plt.legend()
    plt.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```