|  |
| --- |
|  |
| 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 3](#_Toc1)

[ХОД РАБОТЫ 4](#_Toc2)

[ВЫВОД 13](#_Toc3)

[ПРИЛОЖЕНИЕ А 14](#_Toc4)

# 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.

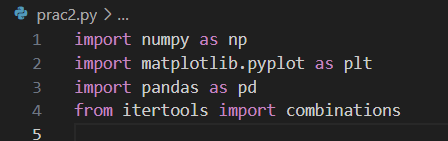


Рисунок 1 – Импорт необходимых библиотек

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

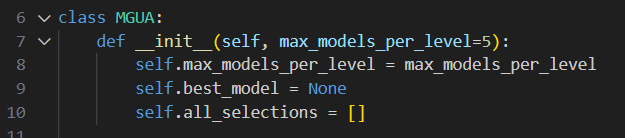


Рисунок 2 – Создание класса

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

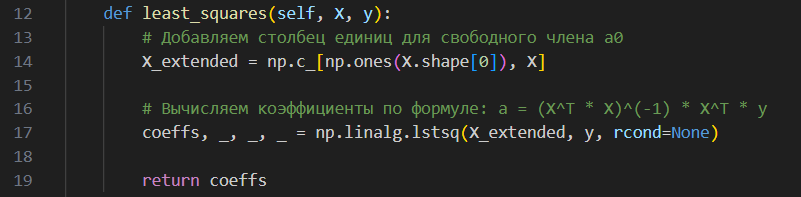


Рисунок 3 – Метод наименьших квадратов (МНК)

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

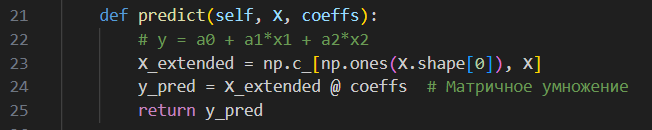


Рисунок 4 – Вычисление предсказанных значений

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

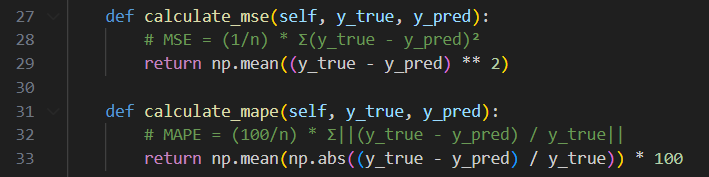


Рисунок 5 – Методы вычисления ошибок

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

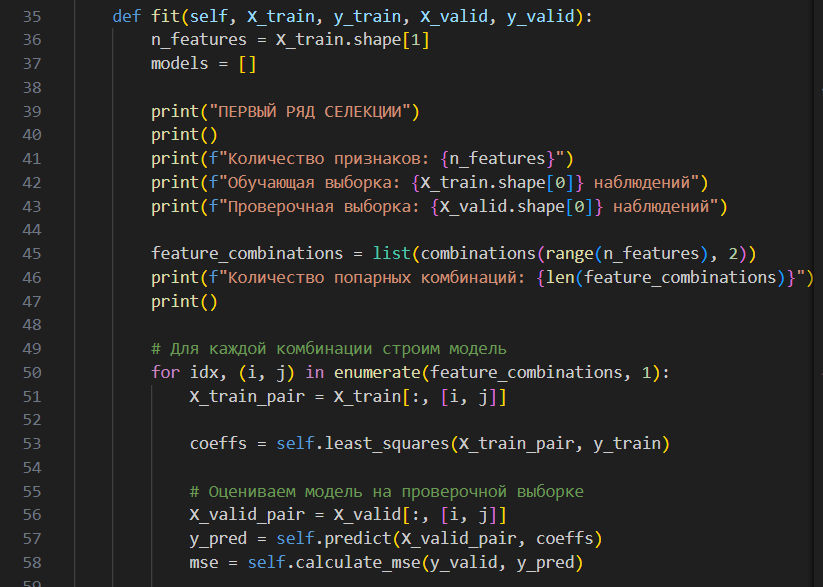


Рисунок 6 – Метод обучения модели. Часть 1

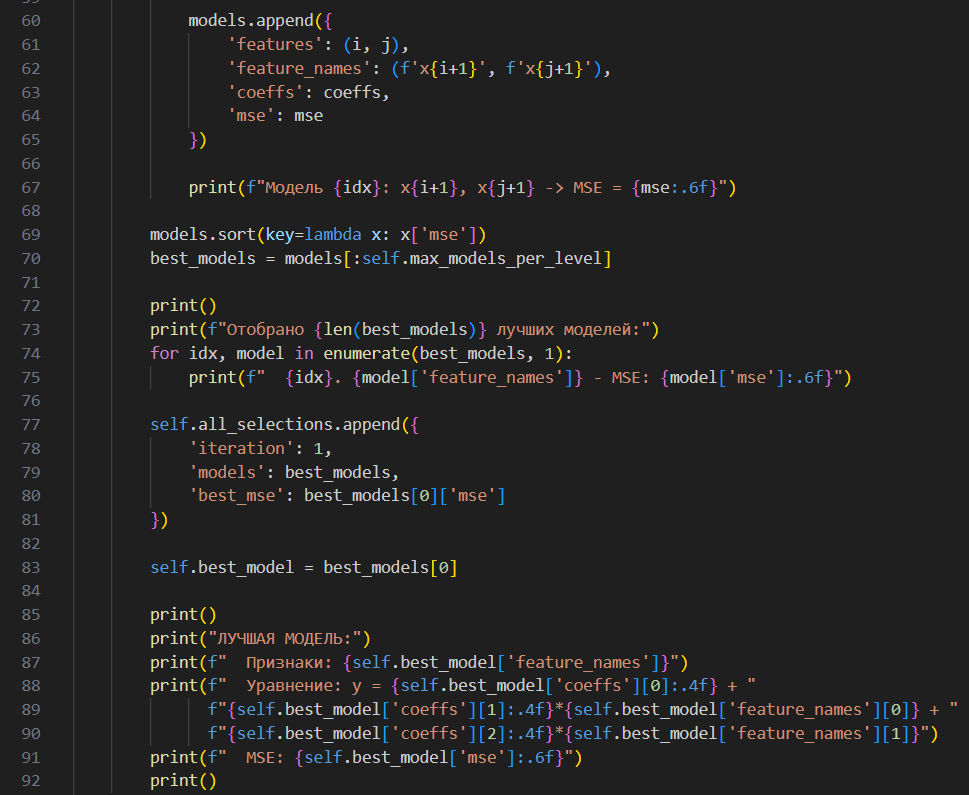


Рисунок 7 – Метод обучения модели. Часть 7

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

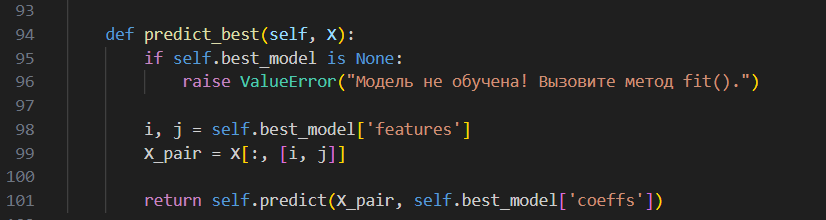


Рисунок 8 – Предсказание по лучшей найденной модели

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

На рисунке 9 показана загрузка данных.

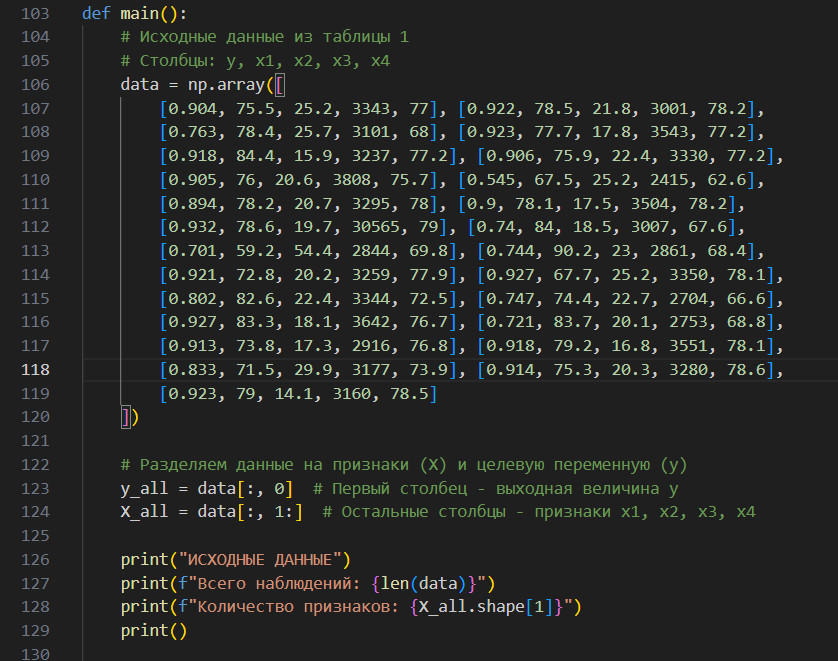


Рисунок 9 – Загрузка данных

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

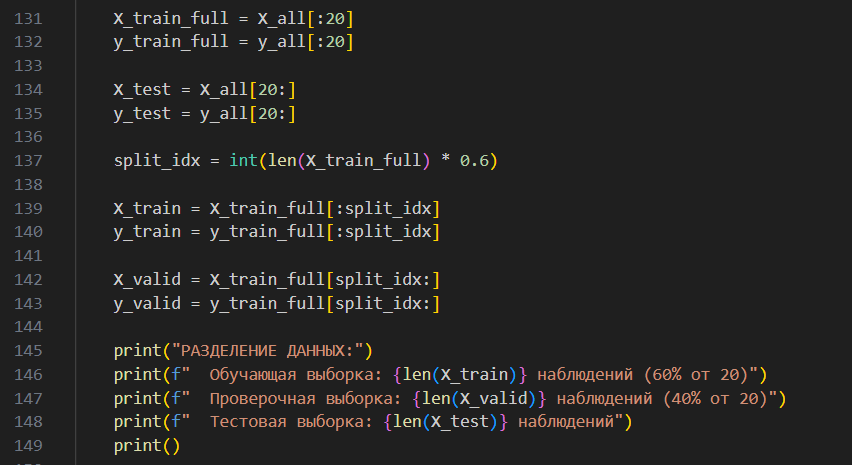


Рисунок 10 – Разбиение данных на обучающую, проверочную и тестовую выборки

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

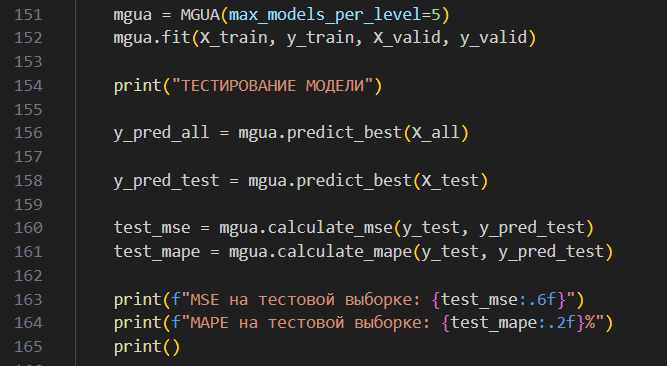


Рисунок 11 – Обучение и тестирование модели

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

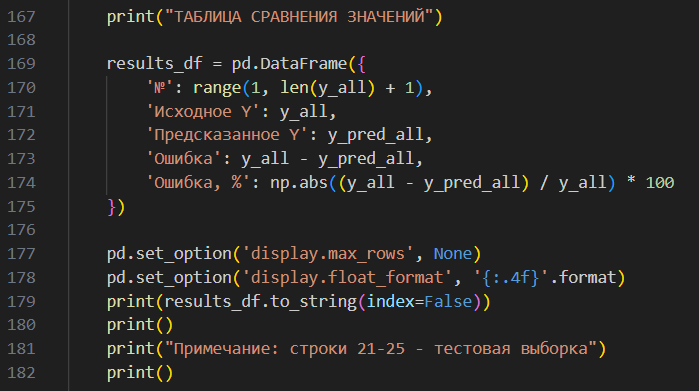


Рисунок 12 – Создание таблицы сравнения значений

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

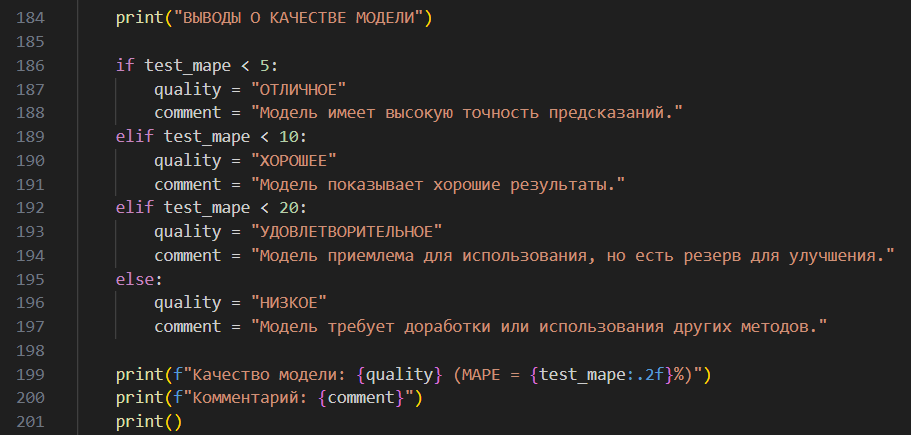


Рисунок 13 – Выводы о качестве модели

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



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.

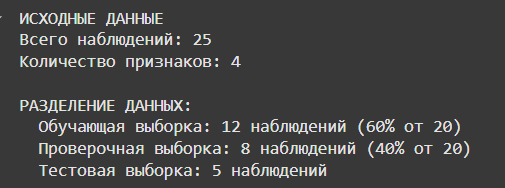


Рисунок 15 – Разбиение тестовых данных

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

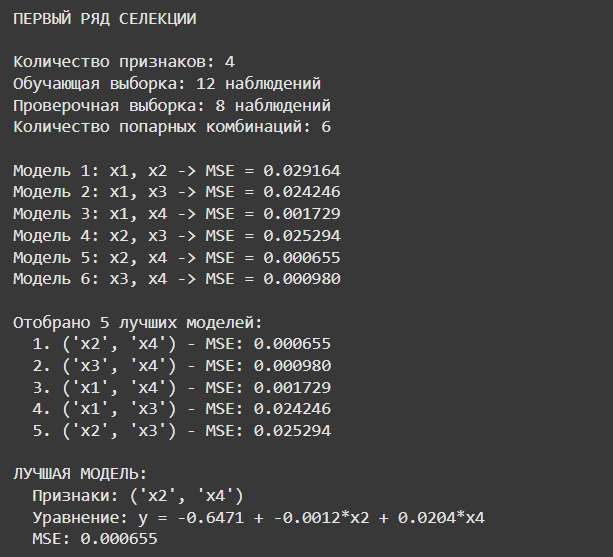


Рисунок 16 – Селекция моделей и выбор лучшей

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.

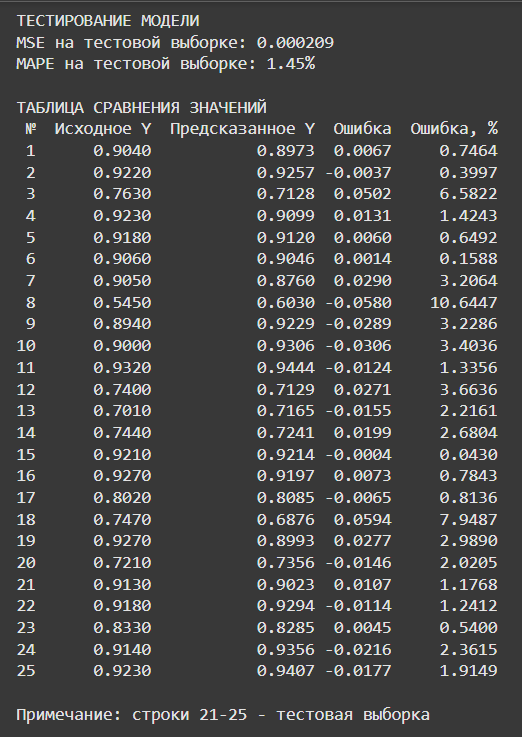


Рисунок 17 – Тестирование модели и таблица сравнений значений

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

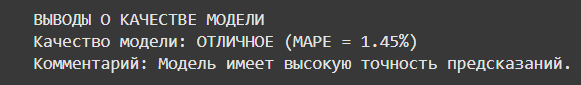


Рисунок 18 – Оценка качества модели

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

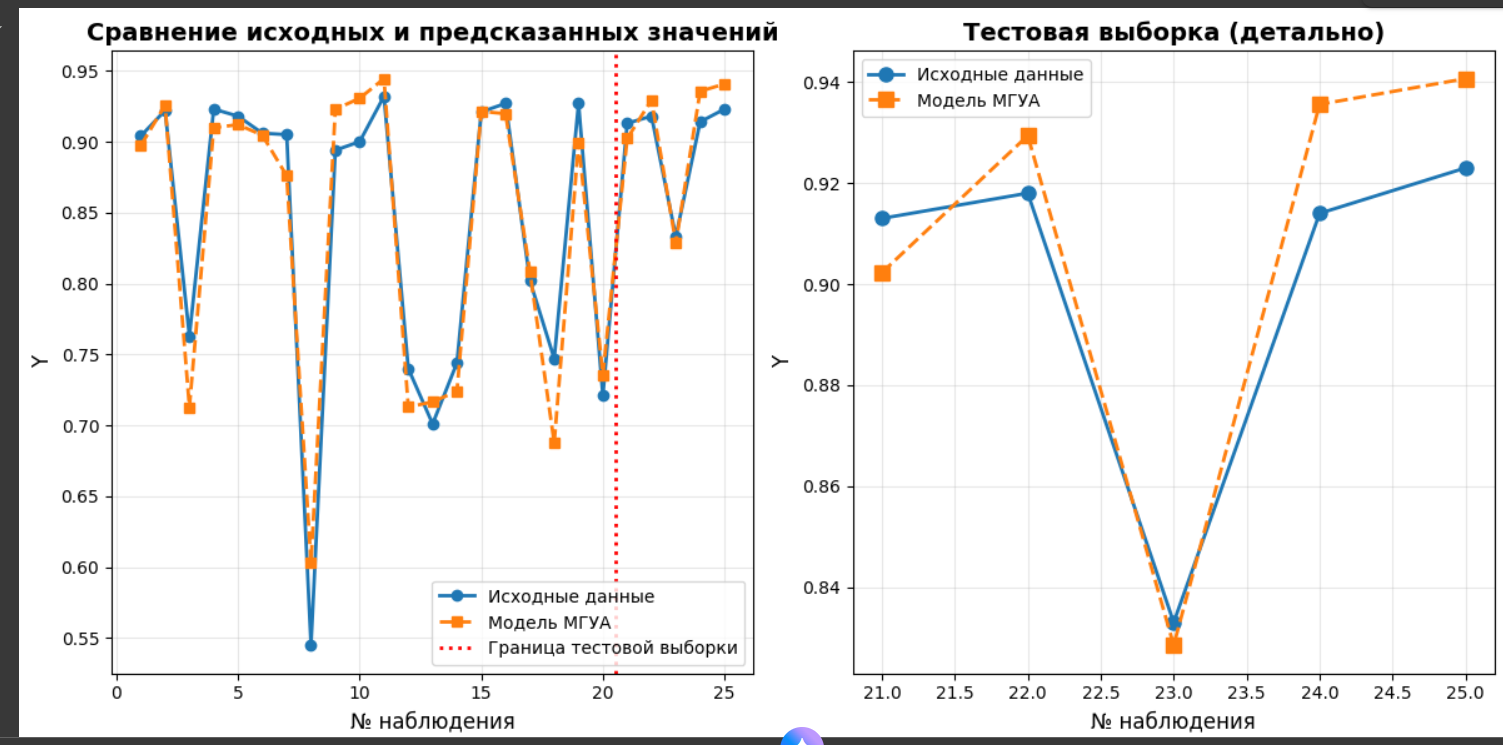


Рисунок 19 – График значений исходной модели и модели, построенной по МГУА

# 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() |