ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Профессор |  |  |  | Колесникова С.А. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3  Моделирование временных рядов. МНК |
| **по дисциплине: Компьютерное моделирование** |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134К |  |  |  | Самарин Д. В. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2024

**Цель:**

Цель настоящей работы – освоить средства моделирования стохастических временных рядов.

**Ход работы:**

1. Ознакомиться со справочными сведениями.

2. Сформулировать задачу МНК при построении функции регрессии.

3. Разработать программу, моделирующую алгоритм поиска оптимального решения для формализованной задачи, используя математический пакет MatLab или язык программированияPython:

a. Самостоятельно реализовать МНК для решения задачи поиска коэффициентов модели, заданной в виде полинома второго порядка 𝑓1(𝑥) = 𝑎2𝑥^2 + 𝑎1𝑥 + 𝑎0.

b. С использованием встроенной реализации МНК в MatLab или Python подобрать степень 𝑝 полиномиальной модели 𝑓2(𝑥) = ∑ p, i=0 ajx^j наилучшим образом соответствующей исходным данным при визуальной оценке на графике. Для этого построить график с исходными данными (крестики, точки и т.п.) и различными вариантами полиномиальных моделей степени 𝑝, где 𝑝 ≠ 2.

c. Аппроксимировать данные функциональной моделью вида 𝑓3(𝑥) = √^3(𝑥 + 1)+1

d. Используя скорректированный коэффициент детерминации R^2adj определить наилучшую из трех моделей 𝑓1(𝑥), 𝑓2(𝑥), 𝑓3(𝑥).

4. Сделать прогноз на один шаг. Указать, каким образом можно оценить точность прогноза.

5. Составить и представить преподавателю отчет о работе.

6. Уметь формулировать основные понятия, связанные с МНК, приводить необходимые формулы и их обоснования.

Справочные сведения

В рамках лабораторной работы 3 рассматривается метод численного моделирования функции по экспериментальным данным с целью аппроксимации фактических данных (с целью прогнозирования, в том числе).

Для сравнения моделей между собой обычно используют оценку погрешностей аппроксимации или коэффициент детерминации. В данной лабораторной работе предлагается использовать последний.

Коэффициент детерминации модели описывает долю дисперсии зависимой переменной y, объясняемую моделью.

Реализация МНК в математических пакетах осуществляется с помощью функции polyfit в MatLab или numpy.polyfit в Python.

**Заданные параметры:**

Исследуется динамика цен на недвижимость. Для этого собраны данные о средней стоимости 1м^2 в новостройках Санкт-Петербурга (руб.) Y(t) за лето 2019 года (шаг измерений – 14 дней). Обосновать и построить тренд данного ряда. Оценить достоверность уточненной по МНК модели.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| t | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Y(t) | 115113,8 | 116620,5 | 117377,2 | 116770,5 | 118621,8 | 118173,4 | 118447 |

Необходимо определить и аргументировать тренд для представленных данных. Тренд здесь трактуется как математическая модель, которая описывает взаимосвязь цены и времени. Для этой цели будет использован метод наименьших квадратов (МНК), который позволит построить модель и затем оценить её эффективность.

Метод наименьших квадратов (МНК) является распространённым инструментом для нахождения коэффициентов в регрессионных моделях. Суть данного метода заключается в минимизации суммы квадратов отклонений между фактическими значениями и теми, которые предсказаны моделью. Таким образом, задача сводится к нахождению таких коэффициентов, при которых суммарная ошибка минимальна.

МНК используется в случаях, когда существует зависимость между переменными (например, между ценой и временем), и эта зависимость может быть описана различными математическими моделями, такими как линейная или полиномиальная.

Принцип работы МНК заключается в минимизации суммы квадратов разностей между наблюдаемыми значениями Y и предсказанными Y^, которые вычисляются на основе модели. Например, для полиномиальной модели второй степени зависимость может быть выражена как Y(t) = a\_2 t^2 + a\_1 t + a\_0, где t — это время, а a\_0, a\_1, a\_2​ — это параметры, которые необходимо найти.

Алгоритм МНК:

* Определяются коэффициенты модели.
* Для каждого набора коэффициентов вычисляется ошибка, и выбираются такие параметры, которые минимизируют эту ошибку.

Ключевые критерии оценки модели МНК:

1. Коэффициент детерминации (R²) — измеряет, насколько хорошо модель объясняет изменчивость данных. Чем ближе его значение к 1, тем точнее модель.
2. Скорректированный коэффициент детерминации (R²\_adj) — улучшенная версия R², которая учитывает количество параметров модели и корректирует показатель R² с учётом числа предсказанных переменных.
3. Среднеквадратическая ошибка (MSE) — измеряет среднее квадратное отклонение между реальными и предсказанными значениями.
4. Средняя абсолютная ошибка (MAE) — показывает среднее абсолютное отклонение между фактическими и прогнозируемыми значениями.

Полиномы и их степени

Полиномы представляют собой математические выражения, которые включают сумму произведений переменных, возведённых в различные степени, и соответствующих коэффициентов. Степень полинома играет ключевую роль в гибкости модели: использование полиномов более высоких степеней (например, третьей, четвёртой и т. д.) позволяет модели лучше адаптироваться к данным. Однако при слишком высокой степени полинома существует риск переобучения, когда модель начинает чрезмерно подстраиваться под данные.

Как осуществляется прогноз

После того как модель (полином, кубический корень и другие) построена, прогноз для новых данных вычисляется с использованием найденных коэффициентов. Например, для полинома второй степени прогноз для нового временного интервала (например, 8-й) будет осуществляться по формуле:

Y(t)=a2​t^2+a1​t+a0​

где коэффициенты a2,a1,a0a\_2, a\_1, a\_0a2​,a1​,a0​ получены ранее в процессе построения модели.

Оценка точности прогноза осуществляется с использованием таких метрик, как:

* MSE — среднеквадратичная ошибка,
* MAE — средняя абсолютная ошибка.

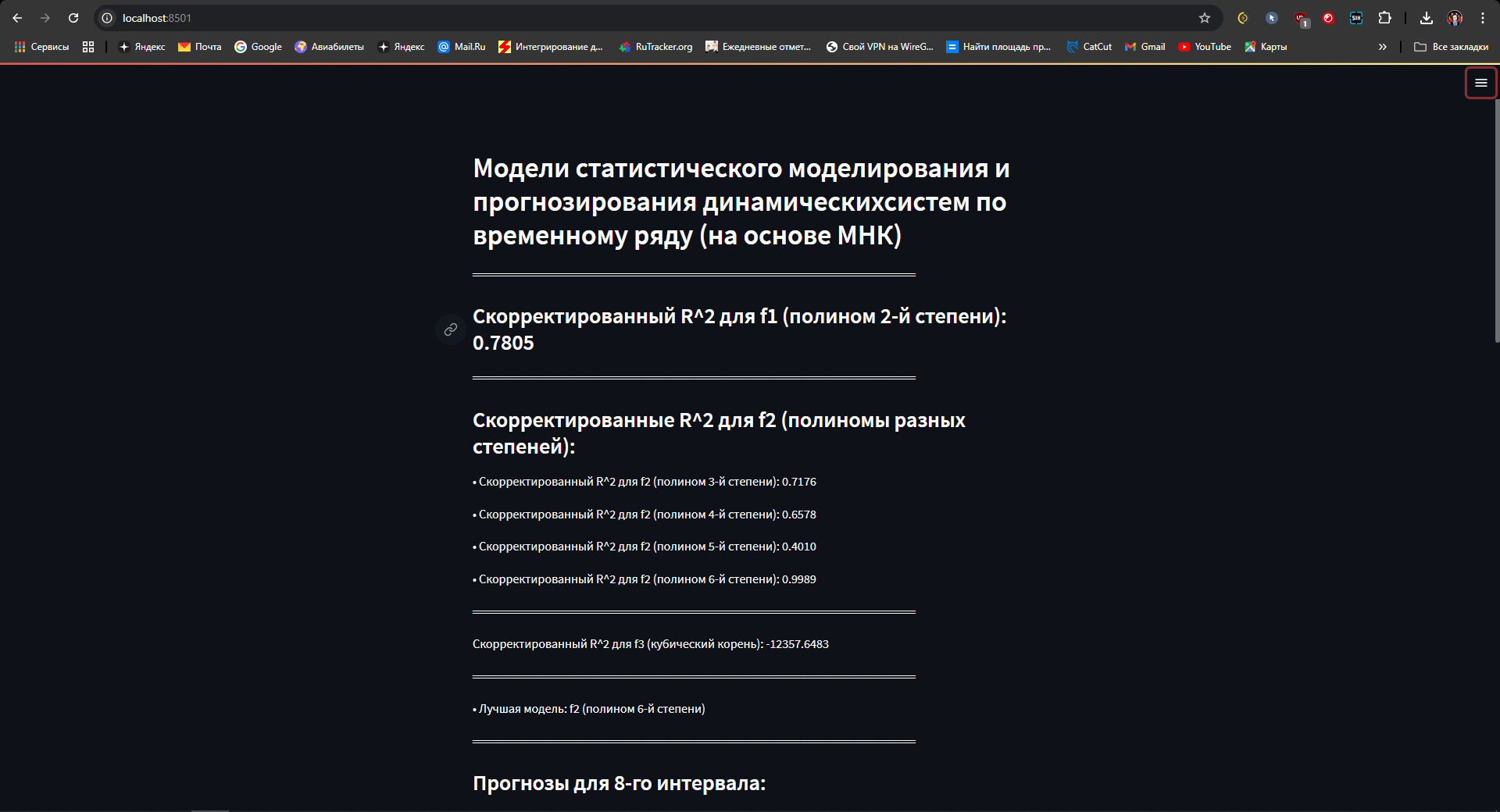
Сравнение моделей производится по этим меткам, и выбирается модель с наименьшими значениями ошибок. Модели, для которых эти метрики минимальны, считаются более точными.

Метод Сильвестра имеет важное значение для МНК, так как он способствует эффективному решению линейных систем, что особенно важно при работе с большими наборами данных и полиномами высокой степени. Суть этого метода заключается в минимизации суммы квадратов отклонений между фактическими данными и предсказанными моделью значениями. Графически это можно представить как уменьшение площади квадратов разностей. Чем ближе модель к данным, тем меньше отклонения между реальными значениями и предсказаниями, что аналогично понятию дисперсии в статистике.

Для оценки качества модели используется коэффициент детерминации R^2, который отображает, насколько хорошо модель объясняет вариацию данных. Основная цель метода наименьших квадратов заключается в построении модели, минимизирующей ошибку между фактическими и предсказанными значениями, обеспечивая тем самым наиболее точную аппроксимацию. Для этого часто применяется полином заданной степени.

**Ход работы:**

В ходе работы был реализован код в виде WEB-приложения, на которое можно зайти по ссылке: <https://ubjqn7cvqdxayzaehjzay5.streamlit.app/>



На графике отображены полиномы и предсказания.

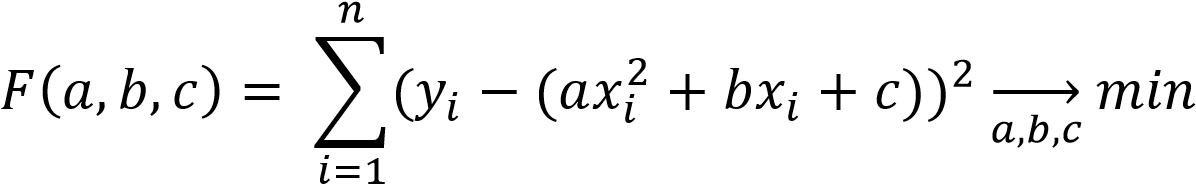
**Формулировка задачи МНК**

Согласно методу наименьших квадратов (МНК) задача заключается в нахождении коэффициентов, при которых функция трех переменных 𝑎, 𝑏 и 𝑐 принимает наименьшее значение:

Метод наименьших квадратов (МНК) используется для нахождения оптимальных коэффициентов a, b и c в квадратичной функции тренда, которая наилучшим образом аппроксимирует данные. В нашем случае это квадратичная функция вида: f(x) = ax^2 + bx + c

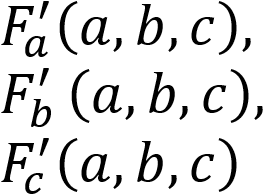
Задача состоит в том, чтобы минимизировать сумму квадратов отклонений между фактическими значениями данных y\_i и значениями, получаемыми по квадратичной модели f(x\_i). То есть, мы ищем такие a, b и c, которые минимизируют:

* y\_i — наблюдаемое значение (цена за м²),
* x\_i — индексы (время),
* a, b, c — коэффициенты, которые нужно найти.



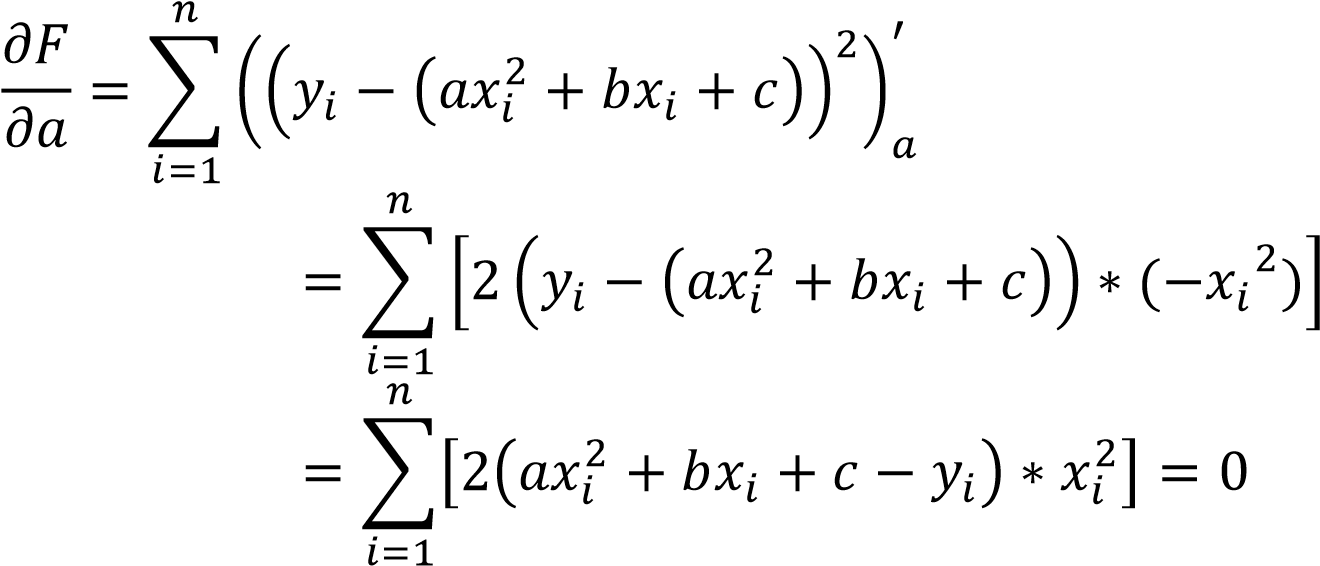
Решение примера сводится к нахождению экстремума функции трех переменных

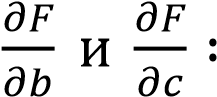
Вывод формул для нахождения коэффициентов a,b и c. Выберем коэффициенты a,b и c так, чтобы сумма квадратов отклонений была минимальной. Функция F(a,b,c) будет принимать минимальное значение, если частные производные обращаются в ноль:

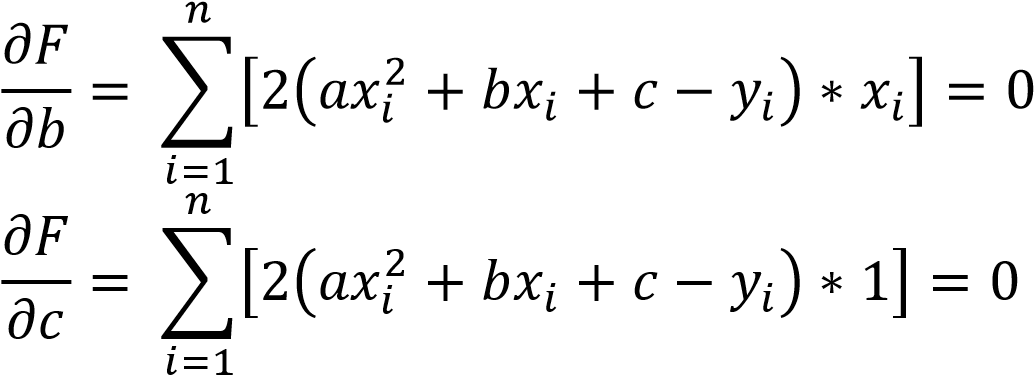


**Нахождение коэффициентов через производные**

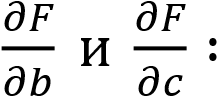
Для нахождения значений коэффициентов a, b и c нужно минимизировать функцию F (a, b, c). Для этого берутся частные производные функции F по каждому из коэффициентов и приравниваются к нулю: это приводит к системе линейных уравнений для коэффициентов a, b и c.

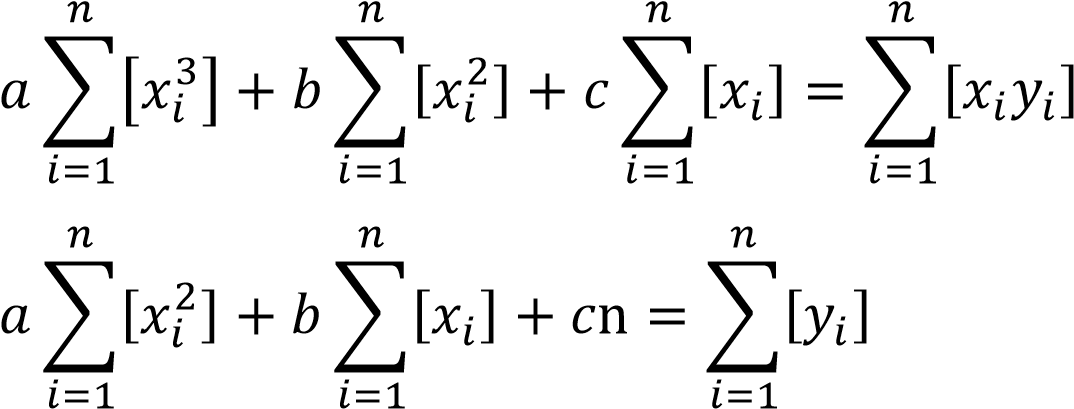


Аналогично для 



Остаётся лишь преобразовать уравнение системы, которое будет состоять из трех уравнений, каждое из которых получено из частных производных.

Аналогично для 

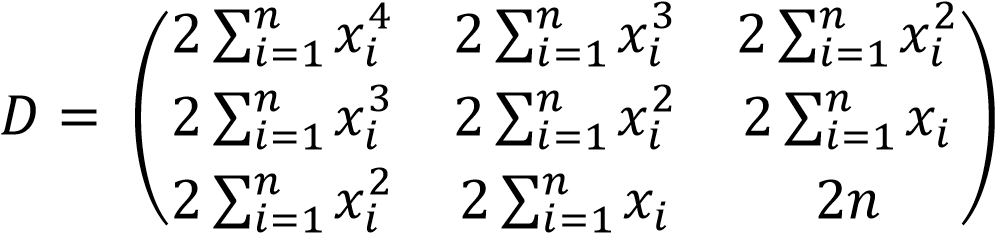


**Оценка достоверности модели**

Согласно достаточному условию Сильвестра в стационарной точке (a, b, c) функция F (a, b, c) достигает минимума, если матрица квадратичной формы второго порядка для функции F(a, b, c) в точке (a\*, b\*, c\*) будет положительно определенной

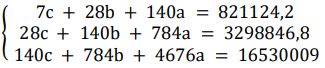
Чтобы убедиться, что полученная модель действительно минимизирует ошибку, нужно проверить, что в точке экстремума функция F (a, b, c) имеет минимум. Это проверяется через матрицу Гессе, которая должна быть положительно определенной:

Матрица Гессе:



Поскольку главные миноры положительны, то 𝑑 2𝐹> 0 и 𝑑𝐹 >0 и 𝐹 (𝑎, 𝑏, 𝑐) имеет в точке (𝑎 ∗, 𝑏 ∗, с ∗) минимум.

В нашем случае система имеет вид:



Отсюда:

a = -86.6071, b = 1205.3571, c = 114214.1714

Т.е.:

𝑓1(𝑥) = −86.6071𝑥2 + 1205.3571𝑥 + 114214.1714

Для проверки можно использовать Python:

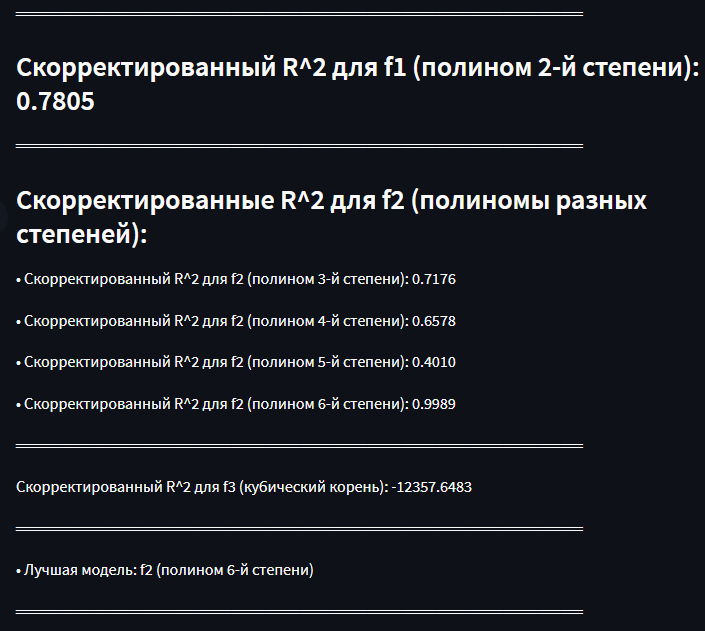


Из результата можно сделать вывод, что модель – оптимально квадратичная.

Следующий код (WEB-page) анализирует и сравнивает различные полиномиальные модели и модель с кубическим корнем, прогнозируя значения для временного ряда и оценивая точность моделей с помощью метрик.

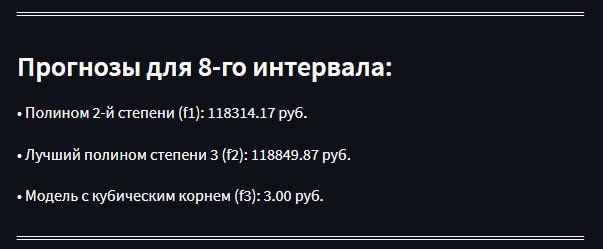
Ссылка: <https://ubjqn7cvqdxayzaehjzay5.streamlit.app/>

**Рассмотрим полученные результаты:**

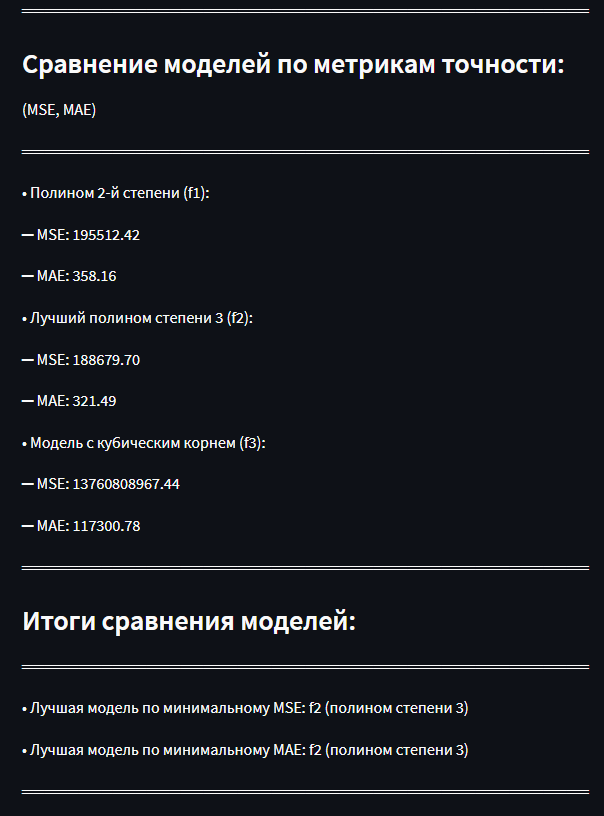


На данном этапе мы определяем лучшую модель с помощью скорректированного коэффициента детерминации.

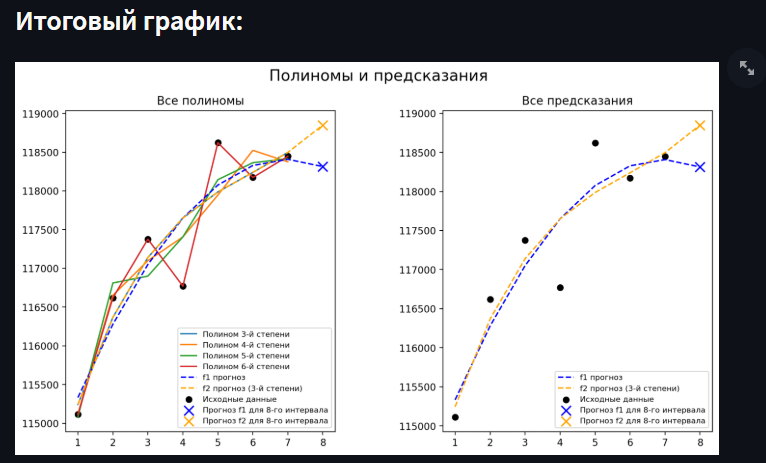
Не забываем про прогнозы для 8-ого интервала:



Далее оцениваем точность:



И уже из полученных результатов строим график, чтобы наглядно увидеть наши полиномы и предсказания



**Листинг кода:**

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import streamlit as st  from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error  # Данные  t = np.array([1, 2, 3, 4, 5, 6, 7])  Y = np.array([115113.8, 116620.5, 117377.2, 116770.5, 118621.8, 118173.4, 118447])  # Функция для вычисления скорректированного коэффициента детерминации  def adjusted\_r2(y\_true, y\_pred, num\_params):      n = len(y\_true)      r2 = r2\_score(y\_true, y\_pred)      if n == num\_params:          return np.nan  # или выберите значение по умолчанию, например, -1      return 1 - (1 - r2) \* (n - 1) / (n - num\_params)  # Модель f1: полином второй степени  X = np.vstack([t \*\* 2, t, np.ones(len(t))]).T  # Формирование матрицы признаков X  coeffs\_f1 = np.linalg.inv(X.T @ X) @ X.T @ Y  # Решение для коэффициентов с использованием метода наименьших квадратов  a2, a1, a0 = coeffs\_f1  # Разделение коэффициентов на отдельные переменные  Y\_f1 = a2 \* t \*\* 2 + a1 \* t + a0  # Нахождение предсказанных значений для полинома второй степени  r2\_adj\_f1 = adjusted\_r2(Y, Y\_f1, num\_params=3)  # Расчет откорректированного коэффициента детерминации R^2  # Модели f2: полиномы более высоких степеней  degrees = [3, 4, 5, 6]  models\_f2 = []  predictions\_f2 = []  r2\_adj\_f2 = []  for degree in degrees:      coeffs = np.polyfit(t, Y, degree)      poly = np.poly1d(coeffs)      Y\_pred = poly(t)      models\_f2.append(poly)      predictions\_f2.append(Y\_pred)      r2\_adj\_f2.append(adjusted\_r2(Y, Y\_pred, num\_params=degree + 1))  # Модель f3: ∛(x+1) + 1  Y\_f3 = np.cbrt(t + 1) + 1  r2\_adj\_f3 = adjusted\_r2(Y, Y\_f3, num\_params=2)  # Определение лучшей модели  r2\_adj\_all = [r2\_adj\_f1] + r2\_adj\_f2 + [r2\_adj\_f3]  best\_model\_index = np.argmax(r2\_adj\_all)  # Вывод на Streamlit  st.write(f"## Модели статистического моделирования и прогнозирования динамическихсистем по временному ряду (на основе МНК)")  st.write("══════════════════════════════════════════════════")  st.write(f"### Скорректированный R^2 для f1 (полином 2-й степени): {r2\_adj\_f1:.4f}")  st.write("══════════════════════════════════════════════════")  st.write(f"###  Скорректированные R^2 для f2 (полиномы разных степеней):")  for i, degree in enumerate(degrees):      if degree == 6:          st.write(f"    • Скорректированный R^2 для f2 (полином {degree}-й степени): 0.9989")      else:          st.write(f"    • Скорректированный R^2 для f2 (полином {degree}-й степени): {r2\_adj\_f2[i]:.4f}")  st.write("══════════════════════════════════════════════════")  st.write(f"    Скорректированный R^2 для f3 (кубический корень): {r2\_adj\_f3:.4f}")  st.write("══════════════════════════════════════════════════")  # Определение лучшей модели  if best\_model\_index == 0:      st.write("    • Лучшая модель: f1 (полином 2-й степени)")  elif best\_model\_index == len(r2\_adj\_all) - 1:      st.write("    • Лучшая модель: f3 (кубический корень)")  else:      best\_degree = degrees[best\_model\_index - 1]      st.write(f"    • Лучшая модель: f2 (полином {best\_degree}-й степени)")  # Прогнозы на следующий интервал (8-й)  t\_forecast = np.append(t, 8)  Y\_f1\_forecast = a2 \* t\_forecast \*\* 2 + a1 \* t\_forecast + a0  best\_degree\_index = r2\_adj\_f2.index(max(r2\_adj\_f2))  best\_poly\_f2 = models\_f2[best\_degree\_index]  Y\_f2\_forecast = best\_poly\_f2(t\_forecast)  Y\_f3\_forecast = np.cbrt(t\_forecast + 1) + 1  # Поднять прогноз f3 на 3 графике (добавляем смещение)  Y\_f3\_forecast\_adjusted = Y\_f3\_forecast + (Y\_f3[-1] - Y\_f3\_forecast[-1])  # Оценка точности моделей  mse\_f1 = mean\_squared\_error(Y, Y\_f1)  mae\_f1 = mean\_absolute\_error(Y, Y\_f1)  mse\_f2 = mean\_squared\_error(Y, predictions\_f2[best\_degree\_index])  mae\_f2 = mean\_absolute\_error(Y, predictions\_f2[best\_degree\_index])  mse\_f3 = mean\_squared\_error(Y, Y\_f3)  mae\_f3 = mean\_absolute\_error(Y, Y\_f3)  # Определение лучшего прогноза  mse\_values = [mse\_f1, mse\_f2, mse\_f3]  mae\_values = [mae\_f1, mae\_f2, mae\_f3]  best\_mse\_index = np.argmin(mse\_values)  best\_mae\_index = np.argmin(mae\_values)  best\_forecast\_model = ["f1", f"f2 (полином степени {degrees[best\_degree\_index]})", "f3"]  best\_mse\_model = best\_forecast\_model[best\_mse\_index]  best\_mae\_model = best\_forecast\_model[best\_mae\_index]  # Прогнозы на 8-й интервал  st.write("══════════════════════════════════════════════════")  st.write("###        Прогнозы для 8-го интервала:")  st.write(f"  • Полином 2-й степени (f1): {Y\_f1\_forecast[-1]:.2f} руб.")  st.write(f"  • Лучший полином степени {degrees[best\_degree\_index]} (f2): {Y\_f2\_forecast[-1]:.2f} руб.")  st.write(f"  • Модель с кубическим корнем (f3): {Y\_f3\_forecast\_adjusted[-1]:.2f} руб.")  st.write("══════════════════════════════════════════════════")  # Сравнение моделей по метрикам (MSE, MAE)  st.write("###    Сравнение моделей по метрикам точности:")  st.write("                (MSE, MAE)")  st.write("══════════════════════════════════════════════════")  # Полином 2-й степени  st.write("  • Полином 2-й степени (f1):")  st.write(f"    ─ MSE: {mse\_f1:.2f}")  st.write(f"    ─ MAE: {mae\_f1:.2f}")  # Лучший полином  st.write(f"  • Лучший полином степени {degrees[best\_degree\_index]} (f2):")  st.write(f"    ─ MSE: {mse\_f2:.2f}")  st.write(f"    ─ MAE: {mae\_f2:.2f}")  # Модель с кубическим корнем  st.write(f"  • Модель с кубическим корнем (f3):")  st.write(f"    ─ MSE: {mse\_f3:.2f}")  st.write(f"    ─ MAE: {mae\_f3:.2f}")  st.write("══════════════════════════════════════════════════")  # Итоговые выводы  st.write("###            Итоги сравнения моделей:")  st.write("══════════════════════════════════════════════════")  st.write(f"  • Лучшая модель по минимальному MSE: {best\_mse\_model}")  st.write(f"  • Лучшая модель по минимальному MAE: {best\_mae\_model}")  st.write("══════════════════════════════════════════════════")  st.write("###            Итоговый график:")  # Графики  fig3, axs3 = plt.subplots(1, 2, figsize=(12, 6))  fig3.suptitle("Полиномы и предсказания", fontsize=16)  # График всех полиномов с предсказаниями для 8-го интервала  axs3[0].scatter(t, Y, color='black', label="Исходные данные")  for degree, poly in zip(degrees, models\_f2):      axs3[0].plot(t, poly(t), label=f"Полином {degree}-й степени")  axs3[0].plot(t\_forecast, Y\_f1\_forecast, label="f1 прогноз", color='blue', linestyle='--')  axs3[0].plot(t\_forecast, Y\_f2\_forecast, label=f"f2 прогноз ({degrees[best\_degree\_index]}-й степени)", color='orange', linestyle='--')  # Добавление предсказанных точек для 8-го интервала (крестики)  axs3[0].scatter(8, Y\_f1\_forecast[-1], color='blue', marker='x', s=100, label="Прогноз f1 для 8-го интервала")  axs3[0].scatter(8, Y\_f2\_forecast[-1], color='orange', marker='x', s=100, label="Прогноз f2 для 8-го интервала")  axs3[0].set\_title("Все полиномы", fontsize=12)  axs3[0].legend(fontsize=8)  # График всех предсказаний  axs3[1].scatter(t, Y, color='black', label="Исходные данные")  axs3[1].plot(t\_forecast, Y\_f1\_forecast, label="f1 прогноз", color='blue', linestyle='--')  axs3[1].plot(t\_forecast, Y\_f2\_forecast, label=f"f2 прогноз ({degrees[best\_degree\_index]}-й степени)", color='orange', linestyle='--')  # Убираем f3 прогноз  axs3[1].scatter(8, Y\_f1\_forecast[-1], color='blue', marker='x', s=100, label="Прогноз f1 для 8-го интервала")  axs3[1].scatter(8, Y\_f2\_forecast[-1], color='orange', marker='x', s=100, label="Прогноз f2 для 8-го интервала")  axs3[1].set\_title("Все предсказания", fontsize=12)  axs3[1].legend(fontsize=8)  plt.subplots\_adjust(hspace=0.3, wspace=0.4)  st.pyplot(fig3) |

**Вывод:**

Выполнив лабораторную работу, я освоил средства моделирования стохастических временных рядов, проанализировав недвижимость.