Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Уфимский государственный авиационный технический университет»

Кафедра вычислительной математики и кибернетики

Лабораторная работа №4

по дисциплине: «Компьютерная обработка экспериментальных данных»

«Прогнозирование временных рядов. Экспоненциальная средняя как предиктор»

Выполнили:

студенты группы МО-417

Ярыгин А.Э.

Алимгафаров А.Р.

Проверила:

Харисова Э. А.

Уфа 2022

**Цель:**

Приобрести навыки прогнозирования методом многократного экспоненциального сглаживания.

**Задачи**

1. Выполнить задание для самостоятельной работы в соответствии с настоящим руководством по выполнению лабораторной работы;
2. Оформить отчет о выполнении лабораторной работы в соответствии с требованиями к его оформлению.

**Ход работы**

1. В качестве исходных данных для прогнозирования возьмем таблицу, в которой записаны данные продаж на конец недели в течение 30-ти недель. Данные будем хранить в excel-файле, данные о первых 10-ти неделях можно рассмотреть на рисунке 1:

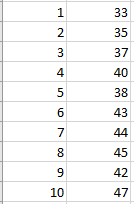
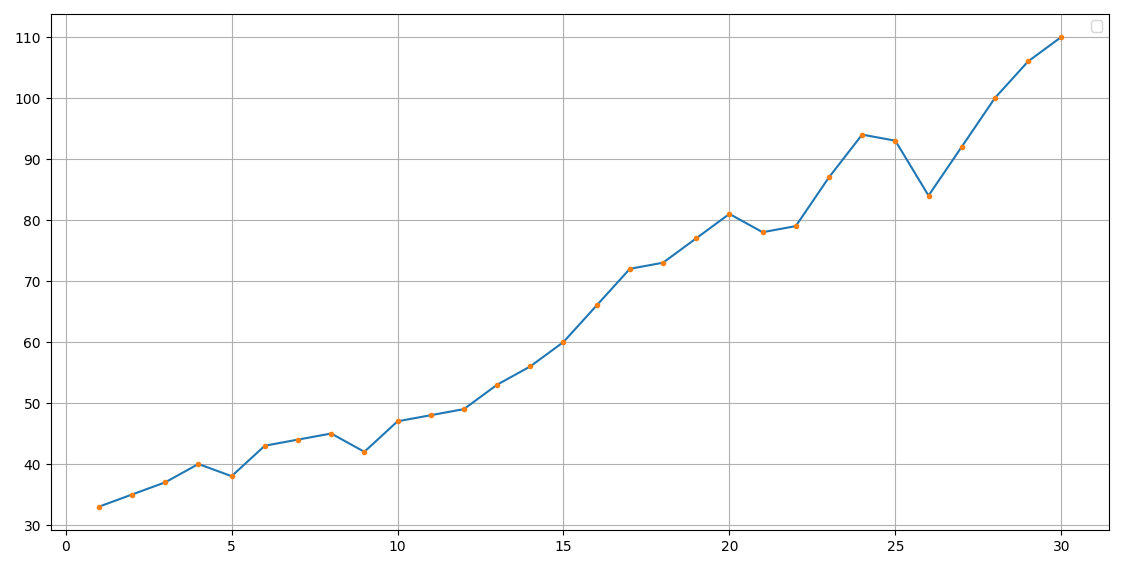


Рисунок 1 - Исходные данные

Оценки мат. ожидания и дисперсии выборки:



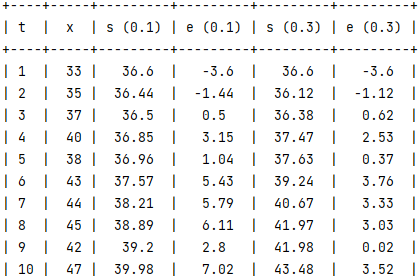
Построим график полученных значений временного ряда (по оси абсцисс – время, по оси ординат – количество продаж) (рисунок 2):

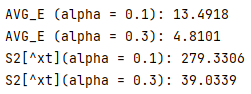
  
Рисунок 2 – График данных о продажах за 30 недель

1. Используем постоянную модель (n = 0) для прогнозирования значений временного ряда при α = 0,1 и α = 0,3. будем считать прогнозом на момент времени *t+1*. Начальное значение найдем как среднее арифметическое пяти первых значений временного ряда. На каждом шаге сравним предсказанное и действительное значения, выписывая ошибку предсказания Δ , *t=1, …, N*. Вычислим дисперсию ошибки предсказания по данным эксперимента.

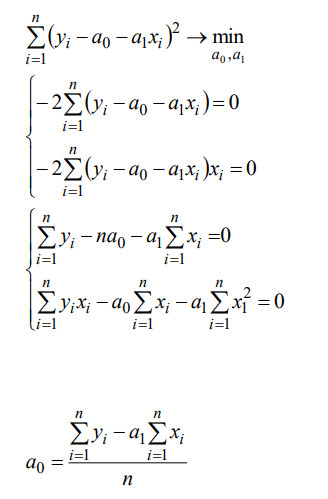
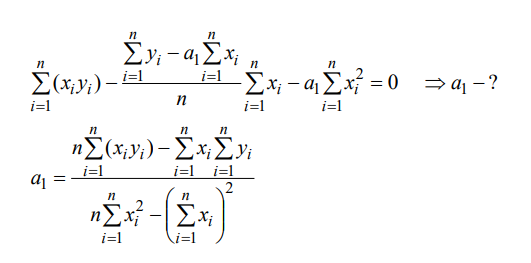
Выведем на экран таблицу с фактическими и предсказанными данными (для α = 0,1 и α = 0,3 с соответствующими столбцами ошибок), а также выведем среднее ошибок и дисперсию (рисунки 3 и 4):

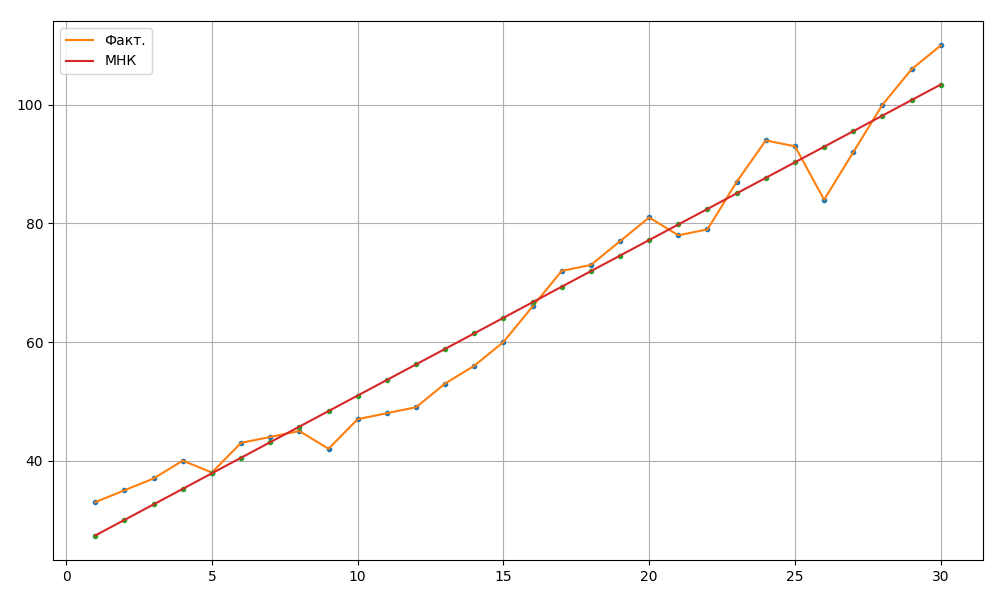
,

  
Рисунок 3 – Часть таблицы прогнозирования постоянной модели

  
Рисунок 4 – Средние и дисперсии ошибок постоянной модели

1. Используем метод наименьших квадратов и построим аппроксимирующую прямую (рисунок 5):



  
Рисунок 5 – График фактических данных и аппроксимирующей прямой

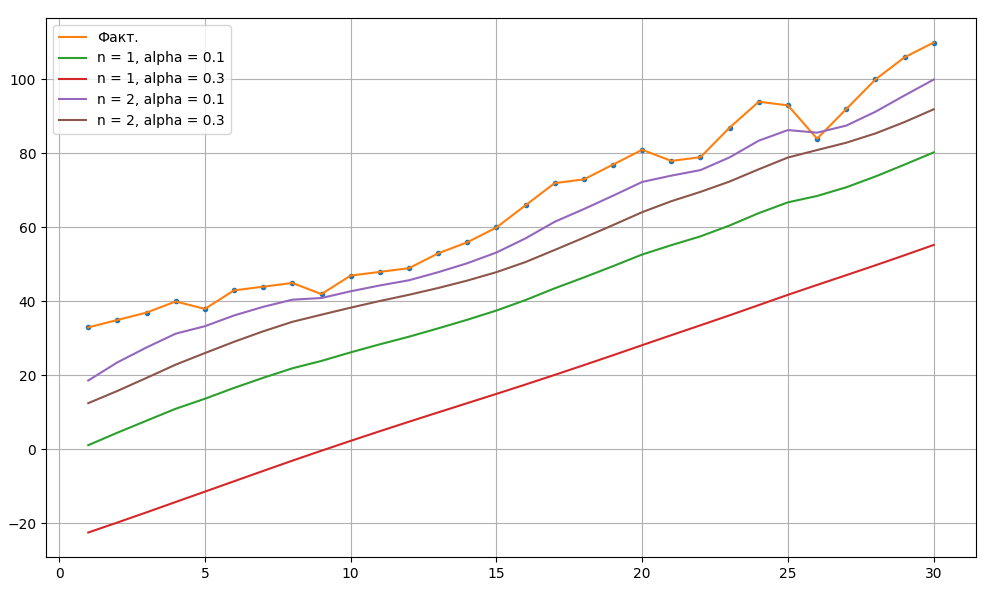
Программно получили уравнение прямой МНК (рисунок 6):

  
Рисунок 6 – Уравнение МНК

1. Используем операторы экспоненциального сглаживания первого и второго порядков и получим сглаженные значения и , *t=1, …, N* при значениях постоянной сглаживания α = 0,1 и α = 0,3. Изобразим все четыре графика на рисунке 7:

– экспоненциальное сглаживание 1-го порядка

– экспоненциальное сглаживание 2-го порядка

  
Рисунок 7 – Графики сглаженных экспоненциальных средних

Из рисунка видно, что лучше всего исходные данные аппроксимирует линия второго экспоненциального сглаживания при α = 0,1.

1. Используем линейную модель (n = 1) для прогнозирования значений временного ряда х с интервалом упреждения τ =∆t, что равносильно   
   m = 1 при α = 0,1 и α = 0,3. На каждом шаге сравним предсказанное и действительное значения, выписывая ошибку предсказания Δ , *t=1, …, N*. Вычислим дисперсию ошибки предсказания по данным эксперимента.

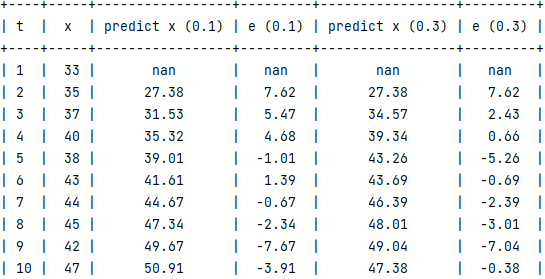
Выведем на экран таблицу с фактическими и предсказанными данными (для α = 0,1 и α = 0,3 с соответствующими столбцами ошибок), а также выведем среднее ошибок и дисперсию (рисунки 8 и 9):

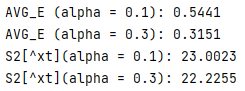
– предсказанное значение,

- оценки коэффициентов линейной модели,

– экспоненциальное сглаживание 1-го порядка

– экспоненциальное сглаживание 2-го порядка

  
Рисунок 8 – Часть таблицы прогнозирования линейной модели (при m = 1)

  
Рисунок 9 – Средние и дисперсии ошибок линейной модели (при m = 1)

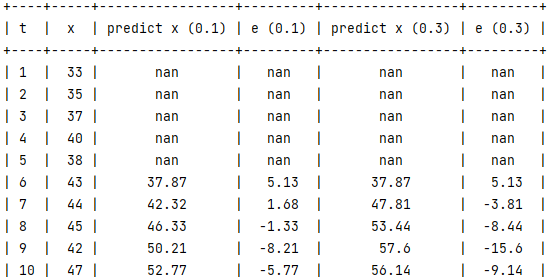
1. Аналогичным образом выполним предыдущий пункт при m = 5 и выведем полученные данные на экран (рисунки 10 и 11):

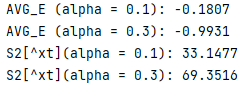
– предсказанное значение,

– оценки коэффициентов линейной модели,

– экспоненциальное сглаживание 1-го порядка

– экспоненциальное сглаживание 2-го порядка

  
Рисунок 10 – Часть таблицы прогнозирования линейной модели (при m = 5)

  
Рисунок 11 – Средние и дисперсии ошибок линейной модели (при m = 1)

1. По алгоритму выбора моделей получили средние значения разностей нескольких порядков и вывели в столбик N – порядок полинома, delta – среднее разностей соответствующего порядка (рисунок 12):

– разности первого порядка,

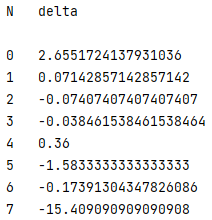
– разности первого порядка,

…

– разности первого порядка,

Если среднее разностей *(n – 1)-*го порядка отлично от нуля, а среднее разностей *n-*го порядка равно нулю, то моделью служит полином

*(n – 1)-*й степени, причем среднее значение разности *(n – 1)-*го порядка может служить начальной оценкой коэффициента при старшем члене полинома.

  
Рисунок 12 – Вывод средних разностей восьми первых порядков

Исходя из полученных значений видим, что при N = 3 вычисляем ближайшее к нулю значение среднего, следовательно, делаем вывод о том, что приемлемым для описания начальной выборки является полином 2-го порядка.

Стоит также отметить, что из полученных на предыдущих этапах моделей лучше всего по средним значениям ошибок и дисперсии себя показала линейная модель (n = 1) при α = 0,1.

**Вывод**

В ходе выполнения лабораторной работы мы реализовали программу для выполнения прогнозирования временного ряда методом многократного экспоненциального сглаживания.

Приложение А

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from prettytable import PrettyTable

def fact(n):

return 0 if n < 1 else n \* fact(n - 1)

# пункт 1

print('STEP 1:')

data = pd.read\_excel('data.xlsx')

X = np.array(data[1])

T = list(range(1, len(X) + 1))

avg = sum(X) / len(X)

var = sum([(x - avg) \*\* 2 for x in X]) / len(X)

print('M[X]:\t', round(avg, 2))

print('D[X]:\t', round(var, 2))

print()

plt.figure(figsize=(12, 7))

plt.plot(T, X)

plt.plot(T, X, '.')

plt.legend()

plt.grid(True)

plt.show()

# пункт 2

print('STEP 2:')

alpha1 = 0.1

beta1 = 1 - alpha1

alpha2 = 0.3

beta2 = 1 - alpha2

S1 = [np.mean(X[:5])]

S2 = [np.mean(X[:5])]

e1 = [X[0] - S1[0]]

e2 = [X[0] - S2[0]]

table = PrettyTable()

table.field\_names = ['t', 'x', f's ({alpha1})', f'e ({alpha1})', f's ({alpha2})', f'e ({alpha2})']

table.add\_row([1, X[0], round(S1[0], 2), round(e1[0], 2), round(S2[0], 2), round(e2[0], 2)])

for t in range(1, len(X)):

S1.append(alpha1 \* X[t] + beta1 \* S1[t - 1])

S2.append(alpha2 \* X[t] + beta2 \* S2[t - 1])

e1.append(X[t] - S1[-1])

e2.append(X[t] - S2[-1])

table.add\_row([t + 1, X[t], round(S1[t], 2), round(e1[t], 2), round(S2[t], 2), round(e2[t], 2)])

print(table)

print(f'AVG\_E (alpha = {alpha1}): {round(np.mean(e1), 4)}')

print(f'AVG\_E (alpha = {alpha2}): {round(np.mean(e2), 4)}')

var\_e1 = sum([e \*\* 2 for e in e1]) / (len(X) - 1)

var\_e2 = sum([e \*\* 2 for e in e2]) / (len(X) - 1)

print(f'S2[^xt](alpha = {alpha1}): {round(var\_e1, 4)}')

print(f'S2[^xt](alpha = {alpha2}): {round(var\_e2, 4)}')

print()

# пункт 3 (МНК)

print('STEP 3:')

N = len(X)

a1 = (N \* sum([X[i] \* T[i] for i in range(N)]) - sum(X) \* sum(T)) / (N \* sum([t \*\* 2 for t in T]) - sum(T) \*\* 2)

a0 = (sum(X) - a1 \* sum(T)) / N

str\_sign = '+' if a1 >= 0 else '-'

print(f'x = {round(a0, 2)} {str\_sign} {round(a1, 2)} \* t')

print()

Y = [(a0 + a1 \* t) for t in T]

plt.figure(figsize=(12, 7))

plt.plot(T, X, '.')

plt.plot(T, X, label='Факт.')

plt.plot(T, Y, '.')

plt.plot(T, Y, label='МНК')

plt.legend()

plt.grid(True)

plt.show()

# пункт 4 (эксп. сглаживание)

print('STEP 4:')

S11 = [a0 - (beta1 / alpha1) \* a1]

S12 = [a0 - (2 \* beta1 / alpha1) \* a1]

S21 = [a0 - (beta2 / alpha2) \* a1]

S22 = [a0 - (2 \* beta2 / alpha2) \* a1]

for t in range(1, len(X)):

S11.append(alpha1 \* X[t] + beta1 \* S11[t - 1])

S12.append(alpha1 \* S11[t] + beta1 \* S12[t - 1])

S21.append(alpha2 \* X[t] + beta2 \* S21[t - 1])

S22.append(alpha2 \* S21[t] + beta2 \* S22[t - 1])

plt.figure(figsize=(12, 7))

plt.plot(T, X, '.')

plt.plot(T, X, label='Факт.')

plt.plot(T, S11, label=f'n = 1, alpha = {alpha1}')

plt.plot(T, S12, label=f'n = 1, alpha = {alpha2}')

plt.plot(T, S21, label=f'n = 2, alpha = {alpha1}')

plt.plot(T, S22, label=f'n = 2, alpha = {alpha2}')

plt.legend()

plt.grid(True)

plt.show()

print()

# пункт 5 (линейная модель, прогноз при m = 1)

print('STEP 5:')

A01 = [2 \* S11[0] - S12[0]]

A11 = [(S11[0] - S12[0]) \* alpha1 / beta1]

A02 = [2 \* S21[0] - S22[0]]

A12 = [(S21[0] - S22[0]) \* alpha2 / beta2]

m = 1

e1 = [0 for \_ in range(m)]

e2 = [0 for \_ in range(m)]

table = PrettyTable()

table.field\_names = ['t', 'x', f'predict x ({alpha1})', f'e ({alpha1})', f'predict x ({alpha2})', f'e ({alpha2})']

for i in range(m):

table.add\_row([i + 1, X[i], np.NaN, np.NaN, np.NaN, np.NaN])

for t in range(m, len(X)):

A01.append(2 \* S11[t] - S12[t])

A11.append((S11[t] - S12[t]) \* alpha1 / beta1)

A02.append(2 \* S21[t] - S22[t])

A12.append((S21[t] - S22[t]) \* alpha2 / beta2)

\_x1 = A01[t - m] + A11[t - m] \* m

\_x2 = A02[t - m] + A12[t - m] \* m

e1.append(X[t] - \_x1)

e2.append(X[t] - \_x2)

table.add\_row([t + 1, X[t], round(\_x1, 2), round(X[t] - \_x1, 2), round(\_x2, 2), round(X[t] - \_x2, 2)])

print(table)

print(f'AVG\_E (alpha = {alpha1}): {round(np.mean(e1[m:]), 4)}')

print(f'AVG\_E (alpha = {alpha2}): {round(np.mean(e2[m:]), 4)}')

var\_e1 = sum([e \*\* 2 for e in e1[m:]]) / (len(e1[m:]) - 2)

var\_e2 = sum([e \*\* 2 for e in e2[m:]]) / (len(e2[m:]) - 2)

print(f'S2[^xt](alpha = {alpha1}): {round(var\_e1, 4)}')

print(f'S2[^xt](alpha = {alpha2}): {round(var\_e2, 4)}')

print()

# пункт 6 (линейная модель, прогноз при m = 5)

print('STEP 6:')

m = 5

e1 = [0 for \_ in range(m)]

e2 = [0 for \_ in range(m)]

table = PrettyTable()

table.field\_names = ['t', 'x', f'predict x ({alpha1})', f'e ({alpha1})', f'predict x ({alpha2})', f'e ({alpha2})']

for i in range(m):

table.add\_row([i + 1, X[i], np.NaN, np.NaN, np.NaN, np.NaN])

for t in range(m, len(X)):

A01.append(2 \* S11[t] - S12[t])

A11.append((S11[t] - S12[t]) \* alpha1 / beta1)

A02.append(2 \* S21[t] - S22[t])

A12.append((S21[t] - S22[t]) \* alpha2 / beta2)

\_x1 = A01[t - m] + A11[t - m] \* m

\_x2 = A02[t - m] + A12[t - m] \* m

e1.append(X[t] - \_x1)

e2.append(X[t] - \_x2)

table.add\_row([t + 1, X[t], round(\_x1, 2), round(X[t] - \_x1, 2), round(\_x2, 2), round(X[t] - \_x2, 2)])

print(table)

print(f'AVG\_E (alpha = {alpha1}): {round(np.mean(e1[m:]), 4)}')

print(f'AVG\_E (alpha = {alpha2}): {round(np.mean(e2[m:]), 4)}')

var\_e1 = sum([e \*\* 2 for e in e1[m:]]) / (len(e1[m:]) - 2)

var\_e2 = sum([e \*\* 2 for e in e2[m:]]) / (len(e2[m:]) - 2)

print(f'S2[^xt](alpha = {alpha1}): {round(var\_e1, 4)}')

print(f'S2[^xt](alpha = {alpha2}): {round(var\_e2, 4)}')

print()

# пункт 7 (выбор модели и порядка полинома)

print('STEP 7:')

print('N\tdelta\n')

dx = [(X[t] - X[t - 1]) for t in range(1, len(X))]

dxs = [(np.mean(dx), dx)]

print(f'0\t{dxs[-1][0]}')

for i in range(7):

x = dxs[-1][1]

dx = [(x[t] - x[t - 1]) for t in range(1, len(x))]

dxs.append((np.mean(dx), dx))

print(f'{i + 1}\t{dxs[-1][0]}')