Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №1**

по дисциплине: «Компьютерные системы моделирования»

Тема: «Построение аналитической модели по текстовому описанию задачи и по результатам экспериментов»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС-22.06  Аванесян Г.Р. |
| Проверила:  старший преподаватель кафедры ПО  Корнеева Е.И. |

Тверь 2025

**Теория. Что такое моделирование? Как связано моделирование с понятием «система»?**

Моделированием называется замещение одного объекта, называемого системой, другим объектом, называемым моделью, и проведение экспериментов с моделью (или на модели), исследование свойств модели, опираясь на результаты экспериментов с целью получения информации о системе. Связь моделирования с понятием «система» заключается в том, что любая система состоит из взаимосвязанных компонентов, которые взаимодействуют для достижения определенной цели.

**Задача 1.**

Построение аналитической модели по вербальному (текстовому) описанию

**Постановка задачи.**

Построить математическую модель задачи, условие которой записано ниже. Найти оптимальное значение целевой переменной с помощью библиотек python 3.\*: pulp, cvxopt или scipy.

Изображение выглядит как текст, чек, снимок экрана, Параллельный

Автоматически созданное описание

**Построение математической модели**

**Минимально допустимые значения товарооборота по j-q группе, ед.**

Целевая функция: 1200\*x1+1000\*x2+1500\*x3+1200\*x4

**Ограничения:**

18x1 + 26x2 + 16x3 + 10x4 <= 110000

150 x1+ 140 x2 + 50 x3 + 80 x4 <=950000

170 x1 + 230 x2 + 280 x3 + 120 x4 <= 1200000

31 x1 + 42 x2 + 30 x3 + 20 x4 <= 180000

200x1 + 150x2 + 170x3 + 50x4 >= 750000

**Переменные:**  
x1, x2, x3, x4 ≥ 0, целочисленные.

**Решение в MS Excel**

Изображение выглядит как текст, число, снимок экрана

Автоматически созданное описание

Минимально допустимые значение товарооборота: 4500000

**Прибыль в расчете на единицу товарооборота j-й группы, ден. Ед.**

120\*x1+50\*x2+30\*x3+100\*x4

**Ограничения:**

18x1 + 26x2 + 16x3 + 10x4 <= 110000

150 x1+ 140 x2 + 50 x3 + 80 x4 <=950000

170 x1 + 230 x2 + 280 x3 + 120 x4 <= 1200000

31 x1 + 42 x2 + 30 x3 + 20 x4 <= 180000

200x1 + 150x2 + 170x3 + 50x4 >= 750000

1200\*x1+1000\*x2+1500\*x3+1200\*x4 >= 4500000

**Решение в MS Excel**

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Прибыль в расчете на единицу товарооборота j-й группы, ден. Ед. = 814285,7143

**Решение на python:**

from pulp import LpVariable, LpProblem, LpMaximize, value, LpMinimize

import time

start = time.time()

x1 = LpVariable("x1", lowBound=0)

x2 = LpVariable("x2", lowBound=0)

x3 = LpVariable("x3", lowBound=0)

x4 = LpVariable("x4", lowBound=0)

problem = LpProblem('0', LpMinimize)

problem += 1200\*x1 + 1000\*x2 + 1500\*x3 + 1200\*x4, "Функция цели"

problem += 18\*x1 + 26\*x2 + 16\* x3 + 10\*x4 <= 110000, "1"

problem += 150\*x1 + 140\*x2 + 50\* x3 + 80\*x4 <= 950000, "2"

problem += 170\*x1 + 230\*x2 + 280\*x3 + 120\*x4 <= 1200000, "3"

problem += 31\*x1 + 42\*x2 + 30\*x3 + 20\*x4 <= 180000, "4"

problem += 200\*x1 + 150\*x2 + 170\*x3 + 50\*x4 >= 750000, "5"

problem.solve()

print("Результат:")

for variable in problem.variables():

    print (variable.name, "=", round(variable.varValue, 2))

print("Объем ресурса:")

print(round(value(problem.objective),2))

stop = time.time()

print ("Время :")

print(stop - start)

print ("первое :")

start = time.time()

t1 = LpVariable("t1", lowBound=0)

t2 = LpVariable("t2", lowBound=0)

t3 = LpVariable("t3", lowBound=0)

t4 = LpVariable("t4", lowBound=0)

secondproblem = LpProblem('0', LpMaximize)

secondproblem += 120\*t1 + 50\*t2 + 30\*t3 + 100\*t4, "Функция цели"

secondproblem += 18\*t1 + 26\*t2 + 16\* t3 + 10\*t4 <= 110000, "1"

secondproblem += 150\*t1 + 140\*t2 + 50\* t3 + 80\*t4 <= 950000, "2"

secondproblem += 170\*t1 + 230\*t2 + 280\*t3 + 120\*t4 <= 1200000, "3"

secondproblem += 31\*t1 + 42\*t2 + 30\*t3 + 20\*t4 <= 180000, "4"

secondproblem += 200\*t1 + 150\*t2 + 170\*t3 + 50\*t4 >= 750000, "5"

secondproblem += 1200\*t1 + 1000\*t2 + 1500\*t3 + 1200\*t4 >= round(value(problem.objective),2), "6"

secondproblem.solve()

print("Результат:")

for var in secondproblem.variables():

    print (var.name, "=", round(var.varValue, 2))

print("Объем ресурса:")

print(round(value(secondproblem.objective),2))

stop = time.time()

print ("Время :")

print(stop - start)

print ("второе :")

**Результат**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Вывод:**

Поставленная задача успешно решена двумя способами: с использованием Excel и Python. Результаты совпадают, что подтверждает корректность модели. Использование Python обеспечивает автоматизацию и масштабируемость для решения более сложных задач.

**Задача 2**

По данным сайта https://ru.investing.com/ даны изменения котировок на

15.03.2023. Данные варианта представлены в таблице ниже.

Создать объект Figure c двумя графиками (точечные, строятся по данным

таблицы). На каждом графике построить возможные линии тренда, например

ниже.. В заголовке графика добавить формулу линии тренда и коэффициент

детерминации (𝑅^2).

Типы линий тренда (аппроксимирующих функций):

• линейная модель,

• полиномиальная модель,

• степенная модель,

• логарифмическая модель

Сделать вывод по коэффициенту 𝑅

2 о том, подходит ли аппроксимирующая

модель данным. Сравнить две линии тренда и сделать вывод

При построении полиномиальной модели рассмотреть степень полинома с

максимальным (𝑅^2).

Изображение выглядит как текст, число, снимок экрана, чек

Автоматически созданное описание

**Решение в Excel**

В Excel были построены линии тренда с использованием встроенных функций:

* Линейная, полиномиальная, степенная и логарифмическая аппроксимации.
* Для каждой модели отображены уравнение тренда и R^2.

Скриншоты графиков и уравнений:

Изображение выглядит как текст, число, Шрифт, Параллельный

Автоматически созданное описание

**Решение на python**

В Python для анализа использовались библиотеки numpy, scikit-learn, matplotlib и функции:

* Линейная регрессия (LinearRegression).
* Полиномиальная регрессия с использованием степени k = 6.
* Степенная модель с функцией y=a⋅x.
* Логарифмическая модель y=a⋅ln(x)+b.

**Код:**

import datetime

import matplotlib.pyplot as plt # импортируем построитель графиков из библиотеки matplotlib

import numpy as np # импортируем библиотеку numpy для работы с массивами numpy

from sklearn.metrics import r2\_score # функция для расчёта критерия r^2

data = [

    "15.03.2023  75,9346",

    "14.03.2023  75,32",

    "13.03.2023  75",

    "10.03.2023  76,1",

    "09.03.2023  75,845",

    "07.03.2023  76,0225",

    "06.03.2023  75,445",

    "03.03.2023  75,6325",

    "02.03.2023  75,3175",

    "01.03.2023  75,15",

    "28.02.2023  74,97",

    "27.02.2023  74,6025",

    "24.02.2023  76,13",

    "22.02.2023  75",

    "21.02.2023  75,0475",

    "20.02.2023  74,5",

    "17.02.2023  74,29",

    "16.02.2023  74,855",

    "15.02.2023  74,5725"

]

x = []

y = []

for entry in data:

    date\_str, value\_str = entry.split()

    date = datetime.datetime.strptime(date\_str, "%d.%m.%Y")

    date\_int = int(date.timestamp())

    value\_float = float(value\_str.replace(',', '.'))

    x.append(date\_int)

    y.append(value\_float)

numpy\_x = np.array(x)

numpy\_y = np.array(y)

# Линии тренда

# линейный (автоматическое создание)

set\_line\_by\_data = np.polyfit(numpy\_x, numpy\_y, 1) # полином первой степени

linear\_trend = np.poly1d(set\_line\_by\_data) # снижение размерности до одномерного массива

print("{0}x + {1}".format(\*set\_line\_by\_data)) # формула

set\_polinom\_by\_data = np.polyfit(numpy\_x, numpy\_y, 2) # работа с полиномом 2 степени

polinom\_trend = np.poly1d(set\_polinom\_by\_data) # Рассчитать значение полинома в точках x

polinom\_title = "${0}x^2 + {1}x + {2}$".format(\*set\_polinom\_by\_data)

print("${0}x^2 + {1}x + {2}$".format(\*set\_polinom\_by\_data)) # формула

# логарифмический

set\_log\_by\_data = np.polyfit(np.log(numpy\_x), numpy\_y, 1) # работа с полиномом 1 степени + логарифмирование x

log\_trend = [set\_log\_by\_data[0]\*np.log(x) + set\_log\_by\_data[1] for x in numpy\_x] # создание одномерного массива для логарифмического тренда

print("${0}ln(x) + {1}$".format(\*set\_log\_by\_data))  # формула

# экспоненциальный

set\_exp\_by\_data = np.polyfit(numpy\_x, np.log(numpy\_y), 1) # работа с полиномом 1 степени + логарифмирование

exp\_trend = [np.exp(set\_exp\_by\_data[1]) \* np.exp(set\_exp\_by\_data[0] \* x) for x in numpy\_x] # создание одномерного массива для экспоненциального тренда

print("${1}e^{0}x$".format(\*set\_exp\_by\_data))  # формула

# Расчёт R^2

linear\_r2 = r2\_score(numpy\_y, linear\_trend(numpy\_x))

polinom\_r2 = r2\_score(numpy\_y, polinom\_trend(numpy\_x))

log\_r2 = r2\_score(numpy\_y, log\_trend)

exp\_r2 = r2\_score(numpy\_y, exp\_trend)

# Отображение графиков

plt.figure(figsize=(15, 15)) # размер графика

# 2 графика по горизонтали, 2 по вертикали

plt.subplot(2, 2, 1)

# !!! Текущая ячейка - 1 (левый верхний график)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

plt.plot(numpy\_x, linear\_trend(numpy\_x), linestyle='dashed', color="orange", label = 'linear trend') # линейный тренд

plt.grid(color="gainsboro") # Сетка

plt.legend(loc='upper right', fontsize=12)

plt.title("Линейный \n$R^2=$" + str(linear\_r2))

# !!! Текущая ячейка - 2

plt.subplot(2, 2, 2)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

x = np.linspace(numpy\_x.min(), numpy\_x.max()) # набор данных для x для большей гладкости графика (50 точек)

plt.plot(x, polinom\_trend(x), linestyle='dashed', color="orange", label = 'polinomial trend') # полиномиальный тренд

plt.grid(color="gainsboro") # Сетка

plt.legend(loc = 'center left', fontsize=12, bbox\_to\_anchor=(1, 0.5))

plt.title("Полиномиальный \n$R^2=$" + str(polinom\_r2) + polinom\_title)

# !!! Текущая ячейка - 3

plt.subplot(2, 2, 3)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

plt.plot(numpy\_x, log\_trend, linestyle='dashed', color="orange", label = 'log trend') # логарифмический тренд

plt.grid(color="gainsboro") # Сетка

plt.legend(loc = 'upper right', fontsize=12)

plt.title("Логарифмический \n$R^2=$" + str(log\_r2))

# !!! Текущая ячейка - 4

plt.subplot(2, 2, 4)

plt.scatter(numpy\_x, numpy\_y, label = 'data') # точечный график по x\_numpy, y\_numpy

plt.plot(numpy\_x, exp\_trend, linestyle='dashed', color="orange", label = 'exp trend')

plt.grid(color="gainsboro") # Сетка

plt.legend(loc = 'center left', fontsize=12, bbox\_to\_anchor=(1, 0.5))

plt.title("Экспоненциальный \n$R^2=$" + str(exp\_r2))

fig = plt.gcf() # Взять текущую фигуру

fig.set\_size\_inches(15, 15) # Задать размеры графика

plt.show()

print("x:", x)

print("y:", y)

Результат работы кода:

Изображение выглядит как текст, число, снимок экрана, диаграмма

Автоматически созданное описание

**Выводы**

**Сравнение Excel и Python**

Графики, созданные в Excel и Python, могут различаться по нескольким причинам:

 **Методы аппроксимации**: Excel применяет свои собственные алгоритмы для вычисления параметров тренда, которые могут отличаться от тех, что используются в Python.

 **Числовая стабильность**: В Python могут использоваться более точные алгоритмы для расчета коэффициентов уравнений и R², что может влиять на результаты.

 **Преобразование данных**: В Python для предотвращения ошибок при работе со степенной и логарифмической аппроксимацией к данным добавляется смещение (x+1), что может изменить параметры модели.

**Коэффициент детерминации (R²)**

Коэффициент R² показывает, насколько хорошо модель объясняет вариацию данных. Значения R² для различных моделей в Python и Excel могут быть близки, но различия в расчетах могут приводить к расхождениям.

**Вывод по моделям**

Согласно значению R², наиболее подходящей моделью для данных является полиномиальная модель шестой степени. Линейная модель в Excel имеет наименьшее значение R², что указывает на ее низкую точность в описании данных. В Python логарифмическая модель показала еще менее точный результат.

**Общий вывод по лабораторной работе**

В процессе выполнения лабораторной работы были решены две задачи с применением различных методов анализа и моделирования.

Первая задача касалась оптимизации производственных процессов на швейной фабрике. В результате были успешно разработаны и решены математические модели, направленные на максимизацию прибыли. Решение было выполнено двумя способами: с использованием MS Excel и библиотеки PuLP в Python. Результаты обоих подходов совпали, что подтверждает правильность математической модели и её реализации. Прибыль в расчете на единицу товарооборота j-й группы, ден. Ед. = 814285,7143

Вторая задача была сосредоточена на анализе котировок и построении аналитических моделей с использованием Excel и Python. Были построены линии тренда с применением различных типов аппроксимации (линейной, полиномиальной, степенной и логарифмической). В результате в Python и Excel были получены близкие, но не идентичные результаты, что объясняется различиями в методах расчёта и обработке данных. Полиномиальная модель шестой степени оказалась наиболее подходящей для описания данных по коэффициенту детерминации R².

Различия в точности между платформами подчеркивают важность выбора инструмента в зависимости от конкретной задачи.

**Итог**

Работа продемонстрировала, что использование математических моделей и современных инструментов анализа данных позволяет эффективно решать задачи оптимизации и анализа. Применение Python предоставляет более широкие возможности для автоматизации, обработки данных и построения сложных моделей.

**Ссылка на git: https://github.com/gr1fnn/lab3.2.1**