## **Постановка задачи**

Исследовать зависимость между стоимостью грузовой автомобильной перевозки Y тысяч рублей, массой груза X1 тонн и расстоянием X2 тысяч километров по 20 транспортным компаниям, используя данные наблюдений, приведенные в таблице 1.

Таблица 1. Исходные данные

|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Y** |
| 1,9 | 1,8 | 9 |
| 2 | 2,2 | 9,6 |
| 2,6 | 0,9 | 7,44 |
| 3 | 1,8 | 6,96 |
| 3,4 | 1 | 6,48 |
| 3,7 | 3 | 16,56 |
| 4,5 | 1,1 | 8,52 |
| 8 | 2,3 | 30,6 |
| 9 | 0,5 | 7,2 |
| 14 | 1,5 | 18,96 |
| 15 | 2,4 | 26,4 |
| 17 | 1,1 | 12,8 |
| 18 | 0,6 | 6,32 |
| 19 | 2,6 | 59,2 |
| 21 | 0,6 | 9,2 |
| 22 | 1,3 | 20,8 |
| 25 | 2,5 | 44,8 |
| 27 | 2,3 | 41,6 |
| 34 | 1,6 | 20,8 |
| 36 | 2 | 40,8 |

## **Алгоритм построения модели**

1. Специфицировать модель, как линейную зависимость объясняемых переменных Y от объясняющих переменных X.
2. По результатам наблюдений сформировать матрицы X= (n, k-1), F=(n,k), Y= (n,1).
3. Решить систему нормальных уравнений (FтF)\*A=FтY для нахождения вектора оценок неизвестных коэффициентов A=(k,1). Решением этой системы будет A = (FТ\*F)-1 \* FТY.

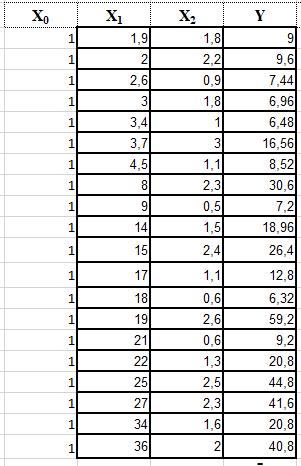
## **Спецификация модели**

Располагаем пространственной выборкой объема n=20 c числом объясняющих переменных k=2. Модель в виде линейной зависимости:

 (1)

По результатам наблюдений формируем матрицы F=[20x3] и Y=[20x1]. Фиктивная переменная X0 равна единице для всех наблюдений i.

Таблица 2. Матрицы F и Y



## **Решение системы нормальных уравнений**

Одним из методов решения таких задач является метод наименьших квадратов. Анализ постановки задачи приводит к системе нормальных уравнений:

*FTFA=FTY*

Находим коэффициенты модели A=(FTF)-1(FTY) или решение системы нормальных уравнений в векторной форме, где F – матрица значений объясняющих переменных, а Y – вектор значений объясняемой переменной.

Решим задачу с помощью матричных операций, основываясь на матричном уравнении A=(FTF)-1(FTY) используя язык программирования python и библиотеки numpy, pandas.

Импортируем необходимые библиотеки

import numpy as np

import pandas as pd

import random

from google.colab import files

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

uploaded = files.upload()

import io

Загрузим наши данные в формате csv, затем используя библиотеку pandas создадим новый DataFrame и инициализируем данные.

Добавим в матрицу X столбец единиц

#Инициализация данных

X = data.iloc[:, 0:2]

w = np.ones(20)

X = pd.DataFrame(X)

W = pd.DataFrame(w)

X['Ones'] = W

При помощи библиотеки PyPlot визуализируем данные.

fig = plt.figure(1)

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(X.iloc[:, 0], X.iloc[:, 1], Y)

ax.set\_xlabel('X1')

ax.set\_ylabel('X2')

ax.set\_zlabel('Y')

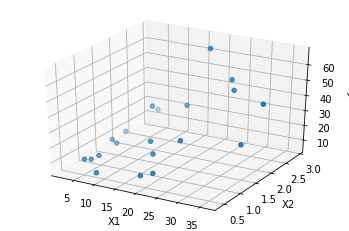


Рисунок 1. Визуализация данных.

При помощи numpy находим решение системы линейных уравнений, используя функцию linalg.solve.

X = np.array(X)

Y = np.array(Y)

a = np.linalg.solve(np.dot(X.T,X ), np.dot(X.T, Y))

На выходе получаем массив содержащий три решения, которые и будут являться искомыми коэффициентами.

[ 0.91375319 13.98004571 -14.04571276]

Для нахождения невязки, используем формулу средний квадратичной ошибки , находим невязку,

predictedY = np.dot(X, a)

ErrorSq = Y - predictedY

ErrorSqMean = Y - Y.mean()

В результате проделанных вычислений мы получили следующие результаты:

Невязка: 0.742

Коэффициенты: 0.913 , 13.98 , 14.04

Визуализируем полученные результаты, отобразив невязку между результатами ожидаемыми данными, используя библиотеку matplotlib.pyplot.

#  Подготовка данных для визуализации

xx, yy, zz = np.meshgrid(X[:, 0], X[:, 1], X[:, 2])

combinedArrays = np.vstack((xx.flatten(), yy.flatten(), zz.flatten())).T

Z = combinedArrays.dot(a)

# Визуализация предсказанных данных и реальных.

fig = plt.figure(2)

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(X[:, 0], X[:, 1], Y, color='r', label='Actual')

ax.scatter(X[:, 0], X[:, 1], predictedY, color='g', label='Predicted')

ax.plot\_trisurf(combinedArrays[:, 0], combinedArrays[:, 1], Z, alpha=0.5)

ax.set\_xlabel('X1')

ax.set\_ylabel('X2')

ax.set\_zlabel('Y')

ax.legend()

plt.show()

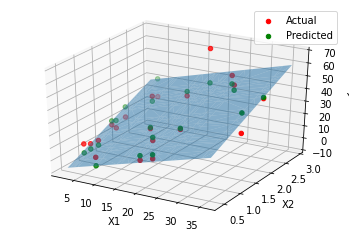


Рисунок 3 после нахождения коэфицентов

Теперь нам необходимо вычислить выборочный коэффициент корреляции (детерминации) R2. Для того чтобы найти коэффициент детерминации, используем метод np.corrcoef. Если 0,6 < R2 < 0.9, то можно предположить, что модель адекватно описывает моделируемый объект, иначе следует перейти к началу расчёта и провести дополнительную спецификацию модели.

correlation\_matrix = np.corrcoef(df)

correlation\_xy = correlation\_matrix[0,1]

r\_squared = correlation\_xy\*\*2

print(r\_squared)

В нашем случае R2 = 0,732, т.е. в 73,2% случаев изменения X приводят к изменению Y. Полученное значение R2 удовлетворяет условиям и, следовательно, модель удовлетворительно описывает экспериментальные данные.

При помощи scipy.stats.shapiro Проверим гепотизу H0 на нормальное распределение при помощм критерия Шапиро-Уилка:

stat, p = scipy.stats.shapiro(df['Y']) # тест Шапиро-Уилка

print('Statistics=%.3f, p-value=%.3f' % (stat, p))

В результате получаем:

Statistics=0.820, p-value=0.002

Значение p меньше чем 0,05 из этого следует, что нулевая гипотеза отвергается и коэффициенты считаются значимо отличающимся от нуля.

Проверим значимость коэф. А: H0 гипотеза при помощи метода распределения Фишера. Для этого используем формулу, изображенную на рисунке 3.

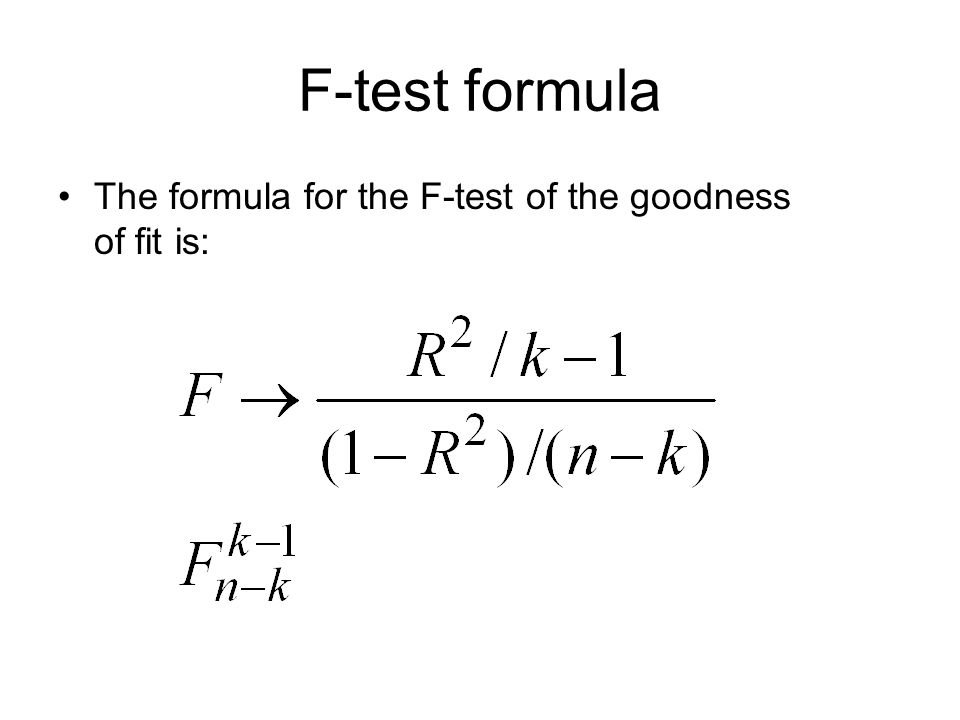


Рисунок 3. F-test

Посчитаем F-Test:

fstat = (r\_squared/(1-r\_squared))\*((19-3-1)/3)

При помощи помощи библиотеки symbulate найдём табличное значение p, где dfN и dfD степени свободы в числителе и знаменателе:

import symbulate as sm

dfN = 15

dfD = 3

pVal = 1-sm.F(dfN,dfD).cdf(fstat)

На выходе мы получим значение 0.0264 равный 2%, что означает, что нулевая гипотеза отвергается и коэффициенты считаются значимо отличающимся от нуля.