```python
import numpy as np
import matplotlib.pyplot as plt

# Данные для полинома первой степени
x_1 = np.array([0, 1, 2, 3])
y_1 = np.array([1.6, 2.4, 3.1, 4.9])

# Данные для полинома второй степени
delt = 1.0
x_2 = np.linspace(-10, 10, 20)
y_2 = x_2 ** 2 + delt * (np.random.rand(20) - 0.5)
x_2 += delt * (np.random.rand(20) - 0.5)


# Данные для полинома третьей степени

delt = 2.0
x_3 = np.linspace(-10, 10, 101)
y_3 = x_3 ** 3 + delt * x_3 ** 2 + delt * (np.random.rand(101) - 0.5)
x_3 += delt * (np.random.rand(101) - 0.5)


# Запись в файл
x_1.tofile("x1_data.txt", "\n")
y_1.tofile("y1_data.txt", "\n")

x_2.tofile("x2_data.txt", "\n")
y_2.tofile("y2_data.txt", "\n")

x_3.tofile("x3_data.txt", "\n")
y_3.tofile("y3_data.txt", "\n")


x_1 = np.fromfile("x1_data.txt", float, sep="\n")
y_1 = np.fromfile("y1_data.txt", float, sep="\n")

x_2 = np.fromfile("x2_data.txt", float, sep="\n")
y_2 = np.fromfile("y2_data.txt", float, sep="\n")

x_3 = np.fromfile("x3_data.txt", float, sep="\n")
y_3 = np.fromfile("y3_data.txt", float, sep="\n")

print("X_1: ", x_1, sep="\n")
print("Y_1: ", y_1, sep="\n")

print("X_2: ", x_2, sep="\n")
print("Y_2: ", y_2, sep="\n")

print("X_3: ", x_3, sep="\n")
print("Y_3: ", y_3, sep="\n")
```

```
print(np.ones(len(x_1)))

# Присоединяем данные попарно по вертикальной оси
a_1 = np.vstack([x_1, np.ones(len(x_1))]).T

a_2 = np.vstack([x_2 ** 2, x_2, np.ones(len(x_2))]).T

a_3 = np.vstack([x_3 ** 3, x_3 ** 2, x_3, np.ones(len(x_3))]).T

print("A_1 по x_1: ", a_1, sep="\n")

print("A_1 по x_2: ", a_2, sep="\n")

print("A_1 по x_3: ", a_3, sep="\n")

# Используем метод lstsq (метод наименьших квадратов с мин.
погрешностью) для решения его относительно вектора p

# m - наклон прямой
# c - сдвиг по оси y

m, c = np.linalg.lstsq(a_1, y_1, rcond=-1)[0]

print("M:", m, "c:", c, sep="\n")

plt.figure(figsize=(40, 16))

plt.subplot(2, 2, 1)
plt.plot(x_1, y_1, 'o', label="Исходные данные")
plt.plot(x_1, m*x_1 + c, "r", label="Линейная экстраполяция")
plt.legend()
plt.grid()



s = np.linalg.lstsq(a_2, y_2, rcond=-1)[0]

print("Coefficients for polinom^2: ", s)

x_prec = np.linspace(-10, 10, 101)

plt.subplot(2, 2, 2)

plt.plot(x_2, y_2, 'D')
plt.plot(x_prec, s[0] * x_prec ** 2 + s[1] * x_prec + s[2], '-', lw=2)
plt.grid()


s = np.linalg.lstsq(a_3, y_3, rcond=-1)[0]
```

```
print("Coefficients for polinom^3: ", s)

x_prec = np.linspace(-10,10, 101)

plt.subplot(2, 2, 3)

plt.plot(x_3, y_3, "D")

plt.plot(x_prec, s[0] * x_prec ** 3 + s[1] * x_prec ** 2 + s[2] *
x_prec + s[3], "--", lw=3)
plt.grid()
```

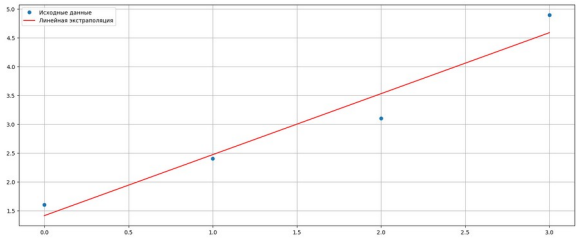
```
X_1:
[0. 1. 2. 3.]
Y_1:
[1.6 2.4 3.1 4.9]
X_2:
[-10.08487683  -8.89809932  -7.44118981  -7.08575298  -5.62773799
  -5.03925112  -3.25905208  -2.74388886  -1.57182058  -0.94235968
   0.36182092   1.50327188   2.80324011   3.78924418   4.92818872
   5.68932647   6.58749123   8.16633255   8.71264318  10.36867896]
Y_2:
[100.38680786  80.42100575  62.80752886  47.30723555  33.32924524
  22.7841825   13.13678464   6.82877259   2.2117469    0.60697104
   0.23234697   2.07038454   6.89657645  13.37327922  22.73169968
  33.08311977  46.83591542  62.82156694  79.58888009  99.77856338]
X_3:
[-1.09954517e+01 -9.59172456e+00 -1.02857365e+01 -9.78384458e+00
 -9.07910373e+00 -8.70812034e+00 -9.36294647e+00 -9.13341123e+00
 -8.46114976e+00 -7.76743913e+00 -7.85645435e+00 -7.85976546e+00
 -7.95306677e+00 -6.83884853e+00 -7.48835208e+00 -6.63496976e+00
 -7.46834482e+00 -6.84053993e+00 -5.43958871e+00 -5.91702329e+00
 -6.78360775e+00 -6.01625631e+00 -4.77766197e+00 -5.65685617e+00
 -5.34321921e+00 -4.54520684e+00 -5.23409768e+00 -3.73265126e+00
 -4.54398281e+00 -3.32126787e+00 -3.76177022e+00 -4.70601900e+00
 -2.78782074e+00 -2.99538665e+00 -2.57071074e+00 -2.62136755e+00
 -2.81064995e+00 -1.69056437e+00 -1.43244065e+00 -2.60551505e+00
 -1.00709747e+00 -2.74933681e+00 -1.66141651e+00 -1.62651337e+00
 -4.56888047e-01 -4.56668056e-03 -1.51269979e+00 -1.26844753e+00
 -1.37124597e+00  3.37399125e-01  4.48982019e-01  6.02714591e-01
 -1.43911986e-01  3.38708847e-01  2.54226747e-01  1.18955450e+00
  1.55745847e+00  8.99981098e-01  1.56547703e+00  1.26593843e+00
  2.25122860e+00  2.00506689e+00  2.29343339e+00  3.45501522e+00
  1.87915280e+00  3.20144524e+00  3.14227605e+00  4.13784575e+00
  3.37682359e+00  3.25761364e+00  4.52834963e+00  4.72832782e+00
  4.06398459e+00  5.05758550e+00  3.87970086e+00  4.12227753e+00
```

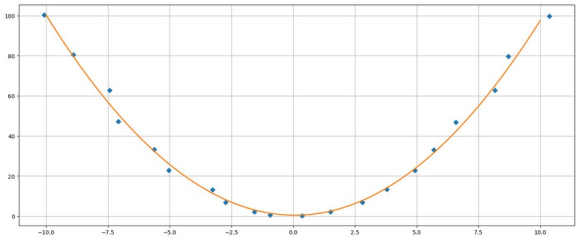
```
  4.62557139e+00  6.24973674e+00  5.97273160e+00  5.52578760e+00
  6.05491814e+00  6.00620114e+00  5.47331801e+00  6.36090860e+00
  6.80227815e+00  6.21607109e+00  6.95901218e+00  7.54595575e+00
  8.29182908e+00  8.16701166e+00  7.89391901e+00  8.87078311e+00
  9.11292246e+00  7.99093705e+00  8.21915335e+00  9.34798620e+00
  9.01617599e+00  9.25619833e+00  9.95112050e+00  8.94626004e+00
  9.34352986e+00]
Y_3:
[-8.00201802e+02 -7.50065884e+02 -7.00777744e+02 -6.54088351e+02
 -6.09889462e+02 -5.66843275e+02 -5.27137509e+02 -4.89079354e+02
 -4.51336754e+02 -4.15945493e+02 -3.83940319e+02 -3.53246201e+02
 -3.23336171e+02 -2.95793712e+02 -2.69921136e+02 -2.44957987e+02
 -2.22245075e+02 -2.00709547e+02 -1.79604211e+02 -1.60944699e+02
 -1.43633336e+02 -1.27374081e+02 -1.12648742e+02 -9.97604622e+01
 -8.63981279e+01 -7.41457651e+01 -6.53114981e+01 -5.57426714e+01
 -4.65265781e+01 -3.85821093e+01 -3.10610107e+01 -2.68829571e+01
 -1.98870609e+01 -1.63617714e+01 -1.16572353e+01 -9.47052503e+00
 -6.06894530e+00 -4.83971208e+00 -1.97271263e+00 -1.90447444e+00
  1.07487372e-01  1.62466946e+00  1.56455134e+00  4.88370585e-01
  1.74428354e+00  1.51527570e+00  1.33608700e+00  1.37107468e+00
  7.86973400e-01  5.31276607e-01 -5.17167332e-01  4.42260386e-02
  1.32143018e+00  6.14820892e-01  2.75512035e+00  2.85658841e+00
  5.41630726e+00  6.86366560e+00  9.53951489e+00  1.13548400e+01
  1.52946750e+01  2.00294645e+01  2.48106825e+01  3.09320570e+01
  3.66586756e+01  4.58932362e+01  5.24154215e+01  6.19572676e+01
  7.16406501e+01  8.39645824e+01  9.66343105e+01  1.09137783e+02
  1.24588983e+02  1.39221831e+02  1.57467902e+02  1.74401182e+02
  1.95014270e+02  2.15741486e+02  2.39275096e+02  2.61873406e+02
  2.88871925e+02  3.15450691e+02  3.43461532e+02  3.74403282e+02
  4.06158264e+02  4.41696805e+02  4.76595638e+02  5.15324619e+02
  5.54134583e+02  5.96487122e+02  6.39115463e+02  6.85209409e+02
  7.33210176e+02  7.83160752e+02  8.35484025e+02  8.91052732e+02
  9.48531752e+02  1.00663489e+03  1.06835845e+03  1.13234732e+03
  1.19934825e+03]
[1. 1. 1. 1.]
A_1 по x_1:
[[0. 1.]
 [1. 1.]
 [2. 1.]
 [3. 1.]]
A_1 по x_2:
[[101.70474076 -10.08487683   1.        ]
 [ 79.17617158  -8.89809932   1.        ]
 [ 55.37130582  -7.44118981   1.        ]
 [ 50.20789533  -7.08575298   1.        ]
 [ 31.67143489  -5.62773799   1.        ]
 [ 25.39405186  -5.03925112   1.        ]
 [ 10.62142043  -3.25905208   1.        ]
 [  7.52892605  -2.74388886   1.        ]
```

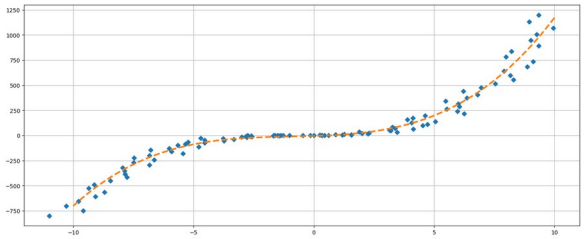
```
 [  2.47061993  -1.57182058   1.        ]
 [  0.88804177  -0.94235968   1.        ]
 [  0.13091438   0.36182092   1.        ]
 [  2.25982636   1.50327188   1.        ]
 [  7.85815512   2.80324011   1.        ]
 [ 14.35837147   3.78924418   1.        ]
 [ 24.28704404   4.92818872   1.        ]
 [ 32.3684357    5.68932647   1.        ]
 [ 43.39504073   6.58749123   1.        ]
 [ 66.68898739   8.16633255   1.        ]
 [ 75.91015118   8.71264318   1.        ]
 [107.50950335  10.36867896   1.        ]]
A_1 по x_3:
[[-1.32934966e+03  1.20899959e+02 -1.09954517e+01  1.00000000e+00]
 [-8.82449978e+02  9.20011800e+01 -9.59172456e+00  1.00000000e+00]
 [-1.08819364e+03  1.05796375e+02 -1.02857365e+01  1.00000000e+00]
 [-9.36544969e+02  9.57236147e+01 -9.78384458e+00  1.00000000e+00]
 [-7.48391651e+02  8.24301245e+01 -9.07910373e+00  1.00000000e+00]
 [-6.60348607e+02  7.58313598e+01 -8.70812034e+00  1.00000000e+00]
 [-8.20800518e+02  8.76647667e+01 -9.36294647e+00  1.00000000e+00]
 [-7.61901864e+02  8.34192006e+01 -9.13341123e+00  1.00000000e+00]
 [-6.05742640e+02  7.15910552e+01 -8.46114976e+00  1.00000000e+00]
 [-4.68633764e+02  6.03331106e+01 -7.76743913e+00  1.00000000e+00]
 [-4.84930807e+02  6.17238750e+01 -7.85645435e+00  1.00000000e+00]
 [-4.85544187e+02  6.17759130e+01 -7.85976546e+00  1.00000000e+00]
 [-5.03041582e+02  6.32512710e+01 -7.95306677e+00  1.00000000e+00]
 [-3.19851914e+02  4.67698492e+01 -6.83884853e+00  1.00000000e+00]
 [-4.19912465e+02  5.60754169e+01 -7.48835208e+00  1.00000000e+00]
 [-2.92090104e+02  4.40228237e+01 -6.63496976e+00  1.00000000e+00]
 [-4.16555703e+02  5.57761743e+01 -7.46834482e+00  1.00000000e+00]
 [-3.20089293e+02  4.67929865e+01 -6.84053993e+00  1.00000000e+00]
 [-1.60952672e+02  2.95891253e+01 -5.43958871e+00  1.00000000e+00]
 [-2.07161877e+02  3.50111646e+01 -5.91702329e+00  1.00000000e+00]
 [-3.12163544e+02  4.60173340e+01 -6.78360775e+00  1.00000000e+00]
 [-2.17760442e+02  3.61953400e+01 -6.01625631e+00  1.00000000e+00]
 [-1.09055170e+02  2.28260539e+01 -4.77766197e+00  1.00000000e+00]
 [-1.81019521e+02  3.20000218e+01 -5.65685617e+00  1.00000000e+00]
 [-1.52548863e+02  2.85499915e+01 -5.34321921e+00  1.00000000e+00]
 [-9.38989976e+01  2.06589053e+01 -4.54520684e+00  1.00000000e+00]
 [-1.43392181e+02  2.73957785e+01 -5.23409768e+00  1.00000000e+00]
 [-5.20058560e+01  1.39326855e+01 -3.73265126e+00  1.00000000e+00]
 [-9.38231563e+01  2.06477798e+01 -4.54398281e+00  1.00000000e+00]
 [-3.66363089e+01  1.10308203e+01 -3.32126787e+00  1.00000000e+00]
 [-5.32324914e+01  1.41509152e+01 -3.76177022e+00  1.00000000e+00]
 [-1.04222390e+02  2.21466149e+01 -4.70601900e+00  1.00000000e+00]
 [-2.16667879e+01  7.77194446e+00 -2.78782074e+00  1.00000000e+00]
 [-2.68756311e+01  8.97234120e+00 -2.99538665e+00  1.00000000e+00]
 [-1.69886800e+01  6.60855372e+00 -2.57071074e+00  1.00000000e+00]
 [-1.80129050e+01  6.87156785e+00 -2.62136755e+00  1.00000000e+00]
```

```
[-2.22034408e+01  7.89975314e+00 -2.81064995e+00  1.00000000e+00]
[-4.83164633e+00  2.85800790e+00 -1.69056437e+00  1.00000000e+00]
[-2.93920524e+00  2.05188622e+00 -1.43244065e+00  1.00000000e+00]
[-1.76880826e+01  6.78870866e+00 -2.60551505e+00  1.00000000e+00]
[-1.02144389e+00  1.01424531e+00 -1.00709747e+00  1.00000000e+00]
[-2.07818325e+01  7.55885290e+00 -2.74933681e+00  1.00000000e+00]
[-4.58601601e+00  2.76030482e+00 -1.66141651e+00  1.00000000e+00]
[-4.30301554e+00  2.64554575e+00 -1.62651337e+00  1.00000000e+00]
[-9.53738662e-02  2.08746687e-01 -4.56888047e-01  1.00000000e+00]
[-9.52361655e-08  2.08545713e-05 -4.56668056e-03  1.00000000e+00]
[-3.46145144e+00  2.28826067e+00 -1.51269979e+00  1.00000000e+00]
[-2.04088023e+00  1.60895913e+00 -1.26844753e+00  1.00000000e+00]
[-2.57837506e+00  1.88031551e+00 -1.37124597e+00  1.00000000e+00]
[ 3.84088987e-02  1.13838169e-01  3.37399125e-01  1.00000000e+00]
[ 9.05079745e-02  2.01584853e-01  4.48982019e-01  1.00000000e+00]
[ 2.18945043e-01  3.63264878e-01  6.02714591e-01  1.00000000e+00]
[-2.98051217e-03  2.07106597e-02 -1.43911986e-01  1.00000000e+00]
[ 3.88579265e-02  1.14723683e-01  3.38708847e-01  1.00000000e+00]
[ 1.64309896e-02  6.46312389e-02  2.54226747e-01  1.00000000e+00]
[ 1.68326709e+00  1.41503991e+00  1.18955450e+00  1.00000000e+00]
[ 3.77789099e+00  2.42567688e+00  1.55745847e+00  1.00000000e+00]
[ 7.28954068e-01  8.09965976e-01  8.99981098e-01  1.00000000e+00]
[ 3.83654327e+00  2.45071834e+00  1.56547703e+00  1.00000000e+00]
[ 2.02879308e+00  1.60260012e+00  1.26593843e+00  1.00000000e+00]
[ 1.14092945e+01  5.06803020e+00  2.25122860e+00  1.00000000e+00]
[ 8.06095687e+00  4.02029324e+00  2.00506689e+00  1.00000000e+00]
[ 1.20630851e+01  5.25983670e+00  2.29343339e+00  1.00000000e+00]
[ 4.12429665e+01  1.19371302e+01  3.45501522e+00  1.00000000e+00]
[ 6.63569307e+00  3.53121526e+00  1.87915280e+00  1.00000000e+00]
[ 3.28124177e+01  1.02492516e+01  3.20144524e+00  1.00000000e+00]
[ 3.10265156e+01  9.87389877e+00  3.14227605e+00  1.00000000e+00]
[ 7.08472325e+01  1.71217674e+01  4.13784575e+00  1.00000000e+00]
[ 3.85057087e+01  1.14029376e+01  3.37682359e+00  1.00000000e+00]
[ 3.45699480e+01  1.06120467e+01  3.25761364e+00  1.00000000e+00]
[ 9.28581129e+01  2.05059504e+01  4.52834963e+00  1.00000000e+00]
[ 1.05711622e+02  2.23570840e+01  4.72832782e+00  1.00000000e+00]
[ 6.71206504e+01  1.65159707e+01  4.06398459e+00  1.00000000e+00]
[ 1.29368845e+02  2.55791711e+01  5.05758550e+00  1.00000000e+00]
[ 5.83975631e+01  1.50520788e+01  3.87970086e+00  1.00000000e+00]
[ 7.00505712e+01  1.69931720e+01  4.12227753e+00  1.00000000e+00]
[ 9.89683122e+01  2.13959107e+01  4.62557139e+00  1.00000000e+00]
[ 2.44109776e+02  3.90592094e+01  6.24973674e+00  1.00000000e+00]
[ 2.13068377e+02  3.56735228e+01  5.97273160e+00  1.00000000e+00]
[ 1.68726214e+02  3.05343286e+01  5.52578760e+00  1.00000000e+00]
[ 2.21985613e+02  3.66620337e+01  6.05491814e+00  1.00000000e+00]
[ 2.16670415e+02  3.60744521e+01  6.00620114e+00  1.00000000e+00]
[ 1.63965337e+02  2.99572101e+01  5.47331801e+00  1.00000000e+00]
[ 2.57369730e+02  4.04611583e+01  6.36090860e+00  1.00000000e+00]
[ 3.14748130e+02  4.62709880e+01  6.80227815e+00  1.00000000e+00]
```

```
 [ 2.40186126e+02  3.86395398e+01  6.21607109e+00  1.00000000e+00]
 [ 3.37010002e+02  4.84278505e+01  6.95901218e+00  1.00000000e+00]
 [ 4.29677649e+02  5.69414482e+01  7.54595575e+00  1.00000000e+00]
 [ 5.70099978e+02  6.87544295e+01  8.29182908e+00  1.00000000e+00]
 [ 5.44740327e+02  6.67000795e+01  8.16701166e+00  1.00000000e+00]
 [ 4.91901333e+02  6.23139574e+01  7.89391901e+00  1.00000000e+00]
 [ 6.98048957e+02  7.86907929e+01  8.87078311e+00  1.00000000e+00]
 [ 7.56785888e+02  8.30453558e+01  9.11292246e+00  1.00000000e+00]
 [ 5.10261884e+02  6.38550749e+01  7.99093705e+00  1.00000000e+00]
 [ 5.55240644e+02  6.75544817e+01  8.21915335e+00  1.00000000e+00]
 [ 8.16872335e+02  8.73848461e+01  9.34798620e+00  1.00000000e+00]
 [ 7.32937834e+02  8.12914294e+01  9.01617599e+00  1.00000000e+00]
 [ 7.93045225e+02  8.56772075e+01  9.25619833e+00  1.00000000e+00]
 [ 9.85407709e+02  9.90247992e+01  9.95112050e+00  1.00000000e+00]
 [ 7.16019011e+02  8.00355688e+01  8.94626004e+00  1.00000000e+00]
 [ 8.15704641e+02  8.73015502e+01  9.34352986e+00  1.00000000e+00]]
M:
1.0600000000000003
c:
1.4099999999999997
Coefficients for polinom^2:  [ 0.98429862 -0.13485585  0.45383146]
Coefficients for polinom^3:  [ 0.86687319  2.39692036  7.02336226 -
5.49120704]
```

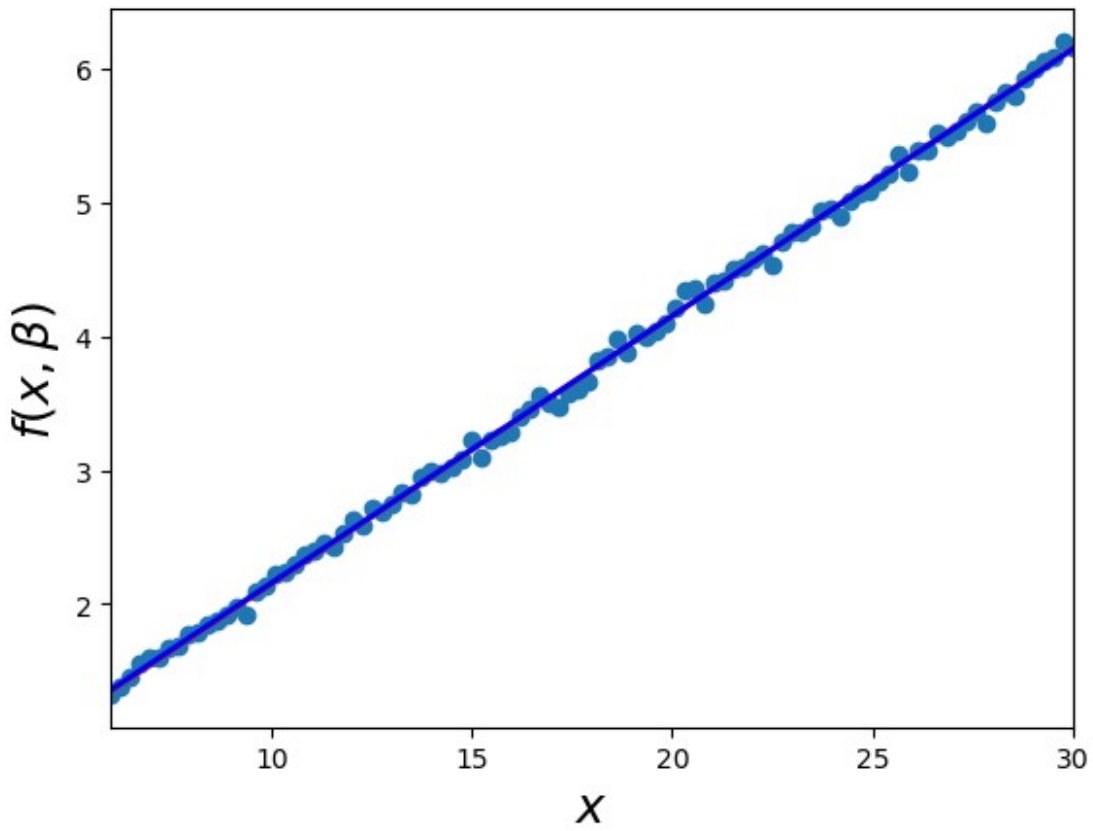






```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


# Задание два
# Необходимо проверить гипотезу, что наши точечно заданная функция
ложится на кривые вида:
# 1) f(x,b) = b_0  + b_1 x
```

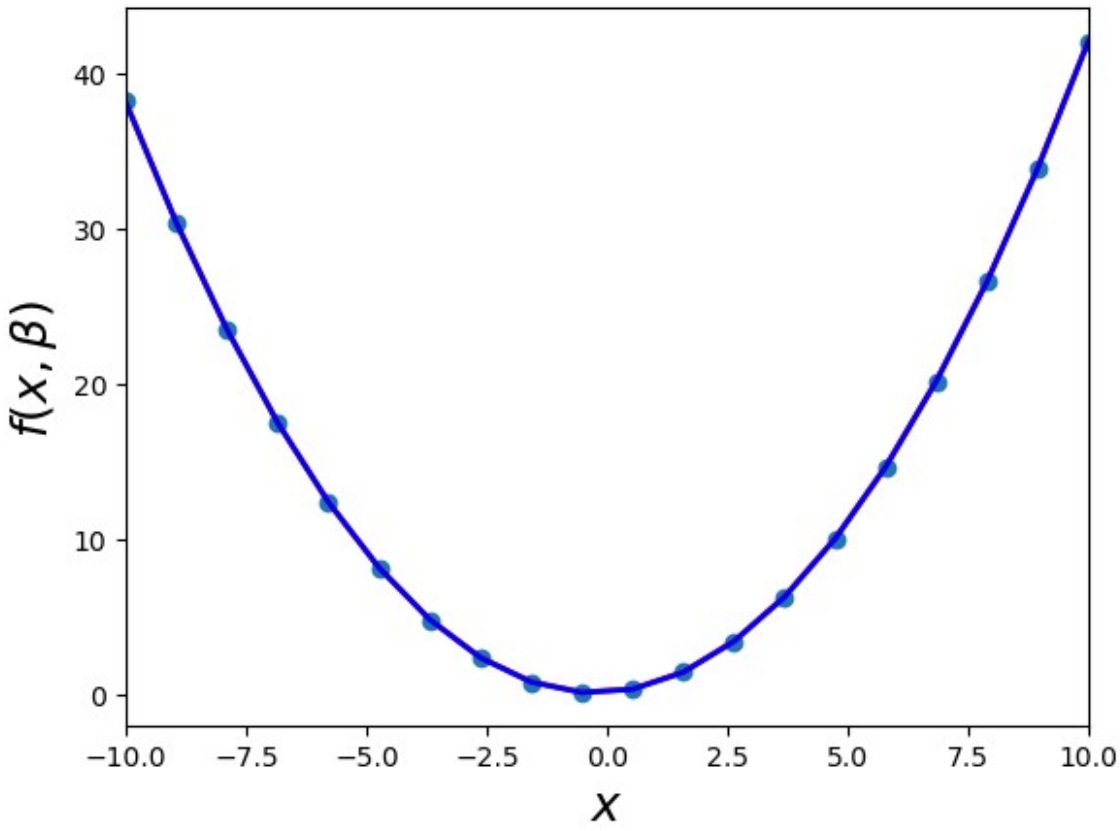
```
# 2) f(x,b) = b_0  + b_1 x + b_2 x^2
# 3) f(x,b) = b_0  + b_1 ln(x)
# 4) f(x,b) = b_0   x^(b_1 )

# Первая функция f(x,b) = b_0  + b_1 x

# Добавим шума в данные, сделанные по функции f(x, b) с коэффициентами
b = (0.15, 0.2)

beta = (0.15, 0.2)

def f(x, b0, b1):
    return b0 + b1 * x

# Зададим массив точек xi
xdata = np.linspace(6, 30, 100)

# создаем тоеритически правильные значения точек y1 (без шума)

y = f(xdata, *beta)

# зашумляем эти данные

ydata = y + 0.05 * np.random.randn(len(xdata))
beta_opt, beta_cov = curve_fit(f, xdata, ydata)
print(beta_opt)

# Вычислим линейное отклонение
lin_dev = sum(beta_cov[0])
print(lin_dev)

# Вычислим квадратичное отклонение
residuals = ydata - f(xdata, *beta_opt)
fres = sum(residuals ** 2)
print(fres)

fig, ax = plt.subplots()
ax.scatter(xdata, ydata)
ax.plot(xdata, y, "r", lw=2)
ax.plot(xdata, f(xdata, *beta_opt), "b", lw=2)

ax.set_xlim(6, 30)
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$f(x, \beta)$", fontsize=18)
plt.show()


[0.16066429 0.19952826]
0.00018347950857342657
0.24805600445858653
```




```
# Вторая функция b_0  + b_1 ln(x)

# Добавим шума в данные, сделанные по функции f(x, b) с коэффициентами
b = (0.15, 0.2, 0.4)

beta = (0.15, 0.2, 0.4)

def f(x, b0, b1, b2):
    return b0 + b1 * x + b2 * x ** 2

# Зададим массив точек xi
xdata = np.linspace(-10,10, 20)

# создаем тоеритически правильные значения точек y1 (без шума)

y = f(xdata, *beta)

# зашумляем эти данные

ydata = y + 0.05 * np.random.randn(len(xdata))
beta_opt, beta_cov = curve_fit(f, xdata, ydata)
print(beta_opt)

# Вычислим линейное отклонение
```

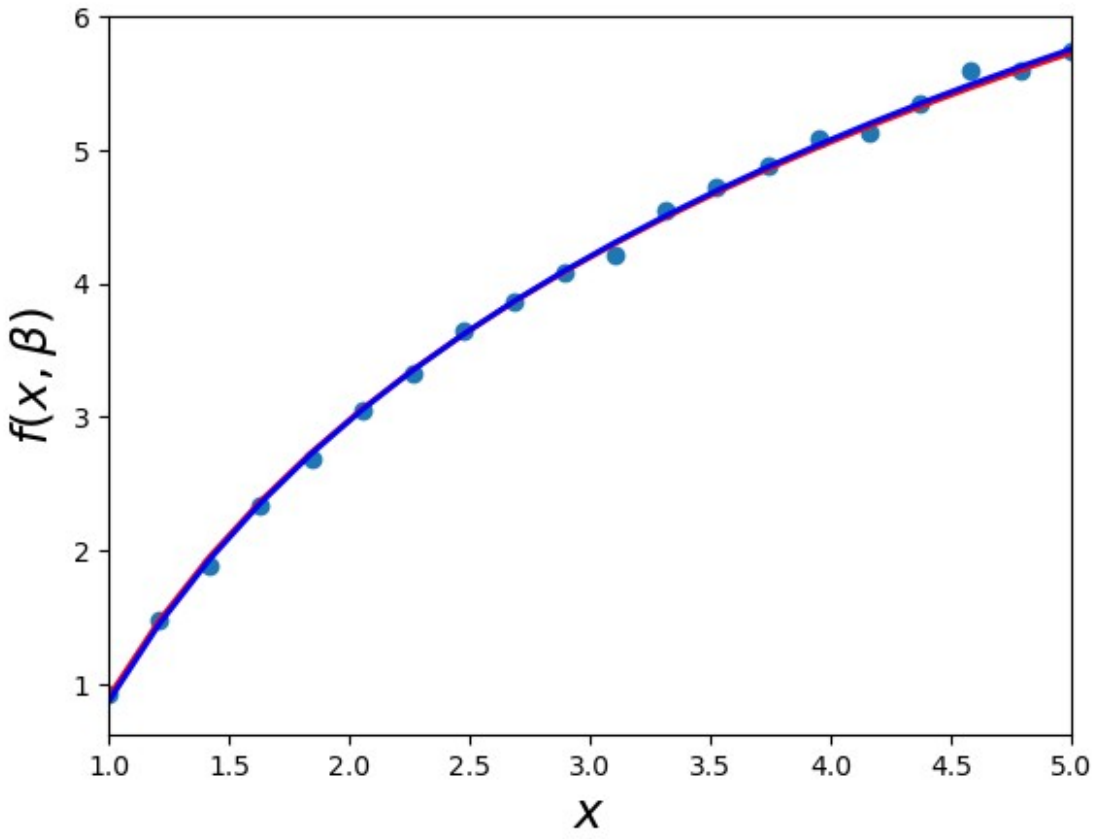
```python
lin_dev = sum(beta_cov[0])
print(lin_dev)

# Вычислим квадратичное отклонение
residuals = ydata - f(xdata, *beta_opt)
fres = sum(residuals ** 2)
print(fres)

fig, ax = plt.subplots()
ax.scatter(xdata, ydata)
ax.plot(xdata, y, "r", lw=2)
ax.plot(xdata, f(xdata, *beta_opt), "b", lw=2)

ax.set_xlim(-10, 10)
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$f(x, \beta)$", fontsize=18)
plt.show()
```

```
[0.13302034 0.19753196 0.39992447]
0.00024434136654730095
0.03733278088950316
```



```
# Третья функция f(x,b) = b_0  + b_1 ln(x)

# Добавим шума в данные, сделанные по функции f(x, b) с коэффициентами
b = (0.9, 3)

beta = (0.9, 3)

def f(x, b0, b1):
    return b0 + b1 * np.log(x)

# Зададим массив точек xi
xdata = np.linspace(1, 5, 20)

# создаем тоеритически правильные значения точек y1 (без шума)

y = f(xdata, *beta)

# зашумляем эти данные
ydata = y + 0.05 * np.random.randn(len(xdata))
beta_opt, beta_cov = curve_fit(f, xdata, ydata)
print(beta_opt)

# Вычислим линейное отклонение
lin_dev = sum(beta_cov[0])
print(lin_dev)

# Вычислим квадратичное отклонение
residuals = ydata - f(xdata, *beta_opt)
fres = sum(residuals ** 2)
print(fres)

fig, ax = plt.subplots()
ax.scatter(xdata, ydata)
ax.plot(xdata, y, "r", lw=2)
ax.plot(xdata, f(xdata, *beta_opt), "b", lw=2)

ax.set_xlim(1, 5)
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$f(x, \beta)$", fontsize=18)
plt.show()
```

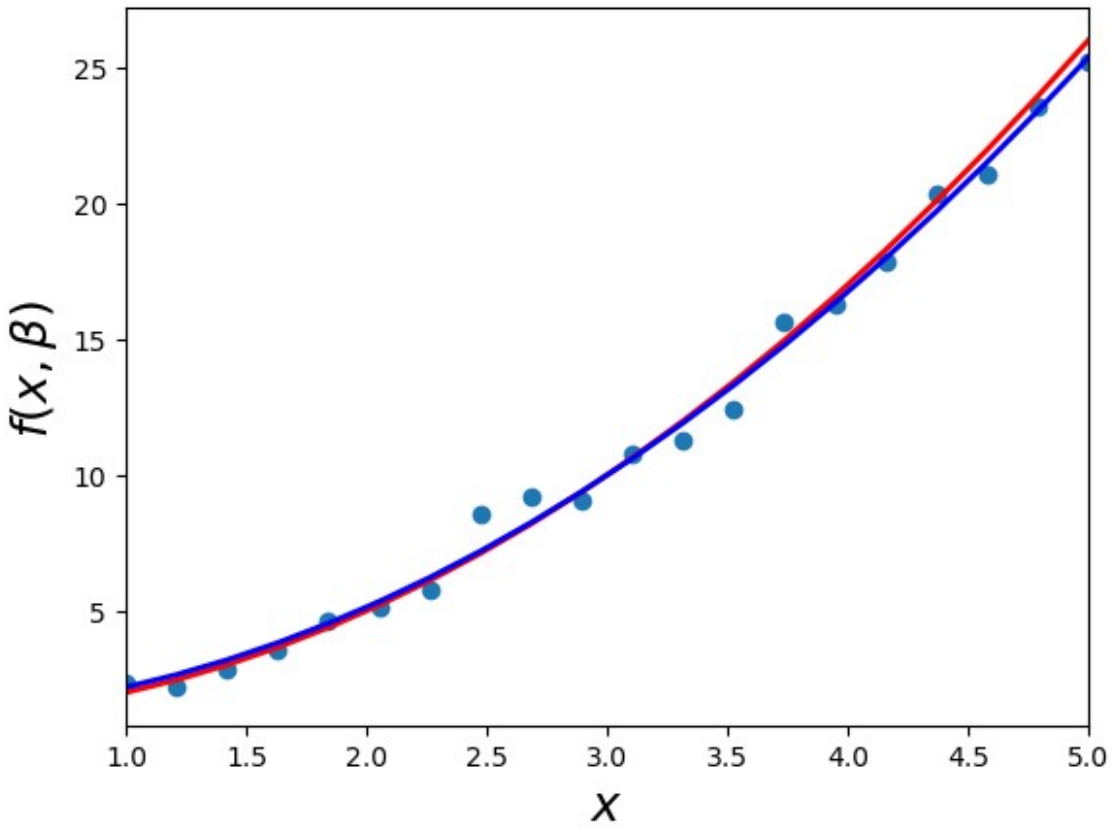
```
[0.86564392 3.03845308]
0.00010644503176363397
0.0381895899781241
```




```
# Четвертая функция f(x,b) = b_0   x^(b_1 )

# Добавим шума в данные, сделанные по функции f(x, b) с коэффициентами
b = (0.15, 0.2)

beta = (1, 2)

def f(x, b0, b1):
    return b0 + x ** b1

# Зададим массив точек xi
xdata = np.linspace(1, 5, 20)

# создаем тоеритически правильные значения точек y1 (без шума)

y = f(xdata, *beta)

# зашумляем эти данные
ydata = y + 0.7 * np.random.randn(len(xdata))
beta_opt, beta_cov = curve_fit(f, xdata, ydata)
print(beta_opt)

# Вычислим линейное отклонение
lin_dev = sum(beta_cov[0])
```

```
print(lin_dev)

# Вычислим квадратичное отклонение
residuals = ydata - f(xdata, *beta_opt)
fres = sum(residuals ** 2)
print(fres)

fig, ax = plt.subplots()
ax.scatter(xdata, ydata)
ax.plot(xdata, y, "r", lw=2)
ax.plot(xdata, f(xdata, *beta_opt), "b", lw=2)

ax.set_xlim(1, 5)
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$f(x, \beta)$", fontsize=18)
plt.show()
```

```
[1.1966699 1.9791577]
0.035616612879187985
6.043983243778371
```



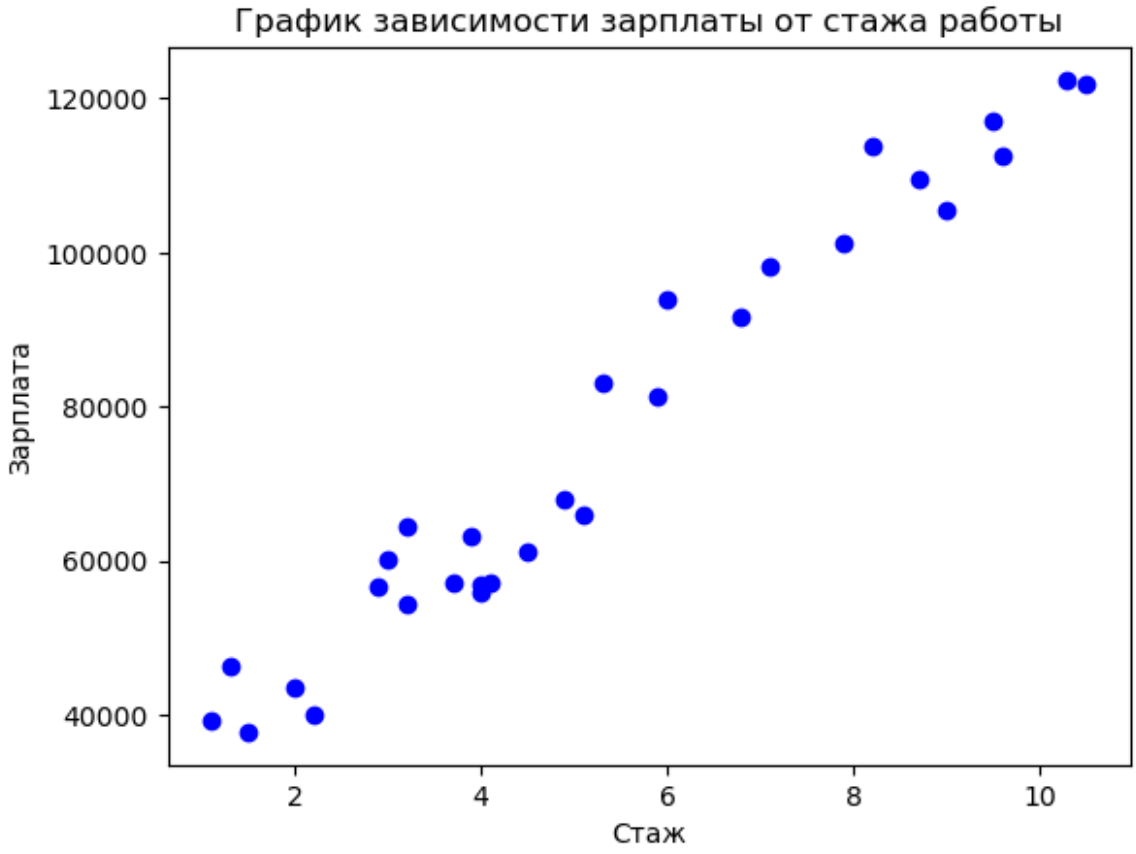

```
# Линейная регрессия
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas import DataFrame, Series
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

dataframe = pd.read_csv("Salary_Data.csv")
print("Размерность DataFrame: ", dataframe.shape)
print("Описание DataFrame'а: ")
print(dataframe.head(30))
print(dataframe.describe())

plt.title("График зависимости зарплаты от стажа работы")
plt.scatter(dataframe["YearsExperience"], dataframe["Salary"],
color="blue")
plt.xlabel("Стаж")
plt.ylabel("Зарплата")
plt.show()

x = dataframe.iloc[:, :-1].values
y = dataframe.iloc[:, -1].values

print("Стаж работы:", x, sep="\n")
print("Зарплата:", y, sep="\n")

# random_state = const означает, что при каждом запуске программы
данные
# для тестирования будут оставаться неизменными
# Берем 20 процентов от всего количество на тестирование и 80
процентов на обучение
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=0)

print("Набор стажа для обучения: ", x_train,
      "Набор стажа для тестирования:", x_test,
      "Набор зарплат для обучения:", y_train,
      "Набор зарплат для тестирования:", y_test, sep="\n")

# Далее обучаем модель линейной регрессии исходя из наших данных для
обучения
regressor = LinearRegression()

regressor.fit(x_train, y_train)

print("Intercept (свободный член):", regressor.intercept_,
      "Коэффициенты (наклоны):", regressor.coef_, sep="\n")

# Построим прогнозы

y_pred = regressor.predict(x_test)
```

```
df = DataFrame({"Actual": y_test, "Predicted": y_pred})

print("Сравниваем прогноз с тестовыми данными:", df, sep="\n")

df.plot(kind="bar")
plt.grid(which="major", linestyle="-", linewidth="0.5", color="green")
plt.grid(which="minor", linestyle=":", linewidth="0.5",
color="black")-
plt.show()

# Построим линию регрессии с тестовыми данными
plt.title("Линия регрессия исходя из тестовых признаков и
предсказанных значений")
plt.scatter(x_test, y_test, color="gray")
plt.plot(x_test, y_pred, color="red", linewidth=2)
plt.xlabel("Стаж")
plt.ylabel("Зарплата")
plt.show()
```

```
Размерность DataFrame:  (30, 2)
Описание DataFrame'a:
    YearsExperience    Salary
0               1.1   39343.0
1               1.3   46205.0
2               1.5   37731.0
3               2.0   43525.0
4               2.2   39891.0
5               2.9   56642.0
6               3.0   60150.0
7               3.2   54445.0
8               3.2   64445.0
9               3.7   57189.0
10              3.9   63218.0
11              4.0   55794.0
12              4.0   56957.0
13              4.1   57081.0
14              4.5   61111.0
15              4.9   67938.0
16              5.1   66029.0
17              5.3   83088.0
18              5.9   81363.0
19              6.0   93940.0
20              6.8   91738.0
21              7.1   98273.0
22              7.9  101302.0
23              8.2  113812.0
24              8.7  109431.0
25              9.0  105582.0
```

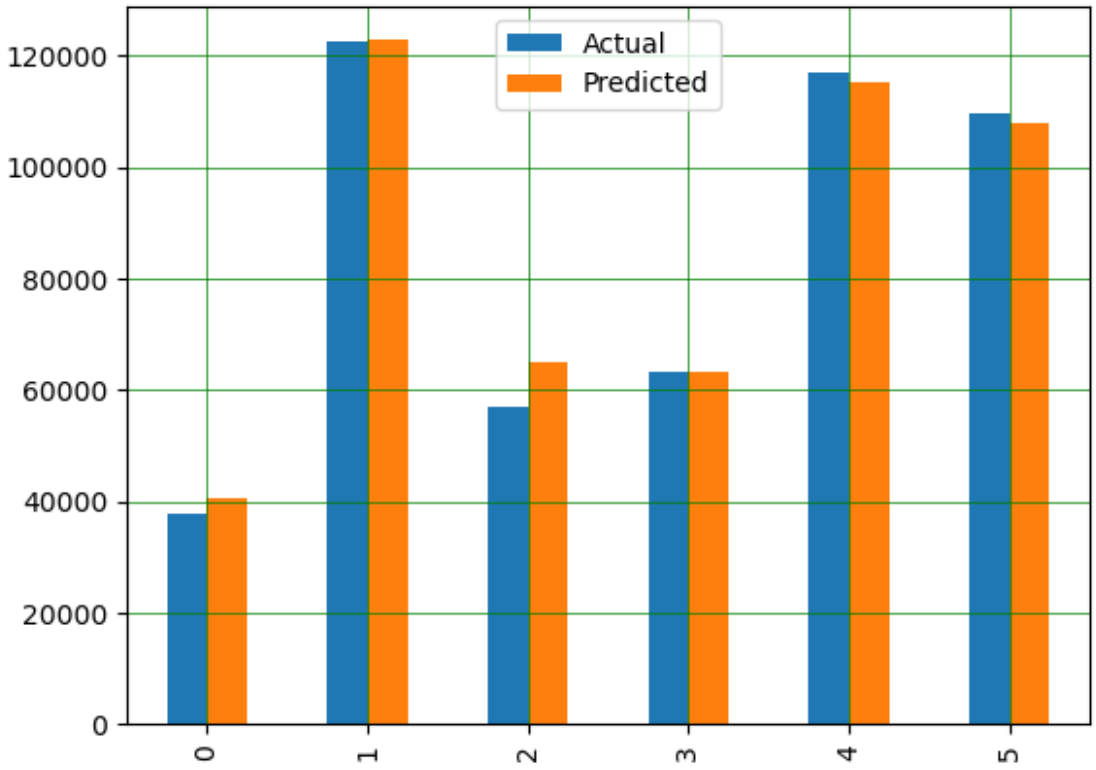
```
26              9.5  116969.0
27              9.6  112635.0
28             10.3  122391.0
29             10.5  121872.0
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000
```



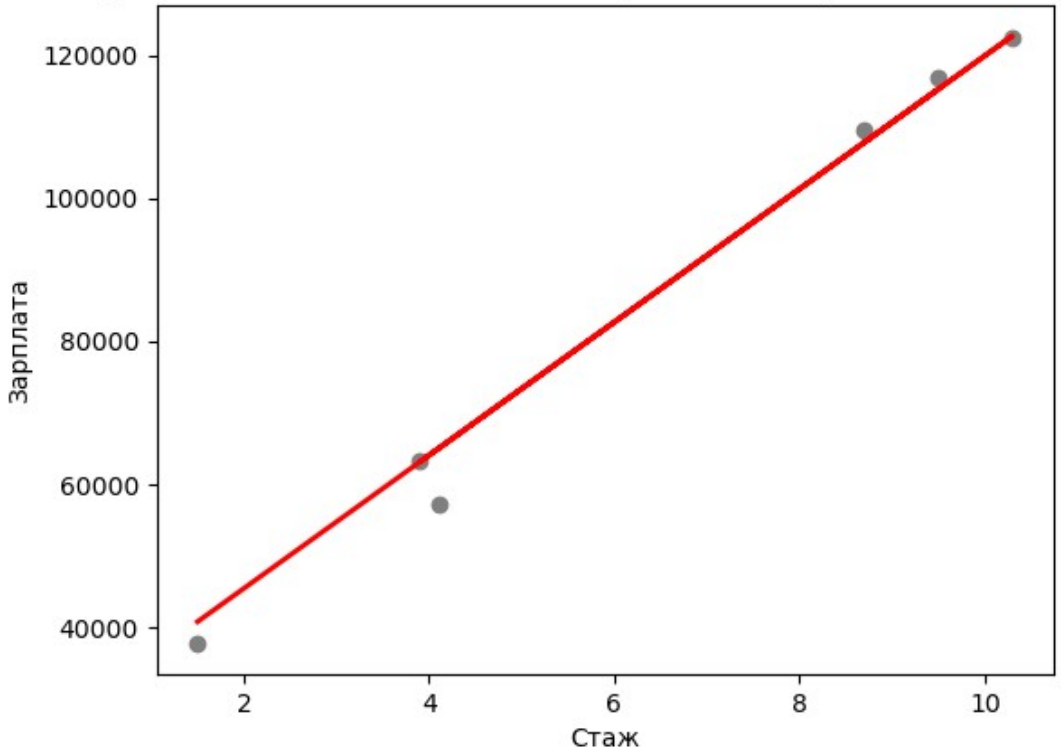

```
Стаж работы:
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]
 [ 2.9]
 [ 3. ]
```

```
 [ 3.2]
 [ 3.2]
 [ 3.7]
 [ 3.9]
 [ 4. ]
 [ 4. ]
 [ 4.1]
 [ 4.5]
 [ 4.9]
 [ 5.1]
 [ 5.3]
 [ 5.9]
 [ 6. ]
 [ 6.8]
 [ 7.1]
 [ 7.9]
 [ 8.2]
 [ 8.7]
 [ 9. ]
 [ 9.5]
 [ 9.6]
 [10.3]
 [10.5]]
Зарплата:
[ 39343.  46205.  37731.  43525.  39891.  56642.  60150.  54445.
64445.
  57189.  63218.  55794.  56957.  57081.  61111.  67938.  66029.
83088.
  81363.  93940.  91738.  98273. 101302. 113812. 109431. 105582.
116969.
 112635. 122391. 121872.]
Набор стажа для обучения:
[[ 9.6]
 [ 4. ]
 [ 5.3]
 [ 7.9]
 [ 2.9]
 [ 5.1]
 [ 3.2]
 [ 4.5]
 [ 8.2]
 [ 6.8]
 [ 1.3]
 [10.5]
 [ 3. ]
 [ 2.2]
 [ 5.9]
 [ 6. ]
 [ 3.7]
```

```
 [ 3.2]
 [ 9. ]
 [ 2. ]
 [ 1.1]
 [ 7.1]
 [ 4.9]
 [ 4. ]]
Набор стажа для тестирования:
[[ 1.5]
 [10.3]
 [ 4.1]
 [ 3.9]
 [ 9.5]
 [ 8.7]]
Набор зарплат для обучения:
[112635.  55794.  83088. 101302.  56642.  66029.  64445.  61111.
113812.
  91738.  46205. 121872.  60150.  39891.  81363.  93940.  57189.
54445.
 105582.  43525.  39343.  98273.  67938.  56957.]
Набор зарплат для тестирования:
[ 37731. 122391.  57081.  63218. 116969. 109431.]
Intercept (свободный член):
26780.099150628157
Коэффициенты (наклоны):
[9312.57512673]
Сравниваем прогноз с тестовыми данными:
     Actual      Predicted
0   37731.0   40748.961841
1  122391.0  122699.622956
2   57081.0   64961.657170
3   63218.0   63099.142145
4  116969.0  115249.562855
5  109431.0  107799.502753
```



Линия регрессия исходя из тестовых признаков и предсказанных значений



```
import pandas as pd
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Задание: Постройте модель множественной линейной регрессии для
произвольных данных из нескольких столбцов.
# Для примера можно взять потребления газа (в миллионах галлонов) в 48
штатах США.
# Найдите коэффициенты множественной регрессии. Постройте прогноз.

dataframe = pd.read_csv("petrol_consumption.csv")
dataframe = dataframe[['Petrol_Consumption', 'Petrol_tax',
'Average_income']]
print("Dataframe first 5 lines:", dataframe.head(5), sep="\n")
print("Dataframe description: ", dataframe.describe(), sep="\n")
print("Dataframe shape: ", dataframe.shape)


x = dataframe[["Average_income", "Petrol_tax"]]
y = dataframe["Petrol_Consumption"]


# Разделим данные на обучающую и тестовую выборки
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=0)

print("Набор стажа для обучения: ", x_train,
      "Набор стажа для тестирования:", x_test,
      "Набор зарплат для обучения:", y_train,
      "Набор зарплат для тестирования:", y_test, sep="\n")

regressor = LinearRegression()

regressor.fit(x_train, y_train)

print("Intercept (свободный член):", regressor.intercept_,
      "Коэффициенты (наклоны):", regressor.coef_, sep="\n")

# Построим прогнозы

y_pred = regressor.predict(x_test)

df = DataFrame({"Actual": y_test, "Predicted": y_pred})

print("Сравниваем прогноз с тестовыми данными:", df, sep="\n")
print()
```

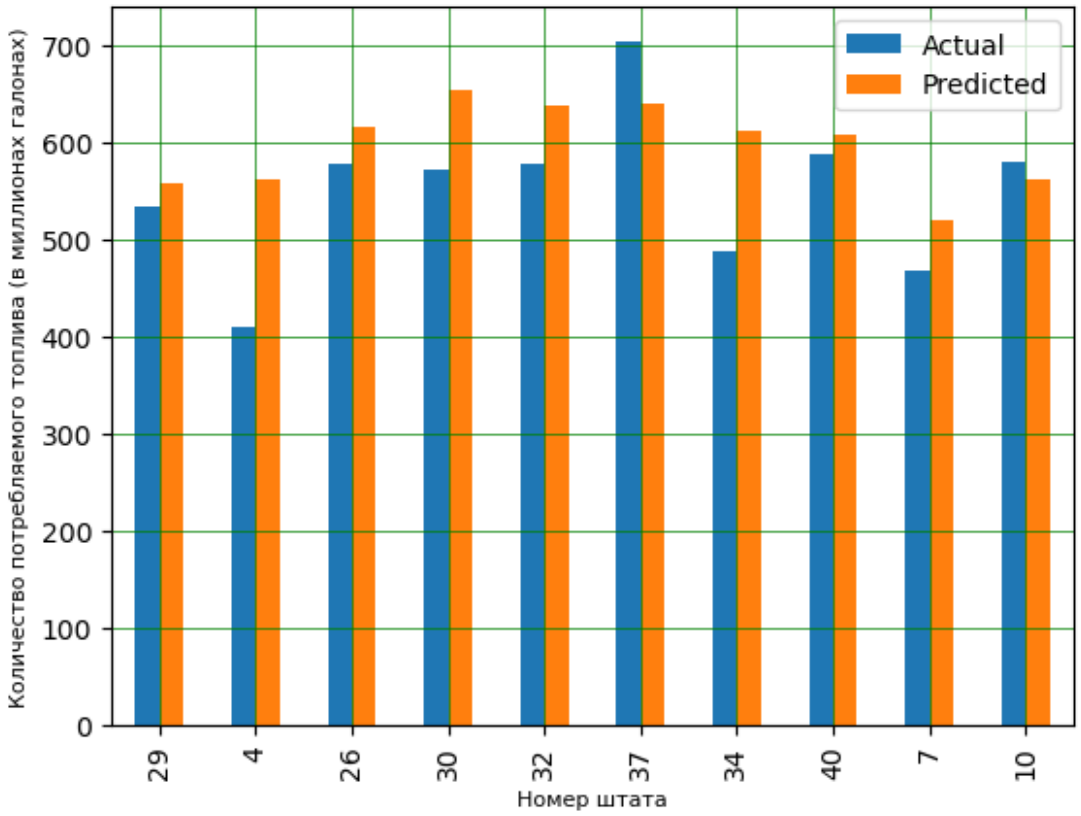
```
df.plot(kind="bar")
plt.grid(which="major", linestyle="-", linewidth="0.5", color="green")
plt.grid(which="minor", linestyle=":", linewidth="0.5", color="black")
plt.xlabel("Номер штата", size=8)
plt.ylabel("Количество потребляемого топлива (в миллионах галонах)",
size=8)
plt.show()

# Оцениваем производительность алгоритма. Для этого находим значение
RMSE
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Dataframe first 5 lines:
   Petrol_Consumption  Petrol_tax  Average_income
0                 541         9.0            3571
1                 524         9.0            4092
2                 561         9.0            3865
3                 414         7.5            4870
4                 410         8.0            4399
Dataframe description:
       Petrol_Consumption  Petrol_tax  Average_income
count           48.000000   48.000000       48.000000
mean           576.770833    7.668333     4241.833333
std            111.885816    0.950770      573.623768
min            344.000000    5.000000     3063.000000
25%            509.500000    7.000000     3739.000000
50%            568.500000    7.500000     4298.000000
75%            632.750000    8.125000     4578.750000
max            968.000000   10.000000     5342.000000
Dataframe shape:  (48, 3)
Набор стажа для обучения:
    Average_income  Petrol_tax
11            5126        7.50
31            3333        7.00
33            3357        7.50
27            3846        7.50
47            5002        7.00
2             3865        9.00
46            4296        7.00
18            4716        7.00
15            4318        7.00
28            4188        8.00
```

```
22              4897         9.00
16              4206         7.00
41              3656         7.00
20              4593         7.00
42              4300         7.00
8               4447         8.00
13              4207         7.00
25              3721         9.00
5               5342        10.00
17              3718         7.00
35              3802         6.58
14              4332         7.00
38              3635         8.50
1               4092         9.00
12              4817         7.00
43              3745         7.00
24              4574         8.50
6               5319         8.00
23              4258         9.00
36              4045         5.00
21              4983         8.00
19              4341         8.50
9               4512         7.00
39              4345         7.00
45              4476         9.00
3               4870         7.50
0               3571         9.00
44              5215         6.00
Набор стажа для тестирования:
    Average_income  Petrol_tax
29              3601         9.0
4               4399         8.0
26              3448         8.0
30              3640         7.0
32              3063         8.0
37              3897         7.0
34              3528         8.0
40              4449         7.0
7               5126         8.0
10              4391         8.0
Набор зарплат для обучения:
11    471
31    554
33    628
27    631
47    524
2     561
46    610
18    865
```

```
15    635
28    574
22    464
16    603
41    699
20    649
42    632
8     464
13    508
25    566
5     457
17    714
35    644
14    566
38    648
1     524
12    525
43    591
24    460
6     344
23    547
36    640
21    540
19    640
9     498
39    968
45    510
3     414
0     541
44    782
Name: Petrol_Consumption, dtype: int64
Набор зарплат для тестирования:
29    534
4     410
26    577
30    571
32    577
37    704
34    487
40    587
7     467
10    580
Name: Petrol_Consumption, dtype: int64
Intercept (свободный член):
1207.5206850328718
Коэффициенты (наклоны):
[ -0.05715924 -49.35500104]
Сравниваем прогноз с тестовыми данными:
    Actual   Predicted
```

```
29      534  557.495264
4       410  561.237194
26      577  615.595629
30      571  653.976056
32      577  637.601935
37      704  639.286132
34      487  611.022890
40      587  607.734234
7       467  519.682429
10      580  561.694468
```




```
Root Mean Squared Error (RMSE): 76.53885318427925
```

```python
import numpy as np

x = np.array([5.0, 5.2, 5.4, 5.6, 5.8, 6.0])
y = np.array([3.0, 2.0, 5.0, 2.0, 2.0, 3.0])

A = np.vstack([x, np.ones(len(x))]).T
B = np.vstack([x ** 2, x, np.ones(len(x))]).T
s = np.linalg.lstsq(B, y, rcond=-1)[0]
m, c = np.linalg.lstsq(A, y, rcond=None)[0]
print(f"Coefficients of first-degree polynomial: {m} {c}")
```

```
print(f"Coefficients of second-degree polynomial: {s}")


y_pred2 = x ** 2 * s[0] + x * s[1] + s[2]
y_pred1 = x * m + c
print("MSE for polinom 1: ", np.sum((y_pred1 - y) ** 2) / 6)
print("MSE for polinom 2: ", np.sum((y_pred2 - y) ** 2) / 6)
```

```
Coefficients of first-degree polynomial: -0.42857142857142866
5.190476190476189
Coefficients of second-degree polynomial: [ -0.89285714   9.39285714 -
21.71428571]
MSE for polinom 1:  1.1174603174603177
MSE for polinom 2:  1.109523809523809
```