МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Отчет по лабораторной работе №3
«Исследование методов работы с матрицами и векторами с помощью библиотеки NumPy»

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-20-1
Бокань И.П. « » _____ 2022г.
Подпись студента _____
Работа защищена « »_____2022г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

1. Вывод (примеры)

```
# Массив 2D
arr2D = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]

print("- Двухмерный массив:")
print(arr2D)
```

```
- Двухмерный массив:
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
```

```
# Вектор
print("- Вектор:")
print("  1) Вектор-строка:")
print("    np.narray([1,2,3,4]) = \n", np.array([1,2,3,4]), end="\n\n")
print("  2) Вектор-столбец:")
print("    np.narray([1,2,3,4]) = \n", np.array([[1],[2],[3],[4]]), end="\n\n")
print("  3) Нулевой:")
print("    3.1) Вектор строка:")
print("      np.zeros((4)) = \n", np.zeros((4)), end="\n\n")
print("    3.2) Вектор столбец:")
print("      np.zeros((4)) = \n", np.zeros((4, 1)), end="\n\n")
print("  4) Единичный:")
print("    4.1) Вектор строка:")
print("      np.zeros((4)) = \n", np.ones((4)), end="\n\n")
print("    4.2) Вектор столбец:")
print("      np.zeros((4)) = \n", np.ones((4, 1)), end="\n\n")
```

```
- Вектор:
  1) Вектор-строка:
    np.narray([1,2,3,4]) =
 [1 2 3 4]

  2) Вектор-столбец:
    np.narray([1,2,3,4]) =
 [[1]
 [2]
 [3]
 [4]]

  3) Нулевой:
    3.1) Вектор строка:
      np.zeros((4)) =
 [0. 0. 0. 0.]

    3.2) Вектор столбец:
      np.zeros((4)) =
 [[0.]
 [0.]
 [0.]
 [0.]]
```

Рисунок 1.1 - Результаты примера вектора

```
# Матрица
print("- Квадратнаяы матрица:")
print("  1) Создать:")
print("    1.1) Квадратную матрицу с помощью метода array():")
print("      np.narray([[1, 2], [3, 4]]) = \n", np.array([[1, 2], [3, 4]]), end="\n\n")
print("    1.2) Отдельно и передать в функцию:")
print("      np.narray(arr2D) = \n", np.array(arr2D), end="\n\n")
print("    1.3) Построение объекта типа matrix с помощью одноименного метода:")
print("      np.matrix([[1, 2], [3, 4]]) = \n", np.matrix([[1, 2], [3, 4]]), end="\n\n")
print("    1.4) Доступен стиль Matlab, когда между элементами ставятся пробелы, а строки разделяются точкой с запятой:")
print("      np.matrix(\"1 2 3; 4 5 6; 7 8 9\") = \n", np.matrix("1 2 3; 4 5 6; 7 8 9"), end="\n\n")
```

```
- Квадратнаяы матрица:
  1) Создать:
    1.1) Квадратную матрицу с помощью метода array():
      np.narray([[1, 2], [3, 4]]) =
 [[1 2]
 [3 4]]

    1.2) Отдельно и передать в функцию:
      np.narray(arr2D) =
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

    1.3) Построение объекта типа matrix с помощью одноименного метода:
      np.matrix([[1, 2], [3, 4]]) =
 [[1 2]
 [3 4]]

    1.4) Доступен стиль Matlab, когда между элементами ставятся пробелы, а строки разделяются точкой с запятой:
      np.matrix("1 2 3; 4 5 6; 7 8 9") =
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Рисунок 1.2 - Результаты примера матрица

```
# Диагональная
print("- Диагональная матрица:")
print("  1) Построить вручную, задав только значения элементам на главной диагонали:")
print("      np.matrix([[1, 0], [0, 1]]) = \n", np.matrix([[1, 0], [0, 1]]), end="\n\n")
print("  2) Извлечем ее главную диагональ:")
print("      np.diag(np.matrix('1 2 3; 4 5 6; 7 8 9')) = \n", np.diag(np.matrix('1 2 3; 4 5 6; 7 8 9')), end="\n\n")
print("  3) Построим диагональную матрицу на базе полученной диагонали:")
print("      np.diag(np.matrix('1 2 3; 4 5 6; 7 8 9')) = \n", np.diag(np.diag(np.matrix('1 2 3; 4 5 6; 7 8 9'))), end="\n\n")
```

```
- Диагональная матрица:
  1) Построить вручную, задав только значения элементам на главной диагонали:
      np.matrix([[1, 0], [0, 1]]) =
 [[1 0]
 [0 1]]

  2) Извлечем ее главную диагональ:
      np.diag(np.matrix('1 2 3; 4 5 6; 7 8 9')) =
 [1 5 9]

  3) Построим диагональную матрицу на базе полученной диагонали:
      np.diag(np.matrix('1 2 3; 4 5 6; 7 8 9')) =
 [[1 0 0]
 [0 5 0]
 [0 0 9]]
```

```
# Единичная (диагональ)
print("- Единичная матрица:")
print("  1) Создать Единичная матрица на базе списка и передадим в качестве аргумента функции matrix():")
print("      np.matrix([[1, 0], [0, 1]]) = \n", np.matrix([[1, 0], [0, 1]]), end="\n\n")
print("  2) Для построения такого типа матриц в библиотеке Numpy есть специальная функция – eye():")
print("      np.eye(2) = \n", np.eye(2), end="\n\n")
print("  3) Можно получить с помощью функции identity():")
print("      np.identity(2) = \n", np.identity(2), end="\n\n")
```

```
- Единичная матрица:
  1) Создать Единичная матрица на базе списка и передадим в качестве аргумента функции matrix():
      np.matrix([[1, 0], [0, 1]]) =
 [[1 0]
 [0 1]]

  2) Для построения такого типа матриц в библиотеке Numpy есть специальная функция – eye():
      np.eye(2) =
 [[1. 0.]
 [0. 1.]]

  3) Можно получить с помощью функции identity():
      np.identity(2) =
 [[1. 0.]
 [0. 1.]]
```

```
# Нулевая (диагональ)
print("- Нулевая матрица:")
print("  1) Создать матрица и заполняет число ноль - zeros():")
print("      np.zeros((3, 3)) = \n", np.zeros((3, 3)), end="\n\n")
```

```
- Нулевая матрица:
  1) Создать матрица и заполняет число ноль - zeros():
      np.zeros((3, 3)) =
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Рисунок 1.3 - Результаты примера диагональной матрицы

```
# Задание матрицы в общем виде
print("- Задание матрицы в общем виде:")
print("  1) Если есть данные о содержимом матрицы, то создать ее можно используя списки:")
print("      np.matrix('1 2 3 4; 5 6 7 8') = \n", np.matrix('1 2 3 4; 5 6 7 8'), end="\n\n")
print("  2) Создать матрицу заданного размера с произвольным содержимым:")
print("      np.zeros((3, 5)) = \n", np.zeros((3, 5)), end="\n\n")
```

```
- Задание матрицы в общем виде:
  1) Если есть данные о содержимом матрицы, то создать ее можно используя списки:
      np.matrix('1 2 3 4; 5 6 7 8') =
 [[1 2 3 4]
 [5 6 7 8]]

  2) Создать матрицу заданного размера с произвольным содержимым:
      np.zeros((3, 5)) =
 [[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

Рисунок 1.4 - Результаты примера задание матрицы в общем виде

```
# Транспонирование
print("- Транспонирование матрицы:")
print("  1) Транспонируем матрицу с помощью метода transpose():")
print("      np.matrix('1 2 3 4; 5 6 7 8').transpose() = \n", np.matrix('1 2 3 4; 5 6 7 8').transpose(),
print("  2) Можно сокращенный вариант:")
print("      np.matrix('1 2 3 4; 5 6 7 8').T = \n", np.matrix('1 2 3 4; 5 6 7 8').T, end="\n\n")
print("  C) Свойство:")
print("      1) Дважды транспонированная матрица равна исходной матрице:")
print("      (np.matrix('1 2 3 4; 5 6 7 8').T).T = \n", (np.matrix('1 2 3 4; 5 6 7 8').T).T, end="\n\n")
print("      2) Транспонирование суммы матриц равно сумме транспонированных матриц:")
print("      (np.matrix('1 2 3; 4 5 6') + np.matrix('7 8 9; 10 11 12')).T = \n", (np.matrix('1 2 3; 4 5
print("      3) Транспонирование произведения матриц равно произведению транспонированных матриц расстав
print("      (np.matrix('1 2 3; 4 5 6').dot(np.matrix('7 8 9; 10 11 12'))).T = \n", (np.matrix('1 2 3; 4
print("      4) Транспонирование произведения матрицы на число равно произведению этого числа на транспо
print("      (np.matrix('1 2 3; 4 5 6') * 3).T = \n", (np.matrix('1 2 3; 4 5 6') * 3).T , end="\n\n")
print("      5) Определители исходной и транспонированной матрицы совпадают:")
print("      format( np.linalg.det(np.matrix('1 2; 3 4')), '.9g') = \n", format( np.linalg.det(np.matrix
```

```
- Транспонирование матрицы:
  1) Транспонируем матрицу с помощью метода transpose():
      np.matrix('1 2 3 4; 5 6 7 8').transpose() =
 [[1 5]
 [2 6]
 [3 7]
 [4 8]]

  2) Можно сокращенный вариант:
      np.matrix('1 2 3 4; 5 6 7 8').T =
 [[1 5]
 [2 6]
 [3 7]
 [4 8]]

  C) Свойство:
      1) Дважды транспонированная матрица равна исходной матрице:
      (np.matrix('1 2 3 4; 5 6 7 8').T).T =
 [[1 2 3 4]
 [5 6 7 8]]

      2) Транспонирование суммы матриц равно сумме транспонированных матриц:
      (np.matrix('1 2 3; 4 5 6') + np.matrix('7 8 9; 10 11 12')).T =
 [[ 8 14]
 [10 16]
 [12 18]]
```

Рисунок 1.5 - Результаты примера задание матрицы в общем виде

```
# Действия над матрицами
print("- Транспонирование матрицы:")
print("  C) Свойство:")
print("      1) Произведение единицы и любой заданной матрицы равно заданной матрице:")
print("      1 * np.matrix('1 2; 3 4') = \n", 1 * np.matrix('1 2; 3 4'), end="\n\n")
print("      2) Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы:")
print("      0 * np.matrix('1 2; 3 4') = \n", 0 * np.matrix('1 2; 3 4'), end="\n\n")
print("      3) Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел:")
print("      (2 + 3) * np.matrix('1 2; 3 4') = \n", (2 + 3) * np.matrix('1 2; 3 4'), end="\n\n")
print("      4) Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному
print("      (2 * 3) * np.matrix('1 2; 3 4') = \n", (2 * 3) * np.matrix('1 2; 3 4'), end="\n\n")
print("      5) Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число:")
print("      3 * (np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8')) = \n", 3 * (np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8')), end
```

```
- Транспонирование матрицы:
  C) Свойство:
      1) Произведение единицы и любой заданной матрицы равно заданной матрице:
      1 * np.matrix('1 2; 3 4') =
 [[1 2]
 [3 4]]

      2) Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы:
      0 * np.matrix('1 2; 3 4') =
 [[0 0]
 [0 0]]

      3) Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел:
      (2 + 3) * np.matrix('1 2; 3 4') =
 [[ 5 10]
 [15 20]]

      4) Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на пе
рвое число:
      (2 * 3) * np.matrix('1 2; 3 4') =
 [[ 6 12]
 [18 24]]

      5) Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число:
      3 * (np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8')) =
 [[18 24]
 [30 36]]
```

Рисунок 1.6 - Результаты примера действия над матрицам

```
# Сложение матриц
print("- Сложение матриц:")
print("  C) Свойство:")
print("      1) Коммутативность сложения. От перестановки матриц их сумма не изменяется:")
print("      np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8') = \n", np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8')
print("      2) Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котор
print("      (np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8')) + np.matrix('9 10; 11 12') = \n", (np.matrix('1 2;
print("      3) Для любой матрицы существует противоположная ей , такая, что их сумма является нулевой матрице:
print("      np.matrix('1 2; 3 4') + (-1) * np.matrix('5 6; 7 8') = \n", np.matrix('1 2; 3 4') + (-1) * np.matr
```

```
- Сложение матриц:
  C) Свойство:
      1) Коммутативность сложения. От перестановки матриц их сумма не изменяется:
      np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8') =
 [[ 6  8]
 [10 12]]

      2) Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта
ыполняться:
      (np.matrix('1 2; 3 4') + np.matrix('5 6; 7 8')) + np.matrix('9 10; 11 12') =
 [[15 18]
 [21 24]]

      3) Для любой матрицы существует противоположная ей , такая, что их сумма является нулевой матрице:
      np.matrix('1 2; 3 4') + (-1) * np.matrix('5 6; 7 8') =
 [[-4 -4]
 [-4 -4]]
```

Рисунок 1.7 - Результаты примера сложение матрица

```
# Умножение матриц
print("- Умножение матриц:")
print("  C) Свойство:")
print("      1) Ассоциативность умножения. Результат умножения матриц не зависит от порядка, в котором будет вып
print("      np.matrix('1 2; 3 4').dot(np.matrix('5 6; 7 8')).dot(np.matrix('9 10; 11 12')) = \n", np.matrix('1
print("      2) Дистрибутивность умножения. Произведение матрицы на сумму матриц равно сумме произведений матриц
print("      np.matrix('1 2; 3 4').dot(np.matrix('5 6; 7 8')) + np.matrix('1 2; 3 4').dot(np.matrix('9 10; 11 12
print("      3) Умножение матриц в общем виде не коммутативно. Это означает, что для матриц не выполняется прави
print("      np.matrix('1 2; 3 4') + (-1) * np.matrix('5 6; 7 8') = \n", np.matrix('1 2; 3 4') + (-1) * np.matri
print("      4) Произведение заданной матрицы на единичную равно исходной матрице:")
print("      np.matrix('1 0; 0 1').dot(np.matrix('1 2; 3 4')) = \n",  np.matrix('1 0; 0 1').dot(np.matrix('1 2;
print("      4) Произведение заданной матрицы на нулевую матрицу равно нулевой матрице:")
print("      np.matrix('0 0; 0 0').dot(np.matrix('1 2; 3 4')) = \n",  np.matrix('0 0; 0 0').dot(np.matrix('1 2;
```

```
- Умножение матриц:
  C) Свойство:
      1) Ассоциативность умножения. Результат умножения матриц не зависит от порядка, в котором будет выполнятьс
      np.matrix('1 2; 3 4').dot(np.matrix('5 6; 7 8')).dot(np.matrix('9 10; 11 12')) =
 [[ 413  454]
 [ 937 1030]]

      2) Дистрибутивность умножения. Произведение матрицы на сумму матриц равно сумме произведений матриц:
      np.matrix('1 2; 3 4').dot(np.matrix('5 6; 7 8')) + np.matrix('1 2; 3 4').dot(np.matrix('9 10; 11 12')) =
 [[ 50  56]
 [114 128]]

      3) Умножение матриц в общем виде не коммутативно. Это означает, что для матриц не выполняется правило неза
ведения от перестановки множителей:
      np.matrix('1 2; 3 4') + (-1) * np.matrix('5 6; 7 8') =
 [[-4 -4]
 [-4 -4]]

      4) Произведение заданной матрицы на единичную равно исходной матрице:
      np.matrix('1 0; 0 1').dot(np.matrix('1 2; 3 4')) =
 [[1 2]
 [3 4]]

      4) Произведение заданной матрицы на нулевую матрицу равно нулевой матрице:
      np.matrix('0 0; 0 0').dot(np.matrix('1 2; 3 4')) =
 [[0 0]
 [0 0]]
```

Рисунок 1.8 - Результаты примера умножения матрица

```
# Определитель матрицы
print("- Определитель матрицы:")
print("  1) Транспонируем матрицу с помощью метода transpose():")
print("      np.linalg.det(np.matrix('-4 -1 2; 10 4 -1; 8 3 1')) = \n",  "%.2f" % np.linalg.det(np.matrix('-4 -
print("  C) Свойство:")
print("      1) Ассоциативность умножения. \n         Результат умножения матриц не зависит от порядка, в котор
print("      round(np.linalg.det(np.matrix('-4 -1 2; 10 4 -1; 8 3 1')), 3) = \n", round(np.linalg.det(np.matrix
print("      2) Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен
print("      np.linalg.det(np.matrix('-4 -1 2; 0 0 0; 8 3 1')) = \n", np.linalg.det(np.matrix('-4 -1 2; 0 0 0;
print("      3) При перестановке строк матрицы знак ее определителя меняется на противоположный:")
print("      np.linalg.det(np.matrix('-4 -1 2; 0 0 0; 8 3 1')) = \n", round(np.linalg.det(np.matrix('-4 -1 2; 1
print("      4) Если матрица содержит пропорциональные строки, то ее определитель равен нулю:")
print("      2 * np.matrix('-4 -1 2; 10 4 -1; 8 3 1')[0, :] = \n", 2 * np.matrix('-4 -1 2; 10 4 -1; 8 3 1')[0,
```

```
- Определитель матрицы:
  1) Транспонируем матрицу с помощью метода transpose():
      np.linalg.det(np.matrix('-4 -1 2; 10 4 -1; 8 3 1')) =
 -14.00

  C) Свойство:
      1) Ассоциативность умножения.
         Результат умножения матриц не зависит от порядка, в котором будет выполняться эта операция:
      round(np.linalg.det(np.matrix('-4 -1 2; 10 4 -1; 8 3 1')), 3) =
 -14.0

      2) Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю:
      np.linalg.det(np.matrix('-4 -1 2; 0 0 0; 8 3 1')) =
 0.0

      3) При перестановке строк матрицы знак ее определителя меняется на противоположный:
      np.linalg.det(np.matrix('-4 -1 2; 0 0 0; 8 3 1')) =
 -14.0

      4) Если матрица содержит пропорциональные строки, то ее определитель равен нулю:
      2 * np.matrix('-4 -1 2; 10 4 -1; 8 3 1')[0, :] =
 [[-8 -2  4]]
```

Рисунок 1.9 - Результаты примера определение матрица

```
# Обратная матрица
print("- Обратная матрица:")
print("  1) обратной матрицы будем использовать функцию *inv()*:")
print("      np.linalg.inv(np.matrix('1 -3; 2 5')) = \n", np.linalg.inv(np.matrix('1 -3; 2 5')),
print("  C) Свойство:")
print("      1) Обратная матрица обратной матрицы есть исходная матрица:")
print("      np.linalg.inv(np.linalg.inv(np.matrix('1. -3.; 2. 5.'))) = \n", np.linalg.inv(np.li
print("      2) Обратная матрица обратной матрицы есть исходная матрица:")
print("      (np.linalg.inv(np.matrix('1. -3.; 2. 5.'))).T = \n", (np.linalg.inv(np.matrix('1. -
print("      3) Обратная матрица произведения матриц равна произведению обратных матриц:")
print("      np.linalg.inv(np.matrix('7. 6.; 1. 8.')).dot(np.linalg.inv(np.matrix('1. -3.; 2. 5.
```

```
- Обратная матрица:
  1) обратной матрицы будем использовать функцию *inv()*:
      np.linalg.inv(np.matrix('1 -3; 2 5')) =
 [[ 0.45454545  0.27272727]
 [-0.18181818  0.09090909]]

  C) Свойство:
      1) Обратная матрица обратной матрицы есть исходная матрица:
      np.linalg.inv(np.linalg.inv(np.matrix('1. -3.; 2. 5.'))) =
 [[ 1. -3.]
 [ 2.  5.]]

      2) Обратная матрица обратной матрицы есть исходная матрица:
      (np.linalg.inv(np.matrix('1. -3.; 2. 5.'))).T =
 [[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]

      3) Обратная матрица произведения матриц равна произведению обратных матриц:
      np.linalg.inv(np.matrix('7. 6.; 1. 8.')).dot(np.linalg.inv(np.matrix('1. -3.; 2. 5.'))) =
 [[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
```

Рисунок 1.10 - Результаты примера обратная матрица

```
# Ранг матрицы
print("- Ранг матрицы:")
print("  x1) Использует функция matrix_rank():")
print("      np.linalg.matrix_rank(np.eye(7)) = \n", np.linalg.matrix_rank(np.eye(7)), end="\n\n"
```

```
- Ранг матрицы:
  x1) Использует функция matrix_rank():
      np.linalg.matrix_rank(np.eye(7)) =
 7
```

Рисунок 1.11 - Результаты примера ранга матрицы

2. Вывод (пример собственный)

```
import csv
import numpy as np

def readData():
    X = []
    y = []
    with open('Housing.csv') as f:
        rdr = csv.reader(f)
        next(rdr)

        for line in rdr:
            xline = [1.0]
            for s in line[:-1]:
                xline.append(float(s))
            X.append(xline)
            y.append(float(line[-1]))
    return (X,y)

X0,y0 = readData()

d = len(X0)-10
X = np.array(X0[:d])
y = (np.array([y0[:d]])).T

XtX = np.dot(X.T,X)
Xty = np.dot(X.T,y)
l1 = np.linalg.solve(XtX,Xty)
print(l1)

for data,actual in zip(X0[d:],y0[d:]):
    x = np.array([data])
    prediction = np.dot(x, l1)
    print('Прогноз = ' + str(prediction[0,0]) + ' фактический = ' + str(actual))
```

```
[[-4.14106096e+03]
 [ 3.55197583e+00]
 [ 1.66328263e+03]
 [ 1.45465644e+04]
 [ 6.77755381e+03]
 [ 6.58750520e+03]
 [ 4.44683380e+03]
 [ 5.60834856e+03]
 [ 1.27979572e+04]
 [ 1.24091640e+04]
 [ 4.19931185e+03]
 [ 9.42215457e+03]]
Прогноз = 97360.65509691095 фактический = 82500.0
Прогноз = 71774.16590136985 фактический = 83000.0
Прогноз = 92359.0891976023 фактический = 84000.0
Прогноз = 77748.27423790596 фактический = 85000.0
Прогноз = 91015.59030664092 фактический = 85000.0
Прогноз = 97545.1179047323 фактический = 91500.0
Прогноз = 97360.65509691095 фактический = 94000.0
Прогноз = 106006.8007559811 фактический = 103000.0
Прогноз = 92451.6931269468 фактический = 105000.0
Прогноз = 73458.29493810149 фактический = 105000.0
```

Рисунок 2.1 - Результаты свойства матричный вычислений

```
import numpy as np

def cramer(mat, constant):
    D = np.linalg.det(mat)
    mat1 = np.array([constant, mat[:, 1], mat[:, 2]])
    mat2 = np.array([mat[:, 0], constant, mat[:, 2]])
    mat3 = np.array([mat[:, 0], mat[:, 1], constant])
    Dx = np.linalg.det([mat1, mat2, mat3])
    X = Dx/D
    print(X)


a = np.array([[10, 40, 70],
              [20, 50, 80],
              [30, 60, 80]])

b = np.array([300, 360, 390])

cramer(a,b)
```

```
[1. 2. 3.]
```

Рисунок 2.2 - Результаты методом Крамера

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

```
M6 = numpy.array([[1., 1., 1.], [1., -1., 1], [2, 1, 0]]) # Матрица (левая часть системы)
v6 = numpy.array([1., 1., 1.]) # Вектор (правая часть системы)

numpy.linalg.solve(M6, v6)
```

```
array([ 0.5, -0. ,  0.5])
```

```
mpl.rc('font', family='Verdana', size= 16)

w = np.linalg.solve(M6, v6) # запишем найденные коэффициенты в переменную
def f(x):
    return w[0]*x**2 + w[1]*x + w[2] # уравнение параболы


fig, ax = plt.subplots(figsize=(10,5))

x = np.linspace(-2,2,200)
ax.axis([-2., 2., 0., 2.])
ax.grid()
ax.plot(x, f(x), label = 'Парабола')
ax.plot(x, x, label = 'Биссектриса 1й\nкоординатной четверти')
ax.set_xlabel(u'x',{'fontname':'Arial', 'size': 24})
ax.set_ylabel(u'f(x)',{'fontname':'Arial', 'size': 24})
plt.plot([-1, 1], [1, 1], 'ro', label = 'Точки для\nпостроения графика')
ax.annotate('Точка\nкасания', xy=(1., 1.), xytext=(1.5, 0.5),
            arrowprops=dict(facecolor='black', shrink=0.05),
            )

ax.legend(bbox_to_anchor=(1.6, 1.))

plt.show()
```

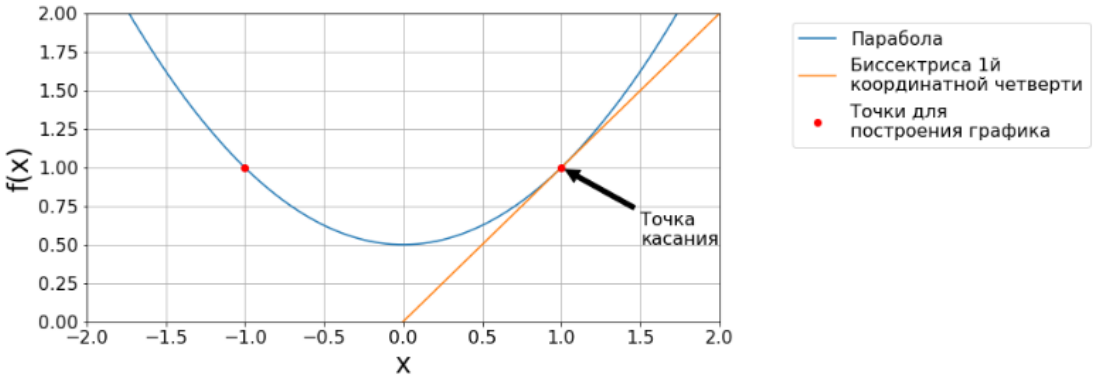

Рисунок 2.3 - Результаты линейных уравнений матричным