МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический
университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

по дисциплине

АНАЛИЗ И ВИЗУАЛИЗАЦИЯ ДАННЫХ

Индекс по учебному плану: **Б1.В.09**
Направление подготовки: **38.03.05 Бизнес-информатика**
Квалификация: **Бакалавр**
Профиль подготовки: **Информационные технологии в бизнесе**
Типы задач профессиональной деятельности: **проектный, аналитический**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

Практическая работа №1

Описание задачи

Задачи заключается в прогнозировании выручки компании в зависимости от уровня ее инвестиций в рекламу по TV, в газетах и по радио используя линейную регрессию и стохастический градиентный спуск.

Подключение библиотек

In [0]:

```
import pandas as pd
import numpy as np
```

Знакомство с данными

In [95]:

```
df = pd.read_csv(r'https://d3c33hcgiwev3.cloudfront.net/_739f9073ae55f970a4924e22bcc93124_advertising.csv?Expires=1563667200&Signature=O2ko6l3J-VMw~B72oE0EJQZkYuin-WwKcD1el8NhbA2ynWQEYr6QgkBY-zqZutjzWSEoa8RWMIPVhOgf9rG3CCI81cow~qqJ3OCXat7CzaxgcnywJ1IrFgGnBMGPZNKD6NtCBfR8AeBZx1~WtB~zhgMawIJjyJMPprTnkwWTkmA_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A')
df.head()
```

Out[95]:

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5

	TV	Radio	Newspaper	Sales
5	180.8	10.8	58.4	12.9

Текущий датасет представляет собой комбинацию инвестиций в 3 медийных канала (TV Radio Newspaper) и последующий уровень продаж Sales, зависимость которого от уровня инвестиций мы попробуем аппроксимировать.

In [96]:

```
print('Количество пропущенных значений в датасете: {}'.format(df.isna().sum().sum()))
df.info()
```

Количество пропущенных значений в датасете: 0

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
TV          200 non-null float64
Radio       200 non-null float64
Newspaper   200 non-null float64
Sales       200 non-null float64
dtypes: float64(4)
memory usage: 7.8 KB
```

Предобработка данных

Подготовим функцию для записи ответов в текстовый файл

In [0]:

```
def write_answer_to_file(answer, filename):
    with open(filename, 'w') as f_out:
        f_out.write(str(round(answer, 3)))
```

Создадим массивы NumPy X из столбцов TV, Radio и Newspaper и y - из столбца Sales

In [0]:

```
X = np.array(df.drop('Sales', axis=1).values)
y = np.array(df['Sales'])
```

Отмасштабируем столбцы матрицы X, вычтя из каждого значения среднее по соответствующему столбцу и поделив результат на стандартное отклонение

In [109]:

```
means, stds = np.mean(X,axis=0), np.std(X, axis=0)
means, stds
```

Out[109]:

```
(array([147.0425,  23.264 ,  30.554 ]),
 array([85.63933176, 14.80964564, 21.72410606]))
```

In [0]:

```
X = (X-means)/stds
```

Добавим к матрице X столбец из единиц. Вектор из единиц нужен для того, чтобы не обрабатывать отдельно коэффициент w0 линейной регрессии.

In [0]:

```
n = np.shape(X)[0]
ones = np.reshape(np.ones(n),(n,1))

X = np.hstack((X,ones))
```

Реализуем функцию mserror - среднеквадратичную ошибку прогноза. Она принимает два аргумента - объекты Series y (значения целевого признака) и y_pred (предсказанные значения)

In [0]:

```
def mserror(y, y_pred):
  return np.mean((y-y_pred)**2)
```

Какова среднеквадратичная ошибка прогноза значений Sales, если всегда предсказывать медианное значение Sales по исходной выборке?

In [113]:

answer1 = mserror(y, np.median(y))

print(answer1)

write_answer_to_file(answer1, '1.txt')

28.34575

Какие продажи предсказываются линейной моделью с весами, найденными с помощью нормального уравнения, в случае средних инвестиций в рекламу по ТВ, радио и в газетах? (то есть при нулевых значениях масштабированных признаков TV, Radio и Newspaper)

Реализуем функцию normal_equation, которая по заданным матрицам (массивам NumPy) X и y вычисляет вектор весов ww согласно нормальному уравнению линейной регрессии.

In [0]:

def normal_equation(X, y):

return np.linalg.solve(np.dot(X.transpose(),X), np.dot(X.transpose(),y))

In [116]:

norm_eq_weights = normal_equation(X, y)

print(norm_eq_weights)

[3.91925365 2.79206274 -0.02253861 14.0225]

In [117]:

*answer2 = np.sum([0., 0., 0., 1.]*norm_eq_weights)*

print(answer2)

write_answer_to_file(answer2, '2.txt')

14.022500000000003

Какова среднеквадратичная ошибка прогноза значений Sales в виде линейной модели с весами, найденными с помощью нормального уравнения?

Напишем функцию linear_prediction, которая принимает на вход матрицу X и вектор весов линейной модели w, а возвращает вектор прогнозов в виде линейной комбинации столбцов матрицы X с весами w

In [0]:

def linear_prediction(X, w):

```
    return np.dot(X,w)
```

In [119]:

```
lin_pred = linear_prediction(X,norm_eq_weights)
answer3 = mserror(y,lin_pred)
print(answer3)
write_answer_to_file(answer3, '3.txt')
```

2.784126314510936

Какова среднеквадратичная ошибка прогноза значений Sales в виде линейной модели с весами, найденными с помощью градиентного спуска?

Напишем функцию stochastic_gradient_step, реализующую шаг стохастического градиентного спуска для линейной регрессии. Функция должна принимать матрицу X, вектора y и w, число train_ind - индекс объекта обучающей выборки (строки матрицы X), по которому считается изменение весов, а также число η (eta) - шаг градиентного спуска (по умолчанию eta=0.01). Результатом будет вектор обновленных весов.

In [0]:

```
def stochastic_gradient_step(X, y, w, train_ind, eta=0.01):
    return w + 2 * eta/X.shape[0] * X[train_ind] * (y[train_ind] - linear_prediction(X[train_ind], w))
```

Напишите функцию stochastic_gradient_descent, реализующую стохастический градиентный спуск для линейной регрессии. Функция принимает на вход следующие аргументы:

X - матрица, соответствующая обучающей выборке

y - вектор значений целевого признака

w_init - вектор начальных весов модели

eta - шаг градиентного спуска (по умолчанию 0.01)

max_iter - максимальное число итераций градиентного спуска (по умолчанию 10000)

max_weight_dist - минимальное евклидово расстояние между векторами весов на соседних итерациях градиентного спуска, при котором алгоритм прекращает работу (по умолчанию 1e-8)

seed - число, используемое для воспроизводимости сгенерированных псевдослучайных чисел (по умолчанию 42)

verbose - флаг печати информации (например, для отладки, по умолчанию False)

На каждой итерации в вектор (список) должно записываться текущее значение среднеквадратичной ошибки. Функция должна возвращать вектор весов w, а также вектор (список) ошибок.

In [0]:

```
def stochastic_gradient_descent(X, y, w_init, eta=1e-2, max_iter=1e4,
                                min_weight_dist=1e-8, seed=42, verbose=False):
    # Инициализируем расстояние между векторами весов на соседних
    # итерациях большим числом.
    weight_dist = np.inf
    # Инициализируем вектор весов
    w = w_init
    # Сюда будем записывать ошибки на каждой итерации
    errors = []
    # Счетчик итераций
    iter_num = 0
    # Будем порождать псевдослучайные числа
    # (номер объекта, который будет менять веса), а для воспроизводимости
    # этой последовательности псевдослучайных чисел используем seed.
    np.random.seed(seed)

    # Основной цикл
    while weight_dist > min_weight_dist and iter_num < max_iter:
        # порождаем псевдослучайный
```

```
    # индекс объекта обучающей выборки
    random_ind = np.random.randint(X.shape[0])

    w_new = stochastic_gradient_step(X, y, w, random_ind, eta)
    weight_dist = np.linalg.norm(w-w_new)
    w = w_new
    errors.append(mserror(y, linear_prediction(X, w)))
    iter_num += 1


  return w, errors
```

Запустите 105 итераций стохастического градиентного спуска. Укажите вектор начальных весов w_init, состоящий из нулей. Оставьте параметры eta и seed равными их значениям по умолчанию (eta=0.01, seed=42 - это важно для проверки ответов).

In [122]:

```
%%time
stoch_grad_desc_weights, stoch_errors_by_iter = stochastic_gradient_descent(X, y, np.zeros(X.shape[1]),max_iter=1e5)
CPU times: user 2.94 s, sys: 0 ns, total: 2.94 s
Wall time: 2.94 s
```

Посмотрим, чему равна ошибка на первых 50 итерациях стохастического градиентного спуска. Видим, что ошибка не обязательно уменьшается на каждой итерации.

In [123]:

```
%pylab inline
plot(range(len(stoch_errors_by_iter)), stoch_errors_by_iter)
xlabel('Iteration number')
ylabel('MSE')
Populating the interactive namespace from numpy and matplotlib
```

/usr/local/lib/python3.6/dist-packages/IPython/core/magics/pylab.py:161:
UserWarning: pylab import has clobbered these variables: ['ones']
*`%matplotlib` prevents importing * from pylab and numpy*
*"\n`%matplotlib` prevents importing * from pylab and numpy"*

Out[123]:

Text(0, 0.5, 'MSE')

Посмотрим на вектор весов, к которому сошелся метод.

In [124]:

stoch_grad_desc_weights

Out[124]:

array([3.91069256e+00, 2.78209808e+00, -8.10462217e-03, 1.40190566e+01])

Посмотрим на среднеквадратичную ошибку на последней итерации.

In [125]:

stoch_errors_by_iter[-1]

Out[125]:

2.784412588406704

In [126]:

answer4 = mserror(y, linear_prediction(X, stoch_grad_desc_weights))
print(answer4)
write_answer_to_file(answer4, '4.txt')
2.784412588406704

Практическая работа №2

Постановка задачи

Представим, что международное круизное агентство "Carnival Cruise Line" решило себя разрекламировать с помощью баннеров и обратилось для этого к нам. Чтобы протестировать, велика ли от таких баннеров польза, их будет размещено всего 20 штук по всему миру. Нам нужно выбрать 20 таких локаций для размещения.

Агентство крупное, и у него есть несколько офисов по всему миру. Вблизи этих офисов оно и хочет разместить баннеры - легче договариваться и проверять результат. Также эти места должны быть популярны среди туристов.

Для поиска оптимальных мест воспользуемся базой данных крупнейшей социальной сети, основанной на локациях - Foursqare. Часть открытых данных есть, например, на сайте archive.org: https://archive.org/details/201309_foursquare_dataset_umn

Преобразование данных

Считаем скачанные данные и преобразуем в датафрейм Для удобной работы с этим данными удалим строки не содержащие координат - они неинформативны для нас:

id,user_id,venue_id,latitude,longitude,created_at

984222,15824,5222,38.8951118,-77.0363658,2012-04-21T17:43:47

984234,44652,5222,33.800745,-84.41052,2012-04-21T17:43:43

984291,105054,5222,45.5234515,-122.6762071,2012-04-21T17:39:22

...

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.cluster import MeanShift
```

In [2]:

df = pd.read_csv(r'D:\projects\coursera-specialization-Machine-learning-and-data-analysis\Поиск структуры в данных\checkins.dat', sep='|', skipinitialspace=True, skiprows=[1], low_memory=False, index_col=0)

df.head()

Out[2]:

	user_id	venue_id	latitude	longitude	created_at
id					
984301	2041916.0	5222.0	NaN	NaN	2012-04-21 17:39:01
984222	15824.0	5222.0	38.895112	-77.036366	2012-04-21 17:43:47
984315	1764391.0	5222.0	NaN	NaN	2012-04-21 17:37:18
984234	44652.0	5222.0	33.800745	-84.410520	2012-04-21 17:43:43
984249	2146840.0	5222.0	NaN	NaN	2012-04-21 17:42:58

In [3]:

df.columns = [column.strip() for column in df.columns]

df.dropna(inplace=True)

df.drop(['user_id', 'venue_id', 'created_at'], axis=1, inplace=True)

df.head()

Out[3]:

	latitude	longitude
id		
984222	38.895112	-77.036366
984234	33.800745	-84.410520
984291	45.523452	-122.676207

	latitude	longitude
id		
984318	40.764462	-111.904565
984232	33.448377	-112.074037

In [4]:

df = df.iloc[:100000]

Кластеризация

Теперь необходимо кластеризовать данные координаты, чтобы выявить центры скоплений туристов. Поскольку баннеры имеют сравнительно небольшую площадь действия, нам нужен алгоритм, позволяющий ограничить размер кластера и не зависящий от количества кластеров.

Используйте MeanShift, указав bandwidth=0.1, что в переводе из градусов в метры колеблется примерно от 5 до 10 км в средних широтах.

Примечание: на 396634 строках, кластеризация будет работать долго (порядка часа). Для получения корректного ответа достаточно и 100000 (~2 минуты на "среднем" ноутбуке).

In [5]:

MS = MeanShift(bandwidth = 0.1)

MS.fit(df)

Out[5]:

MeanShift(bandwidth=0.1, bin_seeding=False, cluster_all=True, min_bin_freq=1,

n_jobs=None, seeds=None)

In [6]:

*labels = MS.labels_*

*cluster_centers = MS.cluster_centers_*

Некоторые из получившихся кластеров содержат слишком мало точек - такие кластеры не интересны рекламодателям. Поэтому надо определить,

какие из кластеров содержат, скажем, больше 15 элементов. Центры этих кластеров и являются оптимальными для размещения.

При желании увидеть получившиеся результаты на карте, можно передать центры получившихся кластеров в один из инструментов визуализации. Например, сайт mapcustomizer.com имеет функцию Bulk Entry, куда можно вставить центры полученных кластеров в формате:

38.8951118,-77.0363658

33.800745,-84.41052

45.5234515,-122.6762071

...

In [7]:

df['cluster'] = MS.predict(df)

In [8]:

cluster_size = pd.DataFrame(df.pivot_table(index = 'cluster', aggfunc = 'count', values = 'latitude'))

cluster_size.columns = ['clust_size']

In [9]:

cluster_centers_df = pd.DataFrame(cluster_centers)

cluster_centers_df.columns = ['cent_latitude', 'cent_longitude']

In [10]:

cluster_df = cluster_centers_df.join(cluster_size)

cluster_df.to_csv('clusters.csv', index = None)

cluster_df = cluster_df[cluster_df.clust_size > 15]

cluster_df.head()

Out[10]:

	cent_latitude	cent_longitude	clust_size
0	40.717716	-73.991835	12506
1	33.449438	-112.002140	4692

	cent_latitude	cent_longitude	clust_size
2	33.446380	-111.901888	3994
3	41.878244	-87.629843	3363
4	37.688682	-122.409330	3526

Как помним, 20 баннеров надо разместить близ офисов компании. Найдем на Google Maps по запросу "Carnival Cruise Line" адреса офисов:

33.751277, -118.188740 (Los Angeles)

25.867736, -80.324116 (Miami)

51.503016, -0.075479 (London)

52.378894, 4.885084 (Amsterdam)

39.366487, 117.036146 (Beijing)

-33.868457, 151.205134 (Sydney)

Осталось определить 20 ближайших к ним центров кластеров. Т.е. посчитать дистанцию до ближайшего офиса для каждой точки и выбрать 20 с наименьшим значением.

Примечание: при подсчете расстояний и в кластеризации можно пренебречь тем, что Земля круглая, так как в точках, расположенных близко друг к другу погрешность мала, а в остальных точках значение достаточно велико.

In [11]:

```
def get_distance(lat1, lon1, lat2, lon2):
    return ((lat1 - lat2)**2 + (lon1 - lon2)**2) ** 0.5


office_coordinates = [
    (33.751277, -118.188740),
    (25.867736, -80.324116),
    (51.503016, -0.075479),
    (52.378894, 4.885084),
```

(39.366487, 117.036146),

(-33.868457, 151.205134)

]

def get_min_distance_to_office(lat, lon):

min_dist = None

for (of_lat, of_lon) in office_coordinates:

dist = get_distance(lat, lon, of_lat, of_lon)

if (min_dist is None) or (dist < min_dist):

min_dist = dist

return min_dist

In [12]:

cluster_df.head()

cluster_df['min_distance'] = np.vectorize(get_min_distance_to_office)(cluster_df.cent_latitude, cluster_df.cent_longitude)

In [13]:

cluster_df.sort_values('min_distance')[:20]

Out[13]:

	cent_latitude	cent_longitude	clust_size	min_distance
420	-33.860630	151.204776	28	0.007835
370	52.372964	4.892317	31	0.009353
419	25.845672	-80.318891	28	0.022674
58	51.502991	-0.125537	254	0.050058
51	33.809878	-118.148924	281	0.070848
29	25.785812	-80.217938	564	0.134109
167	25.705350	-80.283429	80	0.167406
92	26.010098	-80.199991	138	0.188876

	cent_latitude	cent_longitude	clust_size	min_distance
87	33.888325	-118.048928	100	0.195779
42	33.872986	-118.362091	384	0.211811
291	33.972575	-118.168371	37	0.222233
320	26.138844	-80.334347	38	0.271301
119	33.983936	-118.007405	74	0.294979
55	26.120863	-80.158907	246	0.302270
27	33.817306	-117.891249	577	0.304731
11	34.060398	-118.248709	1081	0.314884
32	33.674303	-117.858789	449	0.338810
159	26.200585	-80.250716	42	0.340846
17	34.035487	-118.438998	645	0.378688
47	34.131460	-118.118012	273	0.386706

Практическая работа № 3

Фундаментальная задача программирования — вычисление математических и, в частности, алгебраических функций. Запись выражения на языке математики не принимается напрямую языком программирования. Выражение нужно написать в виде, который будет понятен тому или иному языку программирования.

Например, y = x², должно быть записано как `y = x*x` или `y = x**2`.

Упражнение №1

Запишите выражение, заданное формулой, в виде, подходящем для языка Python.

$$y = log_{1+x^2} \frac{e^{\frac{1}{sinx+1}}}{\frac{5}{4} + \frac{1}{x^{15}}}$$

и найдите его значения в точках 1, 10, 1000.

Для вычисления математических функций мы не будем использовать стандартную библиотеку **math**. Т.к. она не работает с векторами. В нашем случае разумней обратить внимание на библиотеку **numpy**. Данная библиотека обеспечивает удобную работу с векторам.

Т.е., если у нас есть вектор x=[1, 2, 3, 4] и мы вызовим numpy.log(x), то логарифм будет взят от каждого элемента списка и возвращен будет список значений.

Аналогичная функция в модуля math ожидает число, т.е. нельзя сделать math.log(x), нужно делать math.log(x[0]) и т.д.

Традиционно библиотека numpy подключается командой:

```
import numpy as np
```

Данный вызов сообщает, что подключить numpy под псевдонимом np. Это делается, чтобы не писать каждый раз:

```
numpy.cos(x)
```

А писать:

```
np.cos(x)
```

Такой код, с более коротким именем библиотеки, элементарно, проще читать.

Основные математические функции и константы функии, которые нам понадобятся из numpy:

Функция библиотеки math	**Математическая функция**
np.pi	Число pi
np.e	Число e
np.cos	Косинус
np.sin	Синус
np.tan	Тангенс
np.acos	Арккосинус
np.asin	Арксинус
np.atan	Арктангенс
np.exp	Экспонента
np.log	Логарифм

Функция log вычисляет натуральный логарифм. Чтобы вычислить логарифм по другому основанию, нужно воспользоваться формулой перехода. Например, если мы хотим получить логарифм x по основанию 2, нужно написать:

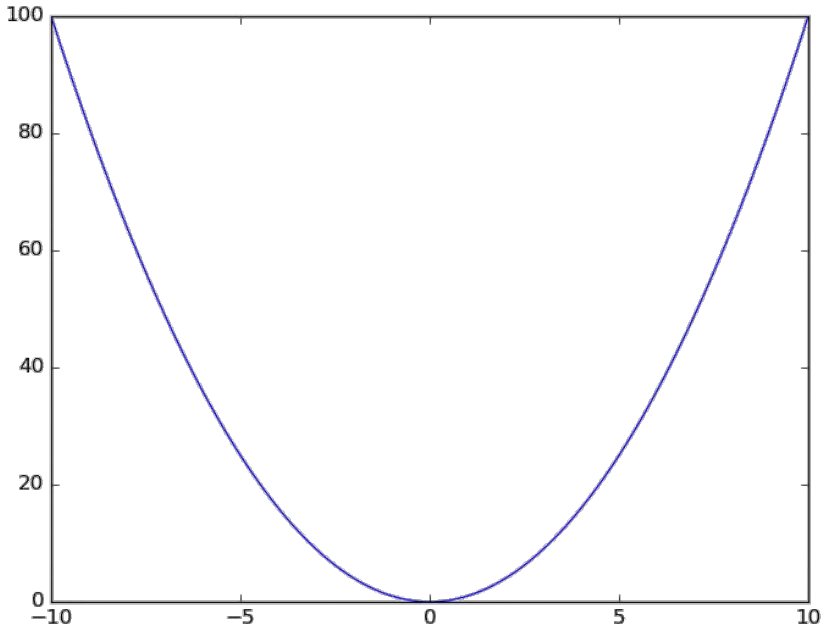
```
np.log(x) / np.log(2)
```

Построение графиков

matplotlib - набор дополнительных модулей (библиотек) языка Python. Предоставляет средства для построения самых разнообразных 2D графиков и диаграмм данных. Отличается простотой использования — для построения весьма сложных и красочно оформленных диаграмм достаточно нескольких строк кода. При этом качество получаемых изображений более чем достаточно для их публикования. Также позволяет сохранять результаты в различных форматах, например Postscript, и, соответственно, вставлять изображения в документы TeX. Предоставляет API для встраивания своих графических объектов в приложения пользователя.

Пример построения графика функции:

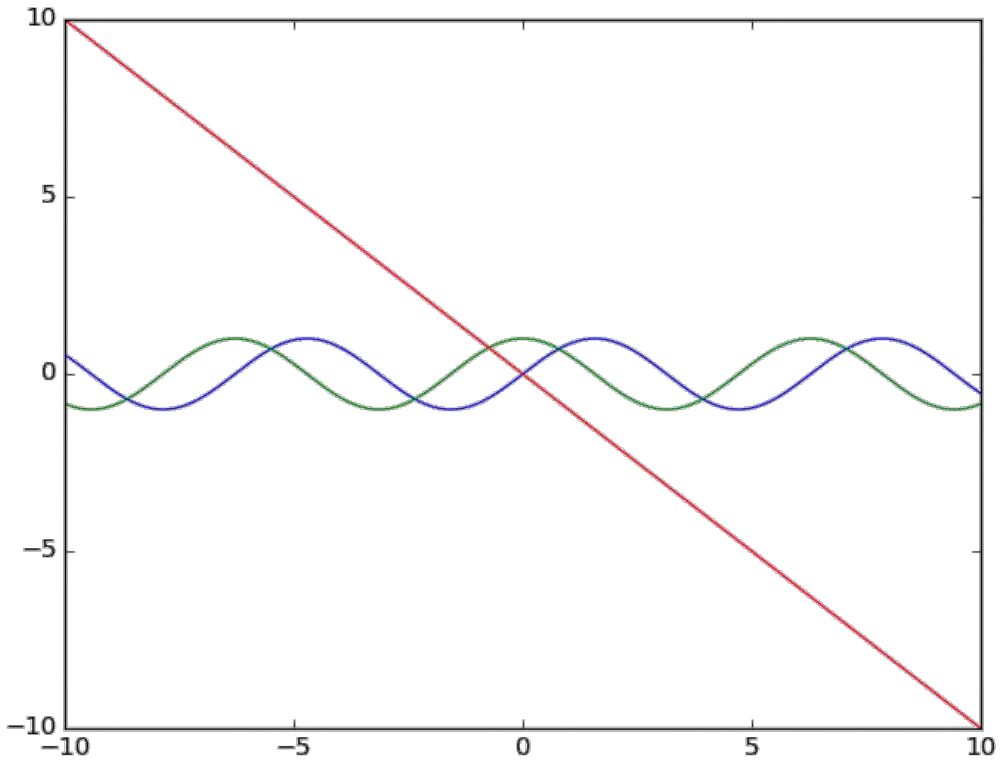
```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-10, 10.01, 0.01)
plt.plot(x, x**2)
plt.show()
```



На одном рисунке можно построить несколько графиков функций:

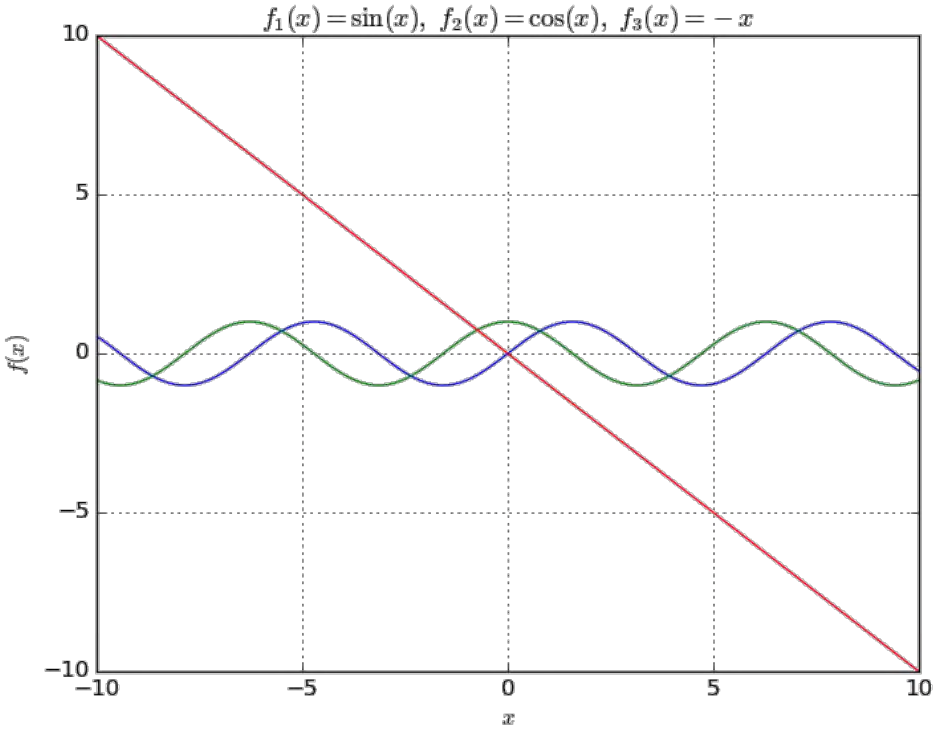
```
import numpy as np
import matplotlib.pyplot as plt
```

```
x = np.arange(-10, 10.01, 0.01)
plt.plot(x, np.sin(x), x, np.cos(x), x, -x)
plt.show()
```



Также довольно просто на график добавить служебную информацию и отобразить сетку:

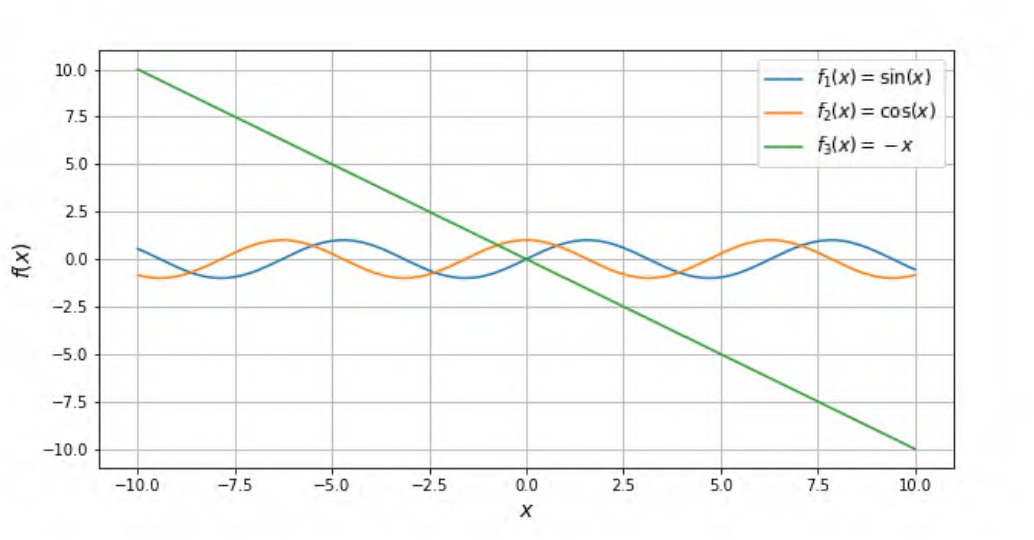
```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-10, 10.01, 0.01)
plt.plot(x, np.sin(x), x, np.cos(x), x, -x)
plt.xlabel(r'$x$')
plt.ylabel(r'$f(x)$')
plt.title(r'$f_1(x)=\sin(x),\ f_2(x)=\cos(x),\ f_3(x)=-x$')
plt.grid(True)
plt.show()
```



Или используя legend(), где можно указать место расположения подписей к кривым на графике в параметре loc. Подписи могут быть явно переданы legend((line1, line2, line3), ('label1', 'label2', 'label3')) или могут быть переданы в аргумет label, как в примере ниже. Чтобы сохранить график нужно воспользоваться savefig(figure_name), где figure_name является строкой назания файла с указанием расширения. Для текстовых полей можно изменять шрифт (fontsize), для большей читаемости графика, а его размер указывается с помощью figure(figsize=(10, 5)).

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-10, 10.01, 0.01)
plt.figure(figsize=(10, 5))
plt.plot(x, np.sin(x), label=r'$f_1(x)=\sin(x)$')
plt.plot(x, np.cos(x), label=r'$f_2(x)=\cos(x)$')
plt.plot(x, -x, label=r'$f_3(x)=-x$')
plt.xlabel(r'$x$', fontsize=14)
plt.ylabel(r'$f(x)$', fontsize=14)
```

```
plt.grid(True)
plt.legend(loc='best', fontsize=12)
plt.savefig('figure_with_legend.png')
plt.show()
```



Текстовые поля в matplotlib могут содержать разметку **LaTeX**, заключенную в знаки $. Буква r перед кавычками говорит python, что символ "\" следует оставить как есть и не интерпретировать как начало спецсимвола (например, перевода строки - "\n").

Работа с matplotlib основана на использовании графических окон и осей (оси позволяют задать некоторую графическую область). Все построения применяются к текущим осям. Это позволяет изображать несколько графиков в одном графическом окне. По умолчанию создаётся одно графическое окно figure(1) и одна графическая область subplot(111) в этом окне. Команда subplot позволяет разбить графическое окно на несколько областей. Она имеет три параметра: nr, nc, np. Параметры nr и nc определяют количество строк и столбцов на которые разбивается графическая область, параметр np определяет номер текущей области (np принимает значения от 1 до nr*nc). Если nr*nc<10, то передавать параметры nr, nc, np можно без использования запятой. Например, допустимы формы subplot(2,2,1) и subplot(221).

```
import numpy as np
```

```
    import matplotlib.pyplot as plt
    x = np.arange(-10, 10.01, 0.01)
t = np.arange(-10, 11, 1)

    #subplot 1
    sp = plt.subplot(221)
    plt.plot(x, np.sin(x))
    plt.title(r'$\sin(x)$')
    plt.grid(True)

    #subplot 2
    sp = plt.subplot(222)
    plt.plot(x, np.cos(x), 'g')
    plt.axis('equal')
    plt.grid(True)
    plt.title(r'$\cos(x)$')

    #subplot 3
    sp = plt.subplot(223)
    plt.plot(x, x**2, t, t**2, 'ro')
    plt.title(r'$x^2$')

    #subplot 4
    sp = plt.subplot(224)
    plt.plot(x, x)
    sp.spines['left'].set_position('center')
    sp.spines['bottom'].set_position('center')
    plt.title(r'$x$')

    plt.show()
```

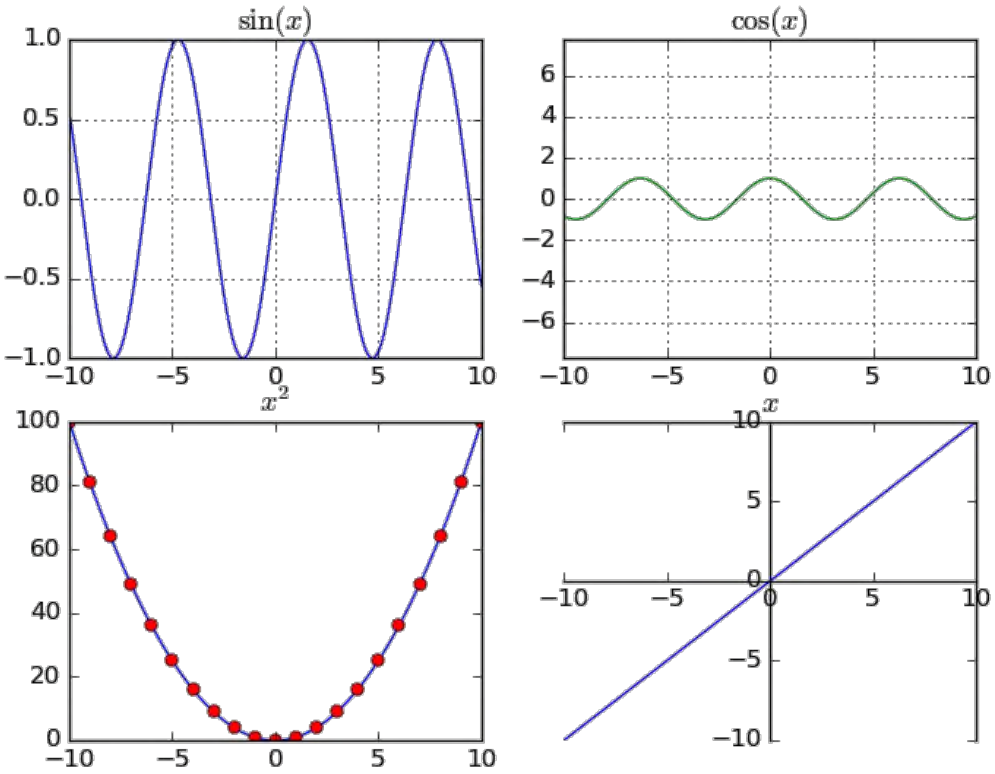
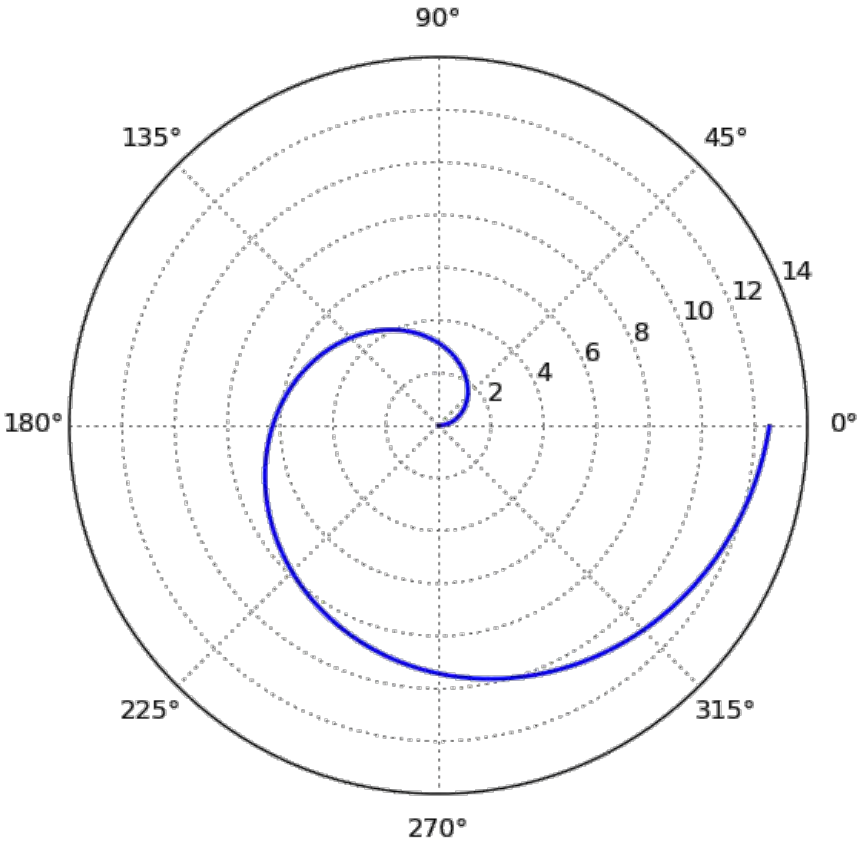


График может быть построен в полярной системе координат, для этого при создании subplot необходимо указать параметр polar=True:

```
import numpy as np
import matplotlib.pyplot as plt
plt.subplot(111, polar=True)
phi = np.arange(0, 2*np.pi, 0.01)
rho = 2*phi
plt.plot(phi, rho, lw=2)
plt.show()
```



Или может быть задан в параметрической форме (для этого не требуется никаких дополнительных действий, поскольку два массива, которые передаются в функцию plot воспринимаются просто как списки координат точек, из которых состоит график):

```
import numpy as np
import matplotlib.pyplot as plt
t = np.arange(0, 2*np.pi, 0.01)
r = 4
plt.plot(r*np.sin(t), r*np.cos(t), lw=3)
plt.axis('equal')
plt.show()
```

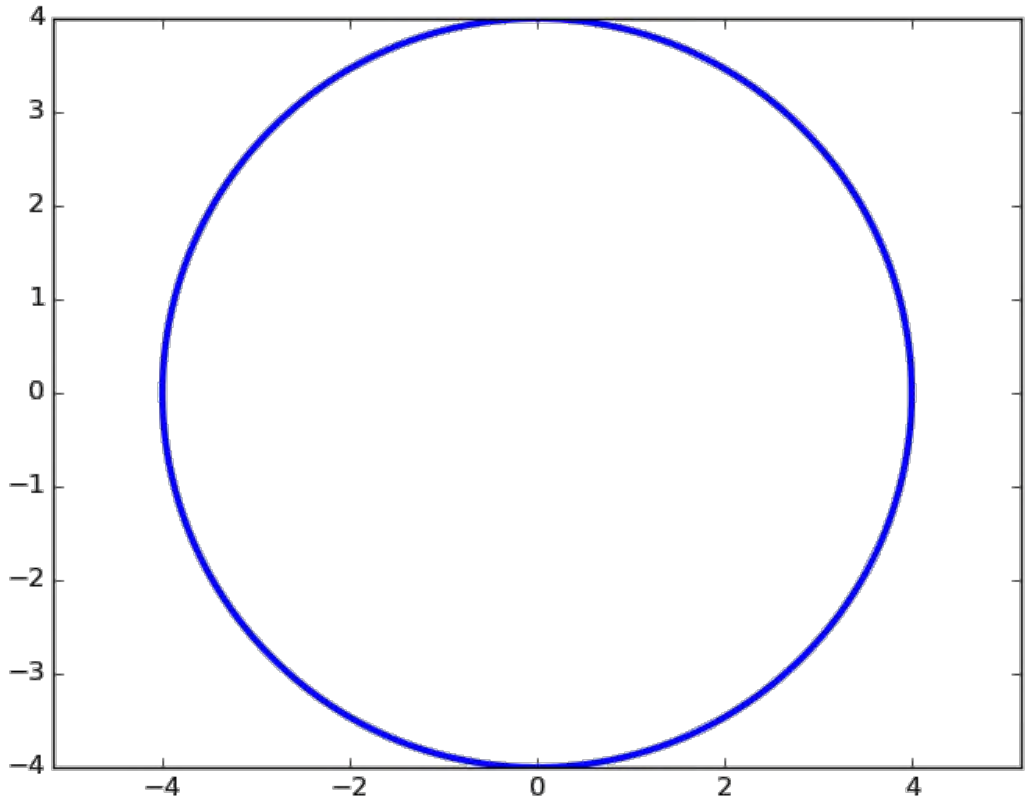
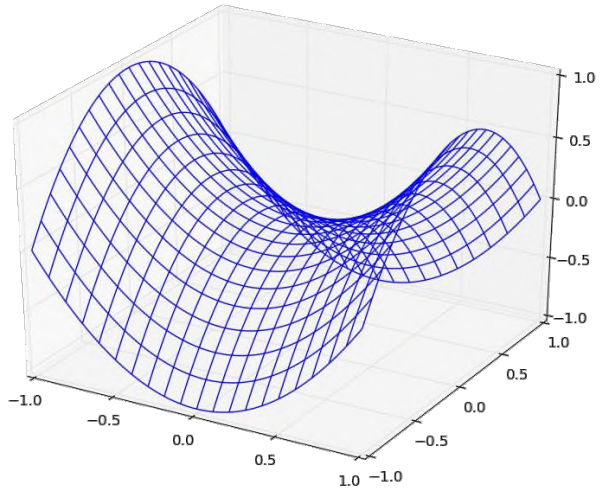


График функции двух переменных может быть построен, например, так:

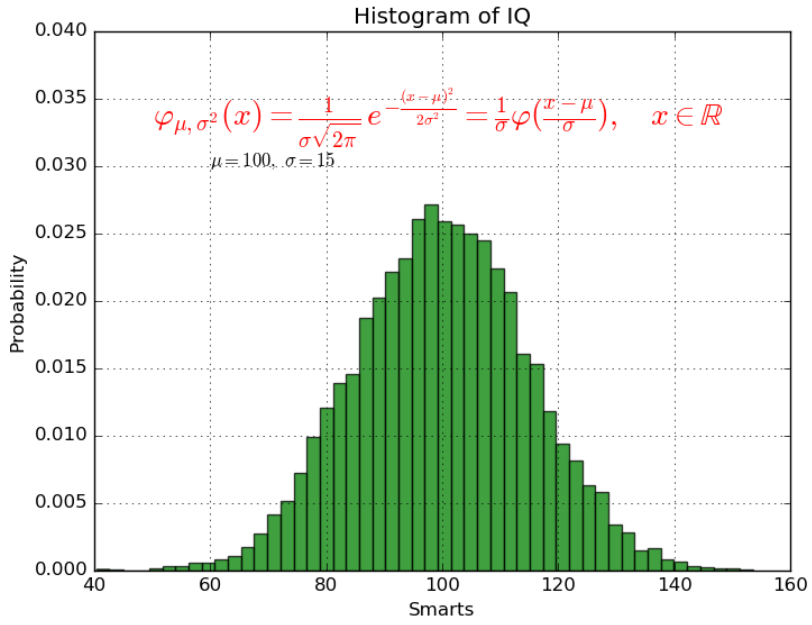
```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
ax = axes3d.Axes3D(plt.figure())
i = np.arange(-1, 1, 0.01)
X, Y = np.meshgrid(i, i)
Z = X**2 - Y**2
ax.plot_wireframe(X, Y, Z, rstride=10, cstride=10)
plt.show()
```



Добавление текста на график: Команду text() можно использовать для добавления текста в произвольном месте (по умолчанию координаты задаются в координатах активных осей), а команды xlabel(), ylabel() и title() служат соответственно для подписи оси абсцисс, оси ординат и всего графика. Для более полной информации смотрите **«Text introduction»** раздел на оф. сайте.

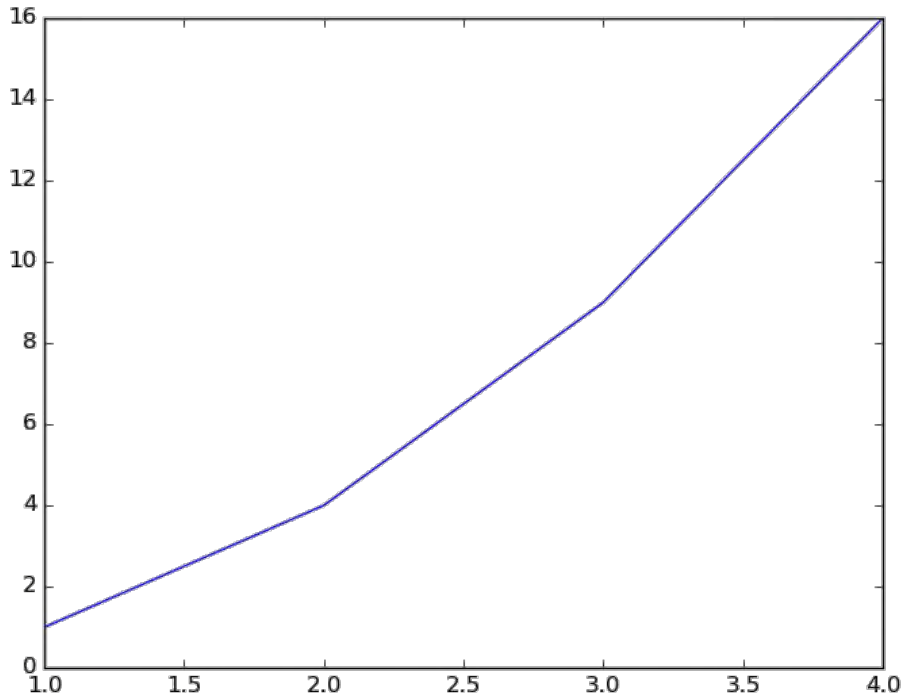
```
import numpy as np
import matplotlib.pyplot as plt
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)
# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .030, r'$\mu=100,\ \sigma=15$')
plt.text(50, .033, r'$\varphi_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} \,e^{ -\frac{(x- \mu)^2}{2\sigma^2}} = \frac{1}{\sigma} \varphi\left(\frac{x - \mu}{\sigma}\right),\quad x\in\mathbb{R}$', fontsize=20, color='red')
plt.axis([40, 160, 0, 0.04])
```

```
plt.grid(True)
plt.show()
```



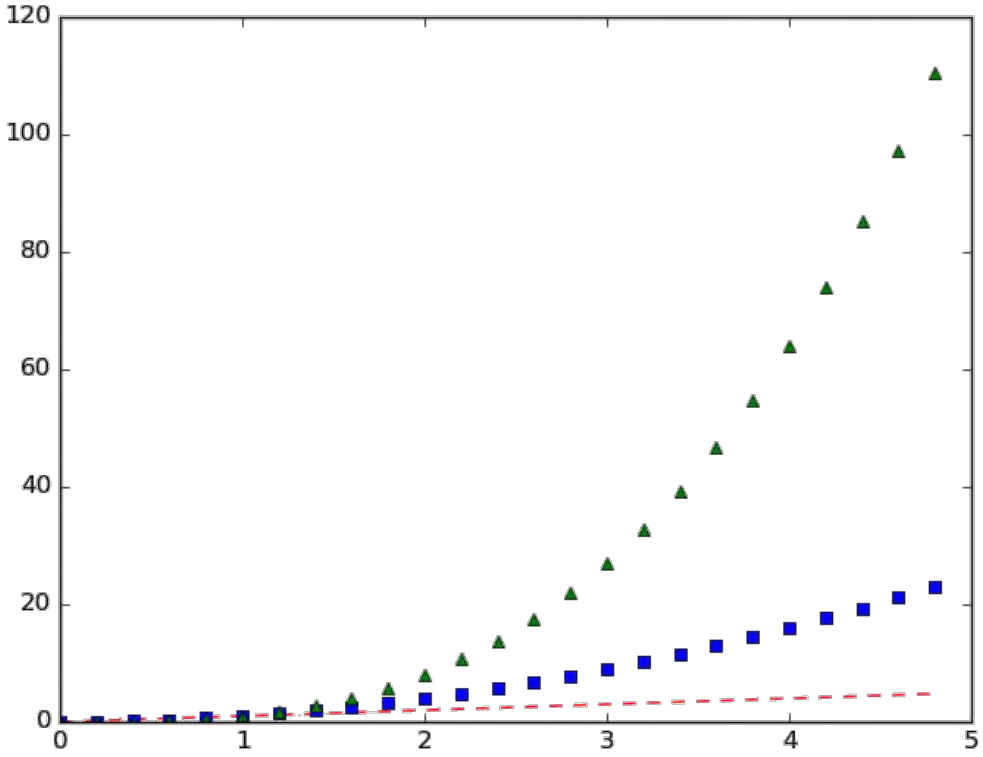
plot() — универсальная команда и в неё можно передавать произвольное количество аргументов. Например, для того, чтобы отобразить y в зависимости от x, можно выполнить команду:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```



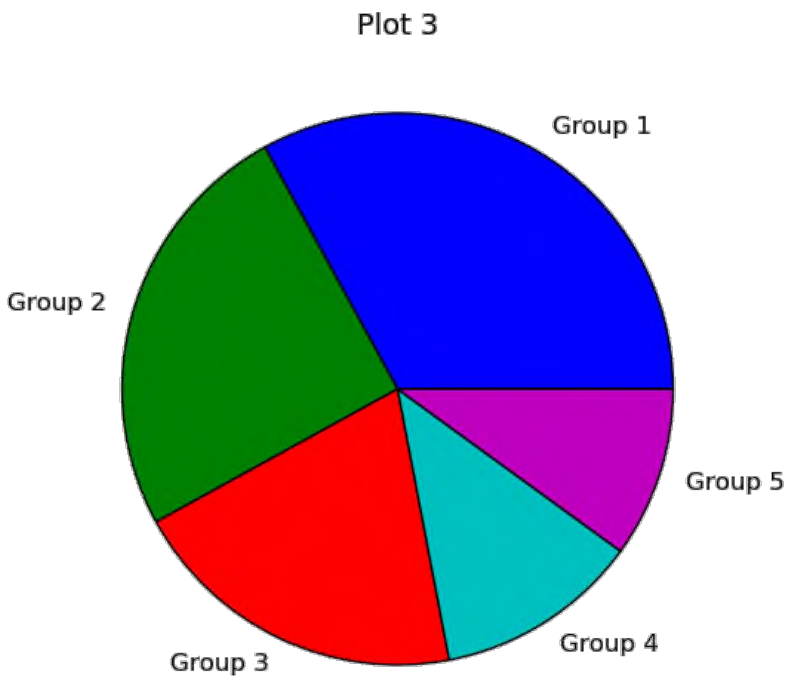
Каждую последовательность можно отобразить своим типом точек:

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

# равномерно распределённые значения от 0 до 5, с шагом 0.2
t = np.arange(0., 5., 0.2)

# красные чёрточки, синие квадраты и зелёные треугольники
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



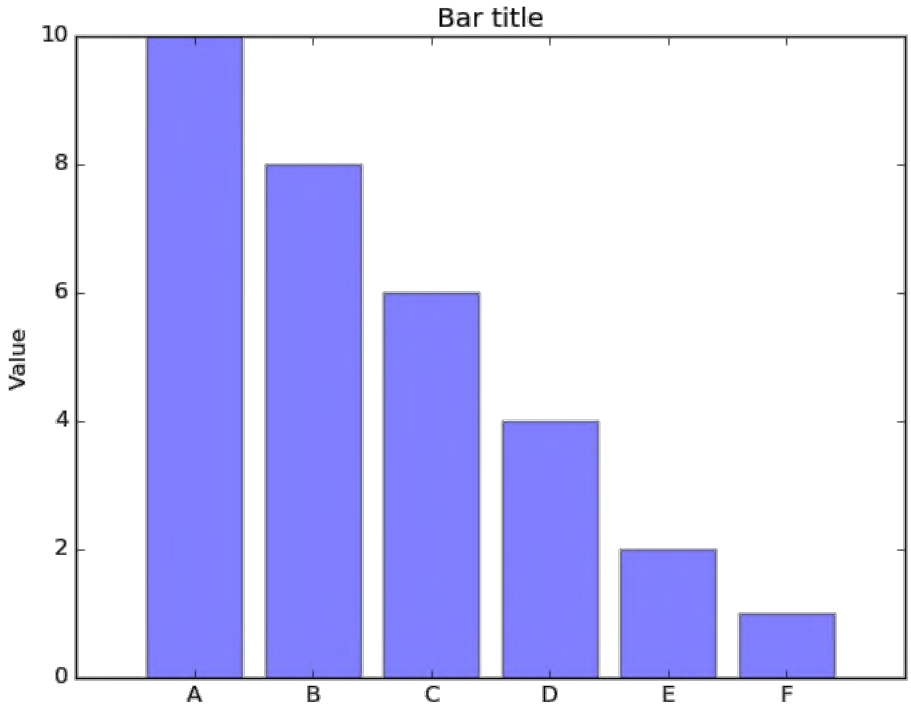
Также в matplotlib существует возможность строить круговые диаграммы:

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

data = [33, 25, 20, 12, 10]
plt.figure(num=1, figsize=(6, 6))
plt.axes(aspect=1)
plt.title('Plot 3', size=14)
plt.pie(data, labels=('Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5'))
plt.show()
```



И аналогичным образом столбчатые диаграммы:

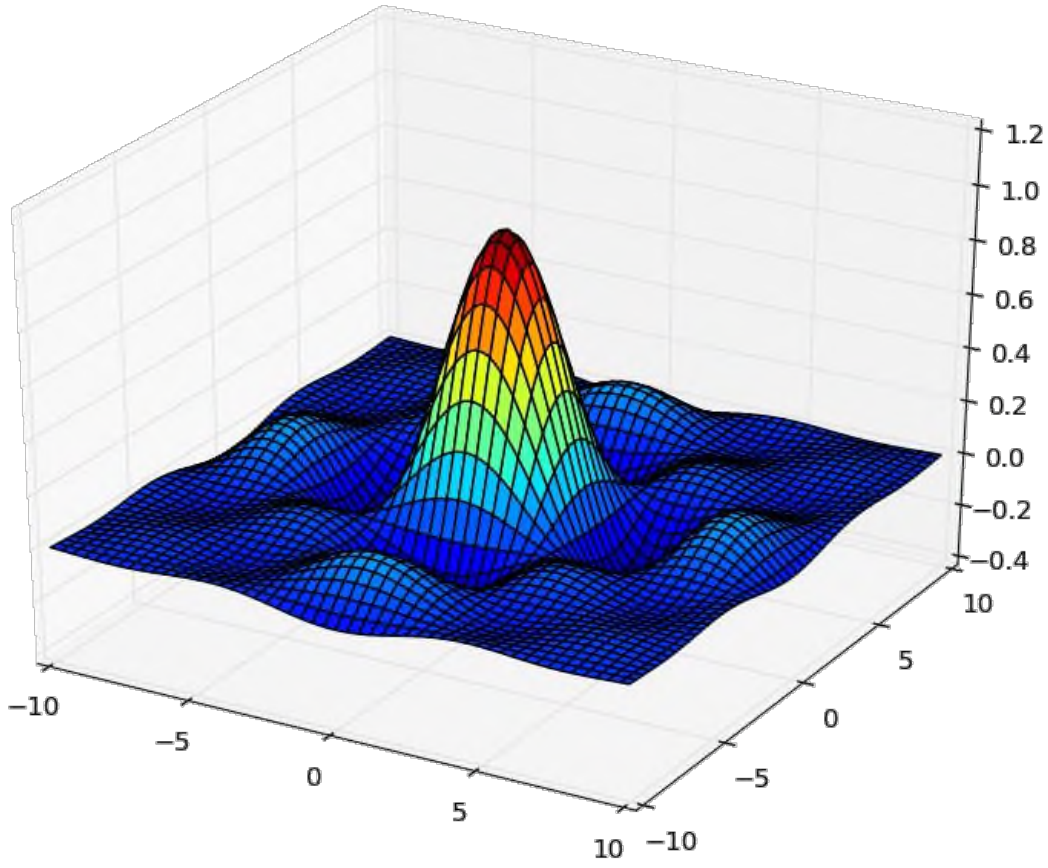
```
import numpy as np
import matplotlib.pyplot as plt

objects = ('A', 'B', 'C', 'D', 'E', 'F')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Value')
plt.title('Bar title')

plt.show()
```



Цветовые карты используются, если нужно указать в какие цвета должны окрашиваться участки трёхмерной поверхности в зависимости от значения Z в этой области. Цветовую карту можно задать самому, а можно воспользоваться готовой. Рассмотрим использование цветовой карты на примере графика функции z(x,y)=sin(x)*sin(y)/(x*y).

```
import pylab
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import numpy

def makeData():
    x = numpy.arange(-10, 10, 0.1)
    y = numpy.arange(-10, 10, 0.1)
    xgrid, ygrid = numpy.meshgrid(x, y)
    zgrid = numpy.sin(xgrid)*numpy.sin(ygrid)/(xgrid*ygrid)
    return xgrid, ygrid, zgrid
```

```
x, y, z = makeData()


fig = pylab.figure()
axes = Axes3D(fig)
axes.plot_surface(x, y, z, rstride=4, cstride=4, cmap=cm.jet)
pylab.show()
```



Альтернативой к использованию mpl_toolkits.mplot3d является библиотека plotly, которая позволяет интерактивно взаимодействовать с графиком, поворачивая его или увеличивая некоторую область в пространстве.

Функция eval()

В Python есть встроенная функция eval(), которая выполняет строку с кодом и возвращает результат выполнения:

```
>>> eval("2 + 3*len('hello')")
17
```

```
>>>
```

Это очень мощная, но и очень опасная инструкция, особенно если строки, которые вы передаёте в eval, получены не из доверенного источника. Если строкой, которую мы решим скормить eval(), окажется "os.system('rm -rf /')", то интерпретатор честно запустит процесс удаления всех данных с компьютера.

Упражнение №2

Постройте график функции

y(x) = x*x - x - 6

и по графику найдите найдите корни уравнения y(x) = 0. (Не нужно применять численных методов — просто приблизьте график к корням функции настолько, чтобы было удобно их найти.)

Упражнение №3

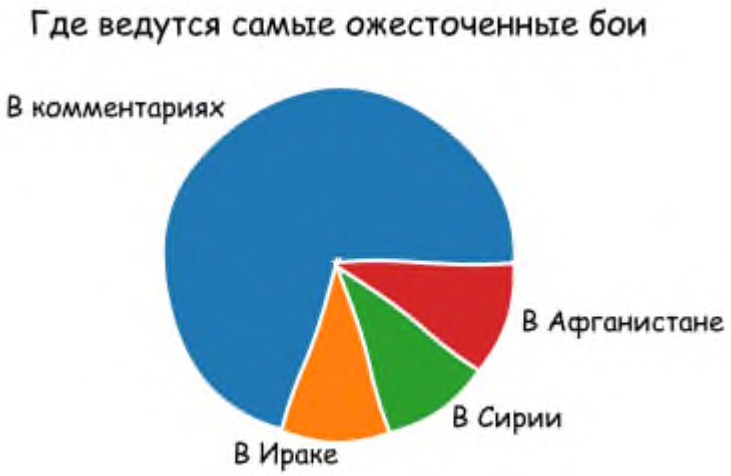
Постройте график функции

$$\log_{1+\tan\left(\frac{1}{1+\sin^2(x)}\right)}\left(x^2+1\right)\exp\left(-\frac{|x|}{10}\right)$$

Упражнение №4

Используя функцию eval() постройте график функции, введённой с клавиатуры. Чтобы считать данные с клавиатуры, используйте функцию input(). Попробуйте включить эффект «рисование от руки» посредством вызова plt.xkcd(). Посольку эта функция применяет некоторый набор настроек, избавиться от которых впоследствие не так просто, удобнее использовать ее как "контекстный менеджер" - это позволяет применить настройки временно, только к определенному блоку кода. Для этого используется ключевое слово with:

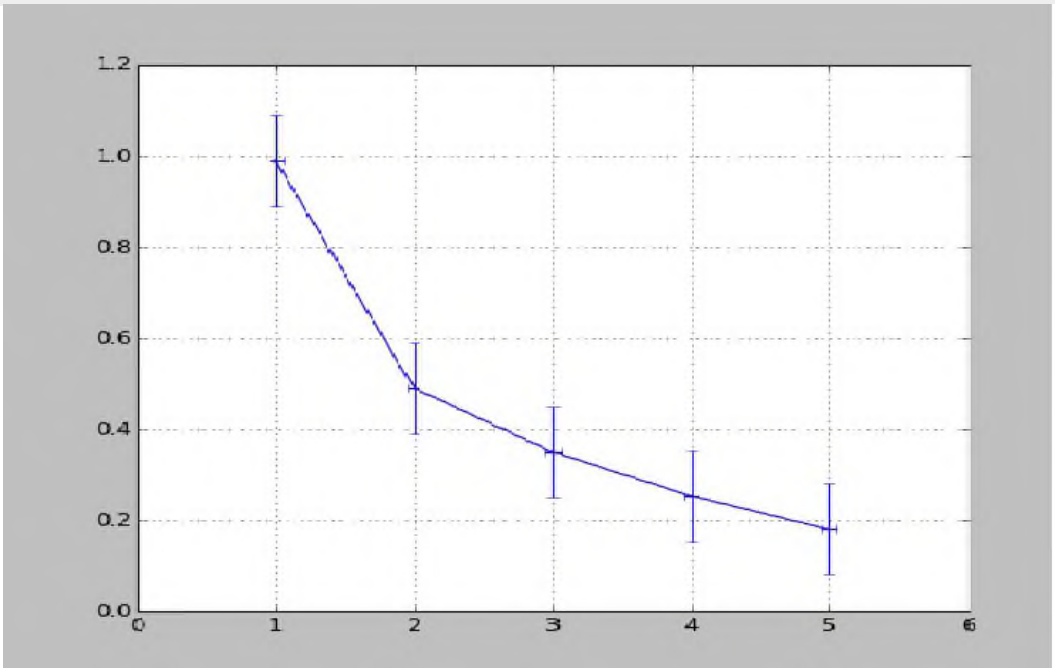
```
with plt.xkcd():
    plt.pie([70, 10, 10, 10], labels=('В комментариях', 'В Ираке', 'В Сирии', 'В Афганистане'))
    plt.title('Где ведутся самые ожесточенные бои')
```



Отображение погрешностей

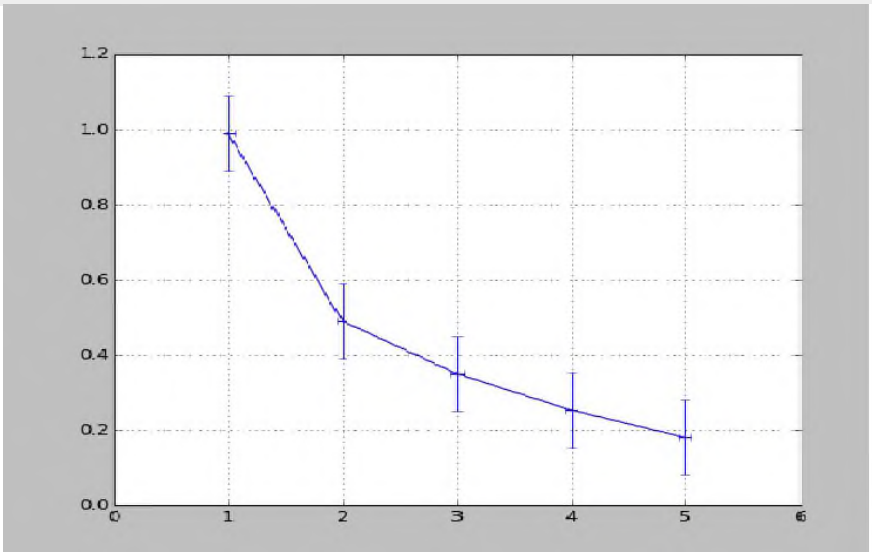
С помощью метода plt.errorbar можно рисовать точки с погрешностями измерений, как для лабораторных работ. Погрешности по осям абсцисс и ординат задаются в параметрах (соответственно) xerr и yerr.

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [0.99, 0.49, 0.35, 0.253, 0.18]
plt.errorbar(x, y, xerr=0.05, yerr=0.1)
plt.grid()
plt.show()
```



Альтернативой для plt.errorbar может слудить plt.fill_between, который заполняет область графика между кривыми, чтобы регулировать прозрачность используется аргумент alpha. Это число из отрезка [0, 1], на которое домножоается интенсивность цвета заполнения между кривыми.

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

x = np.arange(0, 10, 0.01)
plt.plot(x, x**2, label=r'$f = x^2$')
plt.scatter(x, x**2 + np.random.randn(len(x))*x, s=0.3)
plt.fill_between(x, 1.3*x**2, 0.7*x**2, alpha=0.3)
plt.legend(loc='best')
plt.savefig('figure_fill_between.png')
plt.show()
```



В уже использованном модуле numpy есть метод **polyfit**, позволяющий приближать данные методом наименьших квадратов. Он возвращает погрешности и коэффициенты полученного многочлена.

```
x = [1, 2, 3, 4, 5, 6]
y = [1, 1.42, 1.76, 2, 2.24, 2.5]
p, v = np.polyfit(x, y, deg=1, cov=True)
```

```
>>> p
array([0.28517032, 0.80720757])
>>> v
array([[0.00063242, -0.00221348],
[-0.00221348, 0.00959173]])
```

Многочлен задается формулой p(x) = p[0] * x**deg + ... + p[deg]

Для того, чтобы не выписывать каждый раз руками эту формулу для разных степеней, есть функция poly1d, которая возвращает функцию полинома, описанного точками p. Возвращенная функция может принимать на вход не только число, но и список значений, в таком случае, будет вычислено значение функции в каждой точке списка и возвращен список результатов.

```
p_f = np.poly1d(p)
p_f(0.5)
p_f([1, 2, 3])
```

Упражнение №5

Приблизить данные из приведённого примера с погрешностями или свои собственные (из лабораторного практикума по общей физике) многочленами первой и второй степени. Начертить точки с погрешностями и полученные аппроксимационные кривые на одном графике.

Упражнение №6 *

Постройте график функции **Вейерштрасса**

Практическая работа № 4

Основной библиотекой, которую рекомендуется использовать в данной работе, является библиотека scikit-learn (http://scikit-learn.org). Библиотека scikit-learn предоставляет реализацию целого ряда алгоритмов для обучения с учителем (Supervised Learning) и обучения без учителя (Unsupervised Learning) через интерфейс для языка программирования Python.

В данной работе потребуются следующие модули:

confusion_matrix - http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

classification_report – http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

accuracy_score - http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

metrics - http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn.metrics.roc_auc_score

train_test_split - http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

LogisticRegression - http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

KNeighborsClassifier - http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

RandomForestClassifier - http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

GaussianNB - http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

А также, в зависимости от варианта задания – один из модулей встроенных генераторов датасетов –

make_blobs - http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

make_moons - http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

make_classification - http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html#sklearn.datasets.make_classification

Кроме того, в работе будет использоваться библиотека *NumPy* *http://www.numpy.org*, позволяющая работать с многомерными массивами и высокоуровневыми математическими функциями.

Программный код для импорта указанных модулей может выглядеть следующим образом:

```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

Для построения графиков рекомендуется использовать библиотеку `matplotlib` и ее модуль `pyplot` https://matplotlib.org/users/pyplot_tutorial

А для отображения на графике области принятия решения - готовую функцию `plot_2d_separator`, которой нужно передать на вход объект `classifier` – модель классификатора и `X` – массив входных данных:

```
import matplotlib.pyplot as plt
def plot_2d_separator(classifier, X, fill=False, line=True,
ax=None, eps=None):
    if eps is None:
        eps = 1.0 #X.std() / 2.
    x_min, x_max = X[:, 0].min() - eps, X[:, 0].max() + eps
    y_min, y_max = X[:, 1].min() - eps, X[:, 1].max() + eps
    xx = np.linspace(x_min, x_max, 100)
    yy = np.linspace(y_min, y_max, 100)
    X1, X2 = np.meshgrid(xx, yy)
    X_grid = np.c_[X1.ravel(), X2.ravel()]
    try:
        decision_values = classifier.decision_function(X_grid)
        levels = [0]
        fill_levels = [decision_values.min(), 0,
```

```
decision_values.max()]
    except AttributeError:
        # no decision_function
        decision_values = classifier.predict_proba(X_grid)[:, 1]
        levels = [.5]
        fill_levels = [0, .5, 1]
    if ax is None:
        ax = plt.gca()
    if fill:
        ax.contourf(X1, X2, decision_values.reshape(X1.shape),
        levels=fill_levels, colors=['cyan', 'pink', 'yellow'])
    if line:
        ax.contour(X1, X2, decision_values.reshape(X1.shape),
levels=levels, colors="black")
    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)
    ax.set_xticks(())
    ax.set_yticks(())
```

Генерация выборки

Сгенерируем данные, с которыми будем работать. В нашем случае это будут 2 «пузыря» (blob). Передадим в качестве параметра `centers = 2` – количество классов-пузырей, `random_state = 66` – основа, используемая для генерации случайных чисел, `cluster_std = 4` – стандартное отклонение кластеров, `shuffle = 1` – перемешиваем объекты внутри выборки.

В массив с именем X сохраним координаты каждого объекта выборки, а в массив y – метки классов.

```
X, y = make_blobs(centers = 2 , random_state = 66, cluster_std
= 4, shuffle = 1)
```

Посмотрим, что из себя представляют массивы X и y. Выведем первые 15 элементов каждого из массивов.

```
print ("Координаты точек: ")
print (X[:15])
print ("Метки класса: ")
print (y[:15])
```

```
Координаты точек:
[[-14.80437794  -7.18377798]
 [ -2.60632729   0.20074702]
 [ -7.67410393  -0.92722787]
 [-18.73964269  -1.88968606]
 [ -3.7511755    3.11333437]
 [ -6.13559977  -8.39517379]
 [ -8.10457133   6.15227722]
 [  4.91341461  -2.95516942]
 [ -2.86156125  10.56078045]
 [  0.52303829   3.14548666]
 [  3.53563356   5.80649298]
 [ -6.05018557  -2.10920558]
 [  0.8102374    1.86943425]
 [ -8.0759995   -0.91582206]
 [ -9.37948058 -10.35367349]]
Метки класса:
[0 1 0 0 1 0 1 0 1 1 1 0 1 1 0]
```

Первый элемент выборки с координатами `[-14.80437794 -7.18377798]` относится к классу 0,

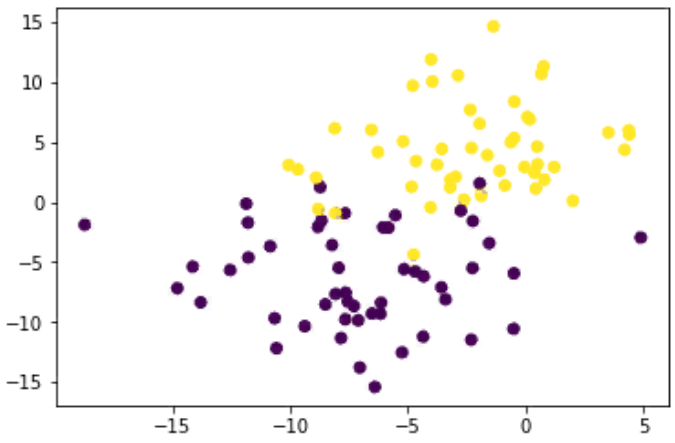
второй элемент `[ -2.60632729  0.20074702]` – к классу 1 и т.д.

Отобразим на графике сгенерированные данные.

В качестве координат точек передадим первый и второй столбец массива X, для указания цвета точки (параметр `c`) используем метку класса из массива y

```
plt.scatter (X[:,0], X[:,1], c=y)
plt.show
```

```
Out[5]: <function matplotlib.pyplot.show(*args, **kw)>
```



Видно, что объекты двух классов пересекаются между собой.

Разобьем выборку на обучающее и тестовое множества, используя функцию `train_test_split`. В качестве аргументов передаем массив X, массив y, test_size = 0.25 – означает, что на тестовую часть пойдет 25% всей выборки, соответственно, на обучающую – 75%, а также указываем, что разбиение будет случайным, но воспроизводимым (`random_state = 1`). Если параметр `random_state = None`, то разбиение будет невоспроизводимым.

Функция `train_test_split` записывает результаты разбиения в 4 переменные. Назовем их `X_train, X_test, y_train, y_test`. В первую и вторую переменную будут записаны координаты объектов из обучающей и тестовой выборки соответственно, а в третью и четвертую – метки классов объектов из обучающей и экзаменационной выборки соответственно:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 1, )
```

Таким образом, в переменной X_train лежат координаты, а в y_train – метки классов соответствующих объектов из тестовой выборки. Эти переменные будут использоваться в дальнейшем для обучения модели.

X_test, y_test – соответственно координаты и метки классов объектов тестовой выборки. Эти переменные мы будем использовать для оценки точности модели.

Обучение модели и классификация

Для обучения модели и последующей классификации с использованием модулей библиотеки scikit-learn используется довольно стандартная процедура. Разберем на примере обучение и классификацию данных методом к-ближайших соседей:

1) Импортировать требуемый модуль, если этого не было сделано ранее

```
from sklearn.neighbors import KNeighborsClassifier
```

2) Создать переменную - модель классификатора, указав при необходимости параметры классификации. В нашем случае мы задаем два параметра – количество ближайших соседей = 1 и евклидову метрику.

```
knn = KNeighborsClassifier(n_neighbors=1, metric = 'euclidean')
```

Для большинства классификаторов, если не задавать никаких параметров, они будут выбраны по умолчанию. Список доступных

параметров можно посмотреть в документации, в нашем случае - http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Посмотреть доступные метрики расстояний также можно в документации на DistanceMetric: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

3) Обучить модель, используя метод `fit()`, передав в него координаты объектов и метки классов обучающей выборки

```
knn.fit(X_train, y_train)
```

4) Оценить качество модели используя метод `predict()` и тестовую выборку.

```
prediction = knn.predict(X_test)
```

Стоит отметить, что в метод `predict()` подаются только координаты объектов (`X_test`) без истинных меток класса (`y_test`). В общем случае, когда модель полностью настроена, в данный метод могут передаваться «боевые» данные – объекты, которые нужно проклассифицировать.

В нашем случае, в переменную `prediction` метод вернул предсказанные метки классов для каждого объекта из переменной `X_test`.

Зная истинные метки классов (переменная `y_test`) мы можем оценить, насколько точно работает наша модель.

Самое простое – вывести на экран истинные и предсказанные ответы:

```
print ('Prediction and test: ')
print (prediction)
print (y_test)
```

Кроме того, можно оценить матрицу неточностей (confusion matrix) используя функцию `confusion_matrix`, и передав в нее истинные и предсказанные ответы:

```
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
```

Для оценки аккуратности классификации можно использовать функцию `accuracy_score`:

```
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

Для оценки показателей полноты-точности и f1-меры воспользуемся функцией `classification_report`:

```
print(classification_report(y_test, prediction))
```

Оценить показатель AUC ROC можно следующим образом:

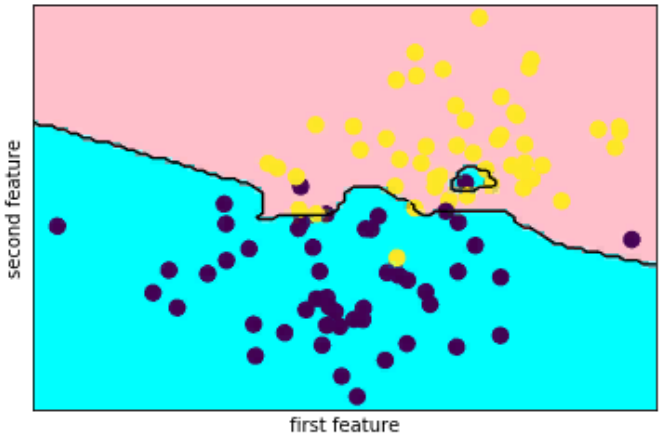
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, prediction)
```

Кроме того, воспользовавшись функцией `plot_2d_separator`, описанной выше, можно наглядно отобразить на графике область принятия решений по каждому классу:

```
plt.xlabel("first feature")
plt.ylabel("second feature")
plot_2d_separator(knn, X, fill=True)
plt.scatter(X[:, 0], X[:, 1], c=y, s=70)
```

```
Prediction and test:
[1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 1 0 1 0]
[1 0 0 1 1 0 0 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 0 0]
Confusion matrix:
[[10  3]
 [ 2 10]]
Accuracy score:  0.8
             precision    recall  f1-score   support

          0       0.83      0.77      0.80        13
          1       0.77      0.83      0.80        12

avg / total       0.80      0.80      0.80        25
```

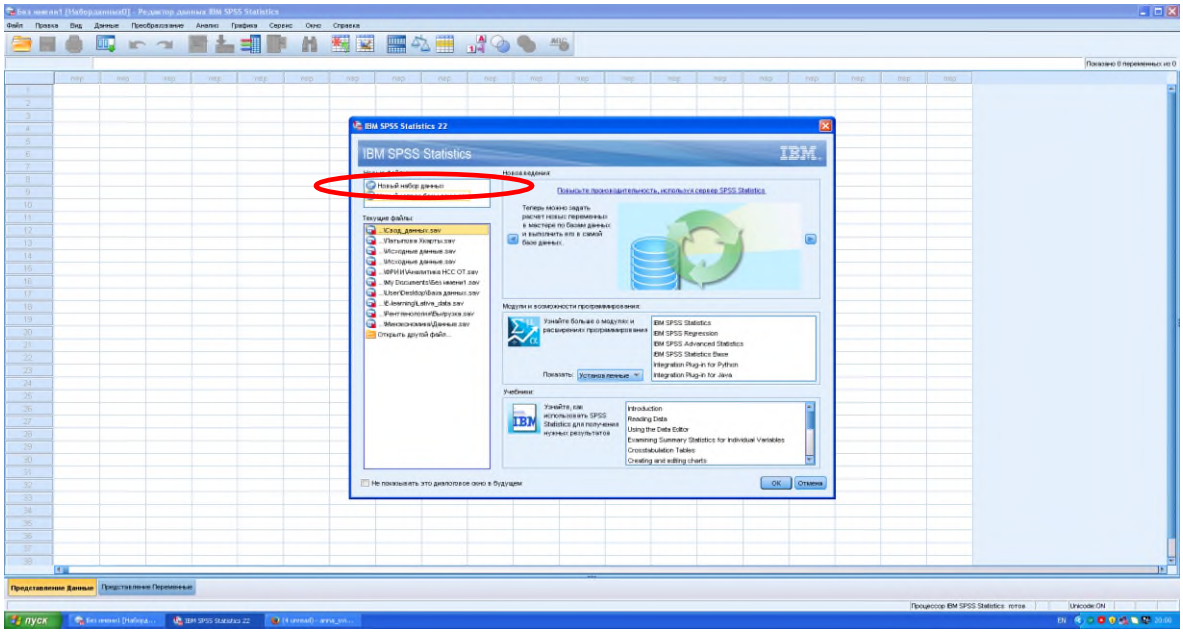
Out[28]: <matplotlib.collections.PathCollection at 0xcfb5690>



Практическая работа № 5

Обработка и анализ статистических данных с использованием программы SPSS Statistics

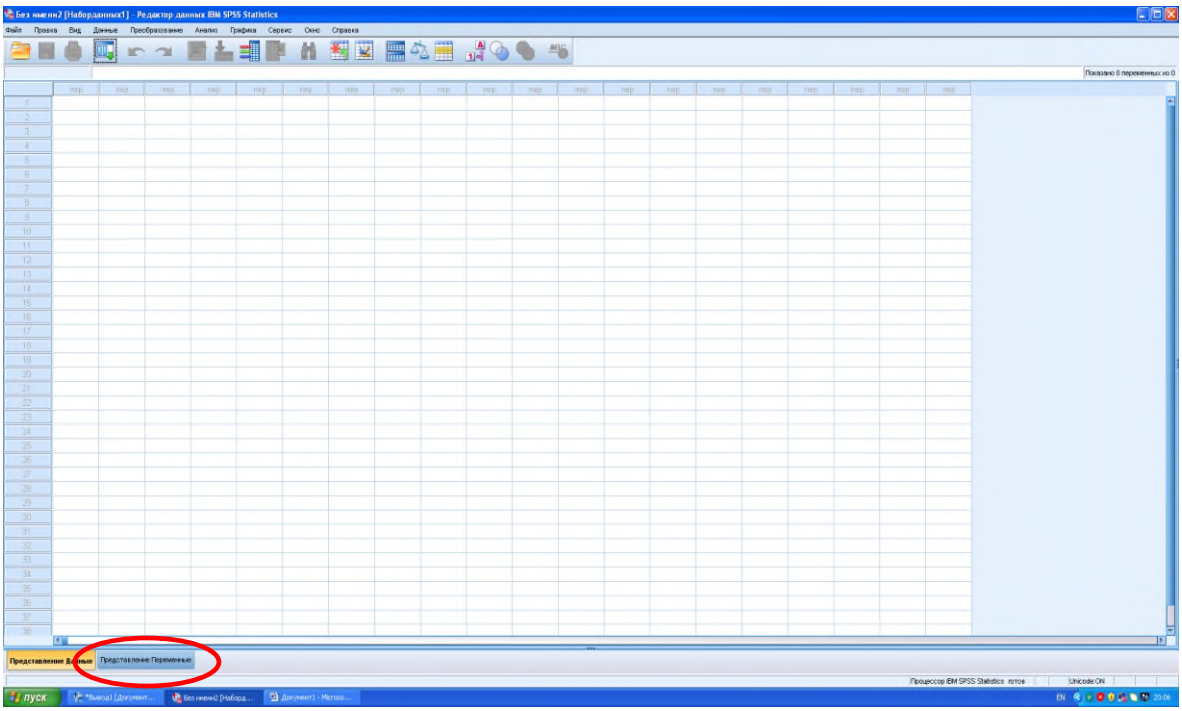
Запустите программу SPSS Statistics. После запуска откроется следующее окно:



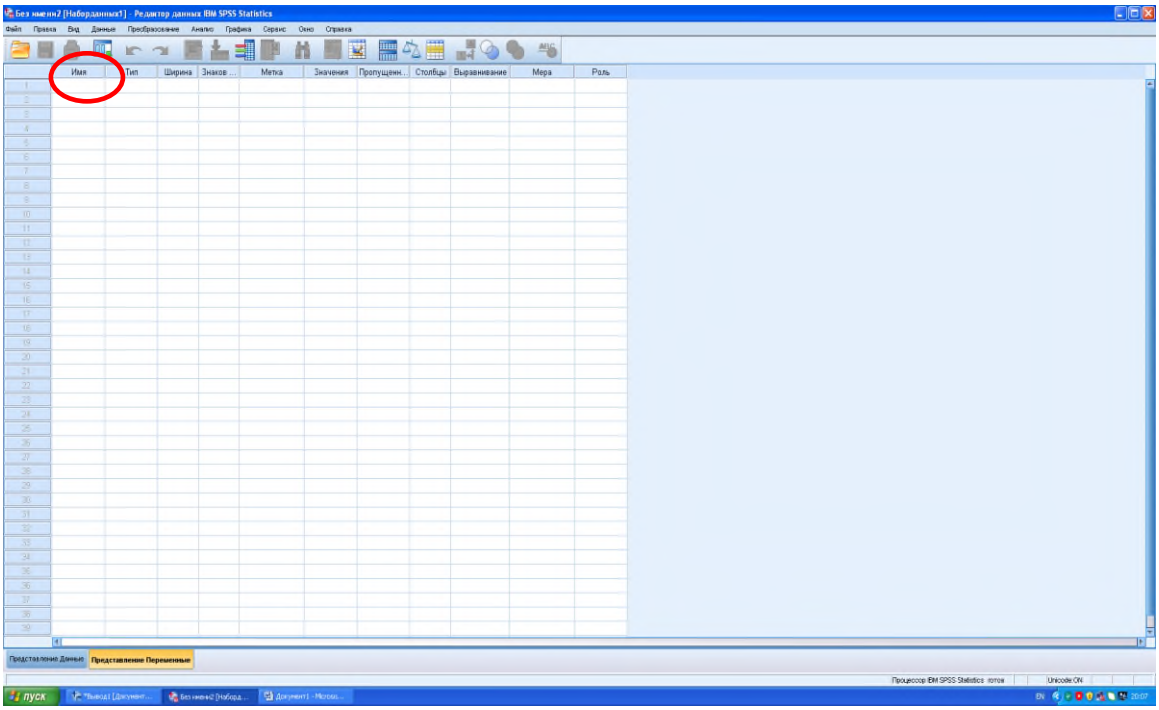
В данном окне необходимо выбрать «Новый набор данных» (см. рис.).

Откроется два окна: файл «Без имени [набор данных]» и «Вывод1». Начало работы происходит в окне «Без имени [набор данных]».

В данном окне необходимо перейти на вкладку «Представление переменные» внизу (см. рис.):



Откроется следующее окно:



Во вновь открывшемся окне необходимо ввести названия переменных в столбце «Имя» (см. рис.).

Например, следующим образом. Допустим, есть опросник, в котором первые 4 вопроса выглядят так:

1. Ваш пол

    1) мужской

2) женский

2. Ваш возраст

1) до 25 лет

2) 25-35 лет

3) 36-45 лет

4) 46-55 лет

5) более 55 лет

3) Сколько раз Вы брали кредит в банке?

1) никогда не брал

2) 1-2 раза

3) 3-5 раз

4) 6-10 раз

5) более 10 раз

6) точно не помню

4) Сколько раз Вы брали кредит в микрофинансовой организации?

1) никогда не брал

2) 1-2 раза

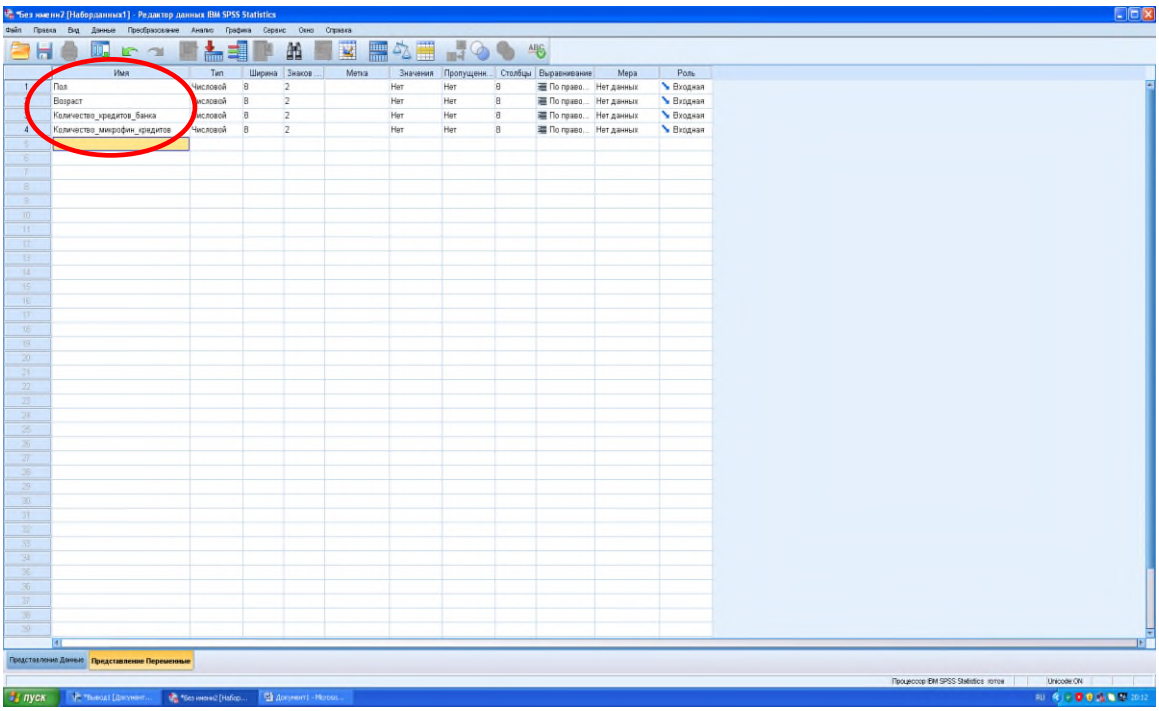
3) 3-5 раз

4) 6-10 раз

5) более 10 раз

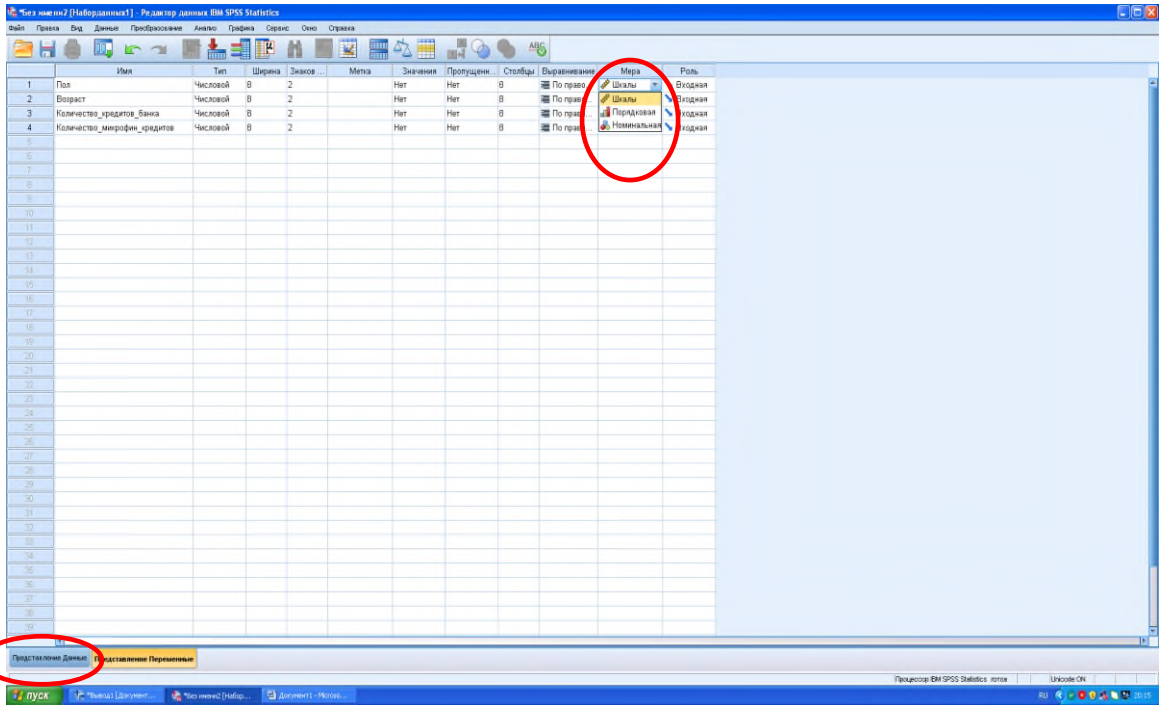
6) точно не помню

В этом случае названия переменных могут выглядеть так:



Программа не даст поставить пробел между словами в имени, поэтому ставьте либо точку, либо нижнее подчеркивание.

По умолчанию программа выставить тип переменной – числовая, и ширину. Оставьте эти настройки по умолчанию. Корректировке подлежит только столбце «Мера», в котором необходимо установить значение «шкала», нажав левой кнопкой на ячейку (см. рис.):

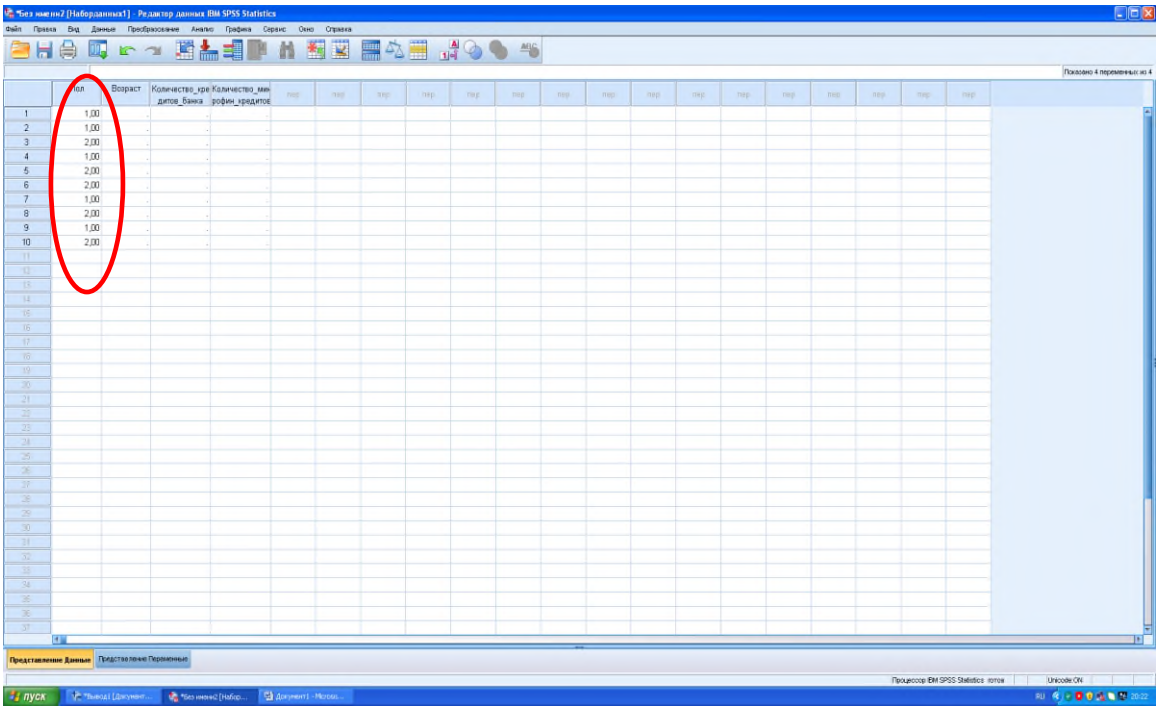


После изменения столбца «Мера» вернитесь на вкладку «Представление Данные» (внизу окна).

На этой вкладке необходимо ввести результаты опроса – ответы респондентов. Например, Вы опросили 10 человек, и получили следующие ответы на первый вопрос «Ваш пол»:

Номер респондента (номер анкеты по порядку)	Ответ респондента	**Номер ответа на вопрос «Ваш пол» в соответствии с анкетой**
1	мужской	1
2	мужской	1
3	женский	2
4	мужской	1
5	женский	2
6	женский	2
7	мужской	1
8	женский	2
9	мужской	1
10	женский	2

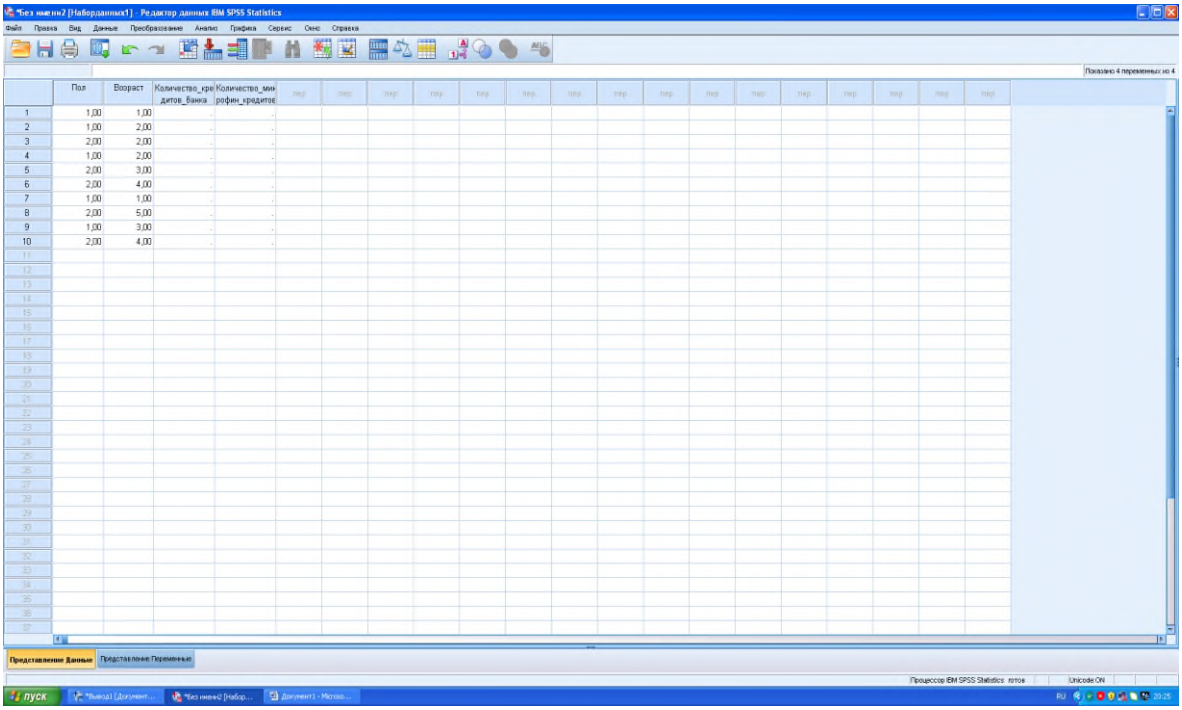
Полученные номера ответов необходимо внести на вкладку «Представление Данные» в столбик «Пол» в той же последовательности, что и в таблице:



Аналогичным образом обрабатывается следующий вопрос (ваш возраст):

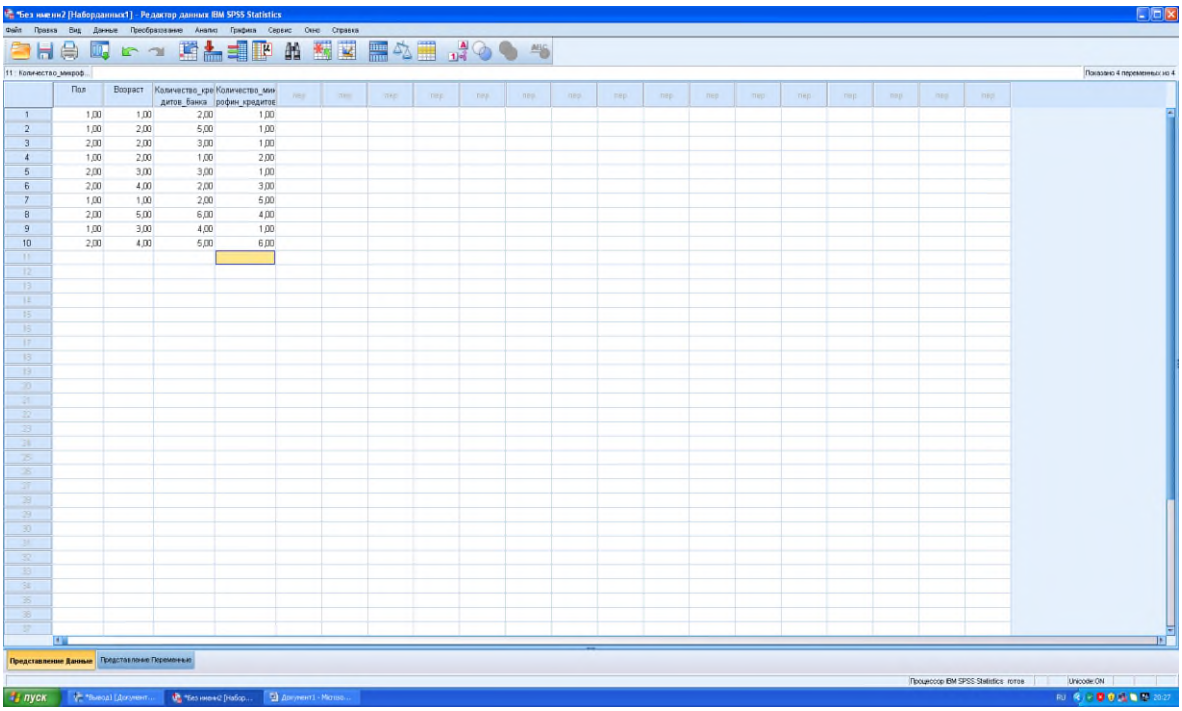
Номер респондента (номер анкеты по порядку)	Ответ респондента	**Номер ответа на вопрос «Ваш возраст» в соответствии с анкетой**
1	До 25 лет	1
2	25-35 лет	2
3	25-35 лет	2
4	25-35 лет	2
5	36-45 лет	3
6	46-55 лет	4
7	До 25 лет	1
8	Более 55 лет	5
9	36-45 лет	3
10	46-55 лет	4

После переноса данных таблица «Представление переменных» будет выглядеть так:



Аналогичным образом обрабатываются все ответы на анкету. В ТАБЛИЦЕ ДОЛЖНЫ БЫТЬ ТОЛЬКО ЧИСЛОВЫЕ ДАННЫЕ!

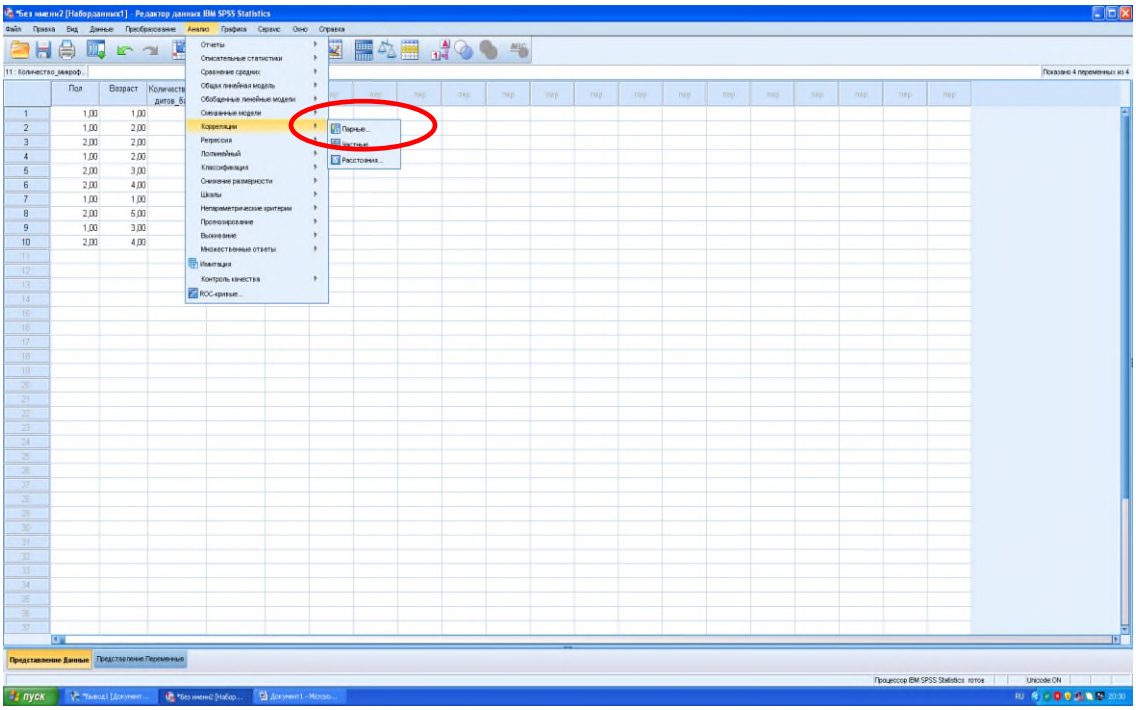
Например, после введения данных по всем 4 вопросам вкладка «Представление данных» будет иметь следующий вид:



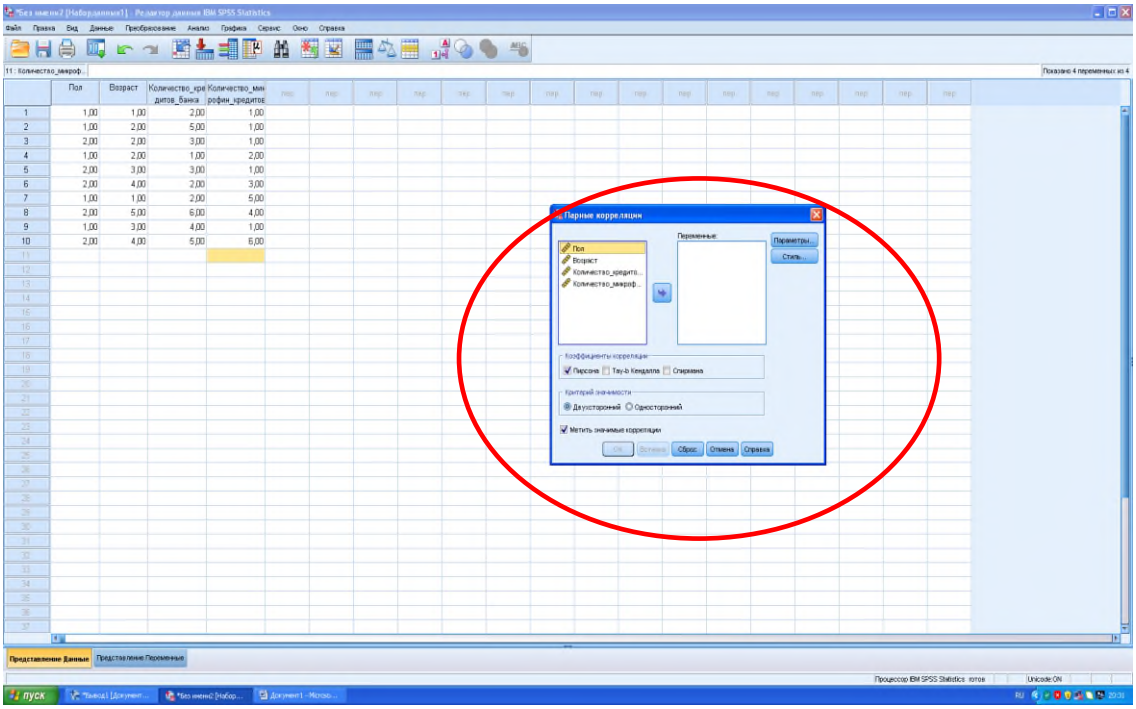
В ваших анкетах по 10 вопросов, соответственно, должно получиться 10 заполненных столбцов.

После этого переходим к анализу. В первую очередь выполняется корреляционный анализ (для этого в верхней строке панели инструментов необходимо нажать «Анализ» и выбрать корреляционный анализ). Внешний вид выбора показан на рис.

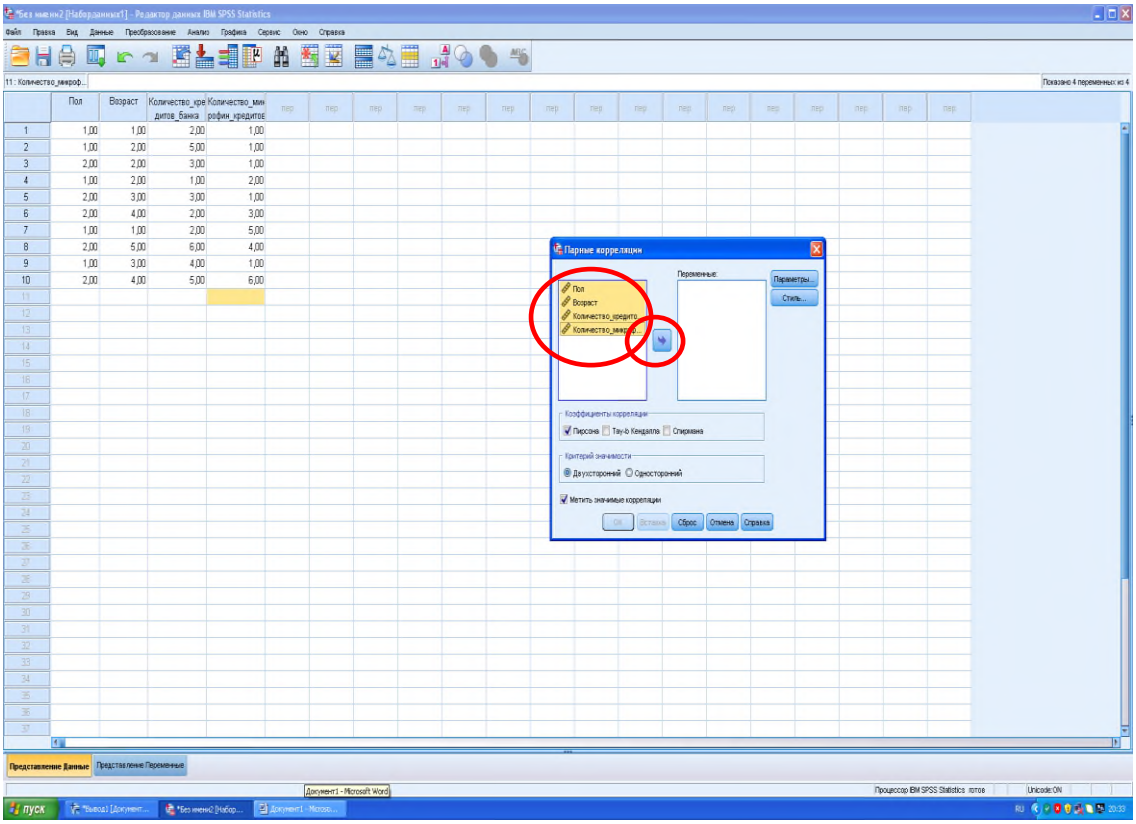
Необходимо выбрать парный тип корреляции.



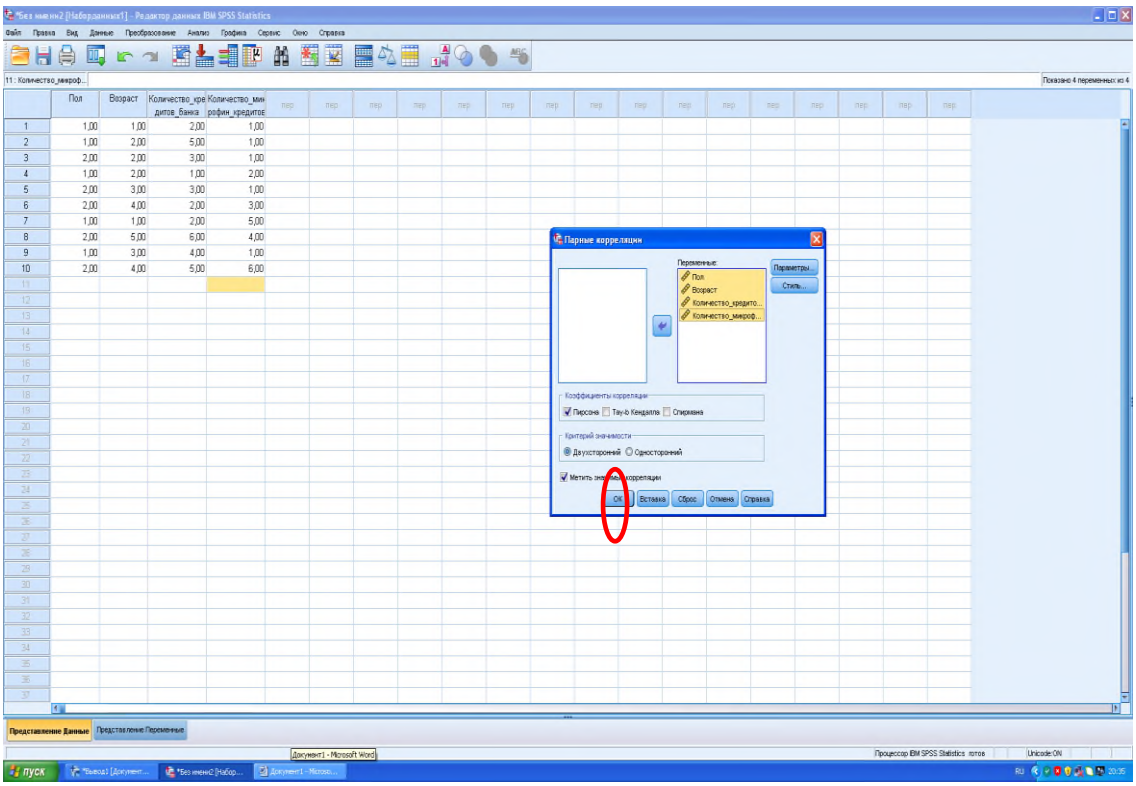
После нажатия на «Парные» левой кнопкой мыши откроется следующее окно:



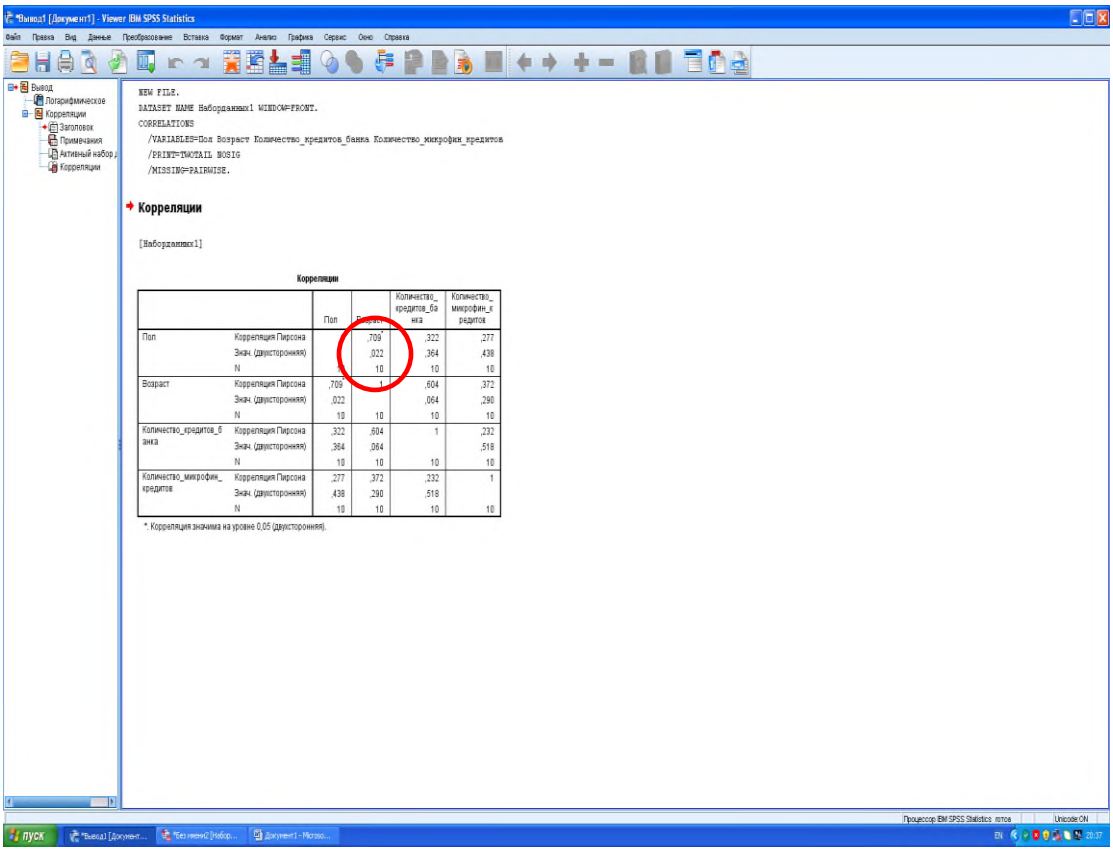
В выделенном окне с помощью клавиши «Shift» выбираем все переменные в левом окне и переносим их с помощью стрелки в правое окно (левое окно и стрелка показаны на рисунке):



После переноса экран будет выглядеть следующим образом:

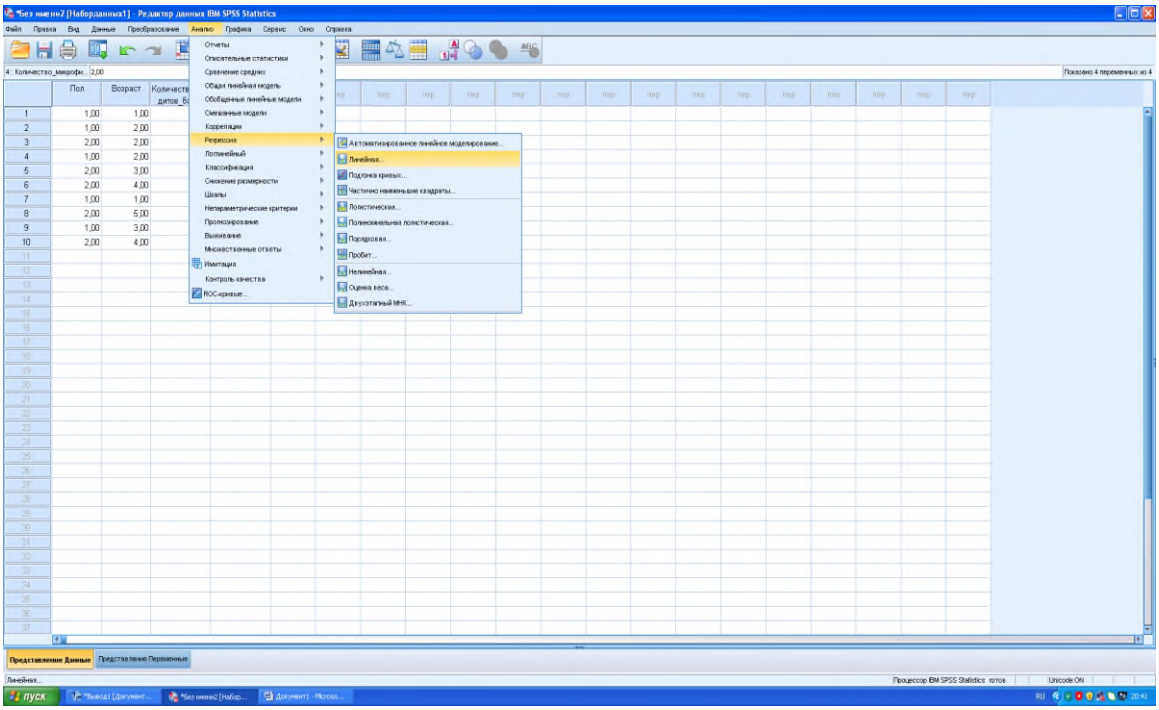


После переноса для запуска анализа необходимо нажать кнопку «ОК». Результаты анализа появятся в окне «Вывод1» и будут выглядеть следующим образом:

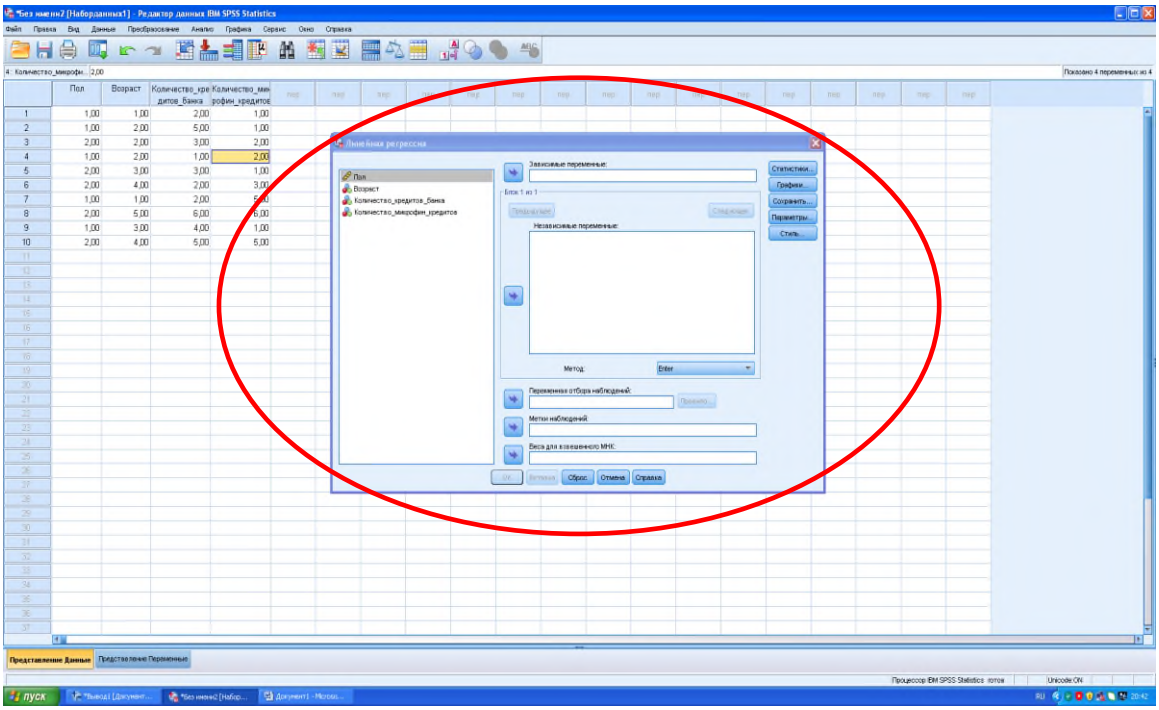


Статистически значимые корреляции отмечены звездочками. В рассмотренном примере наблюдается статистически значимая корреляция между полом и возрастом. Таблицу корреляций необходимо скопировать (нажав на нее правой кнопкой мыши) для отчета по практике.

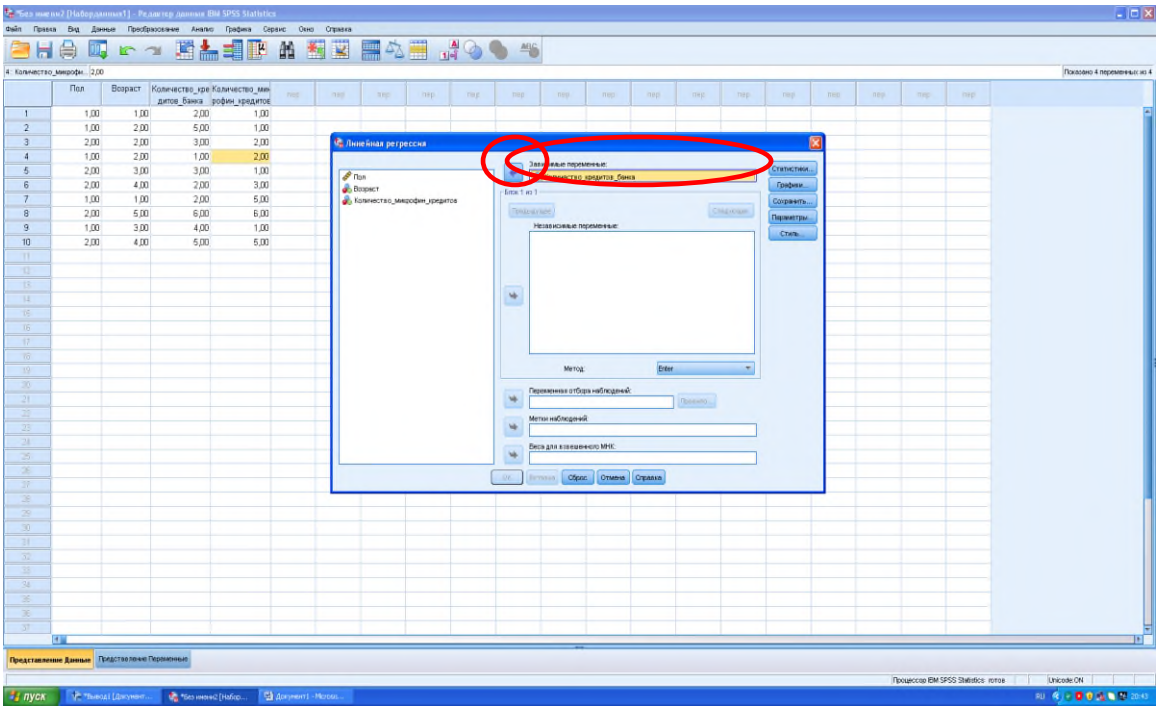
Далее осуществляется возврат на лист «Без имени [Набор данных]» для проведения регрессионного анализа:



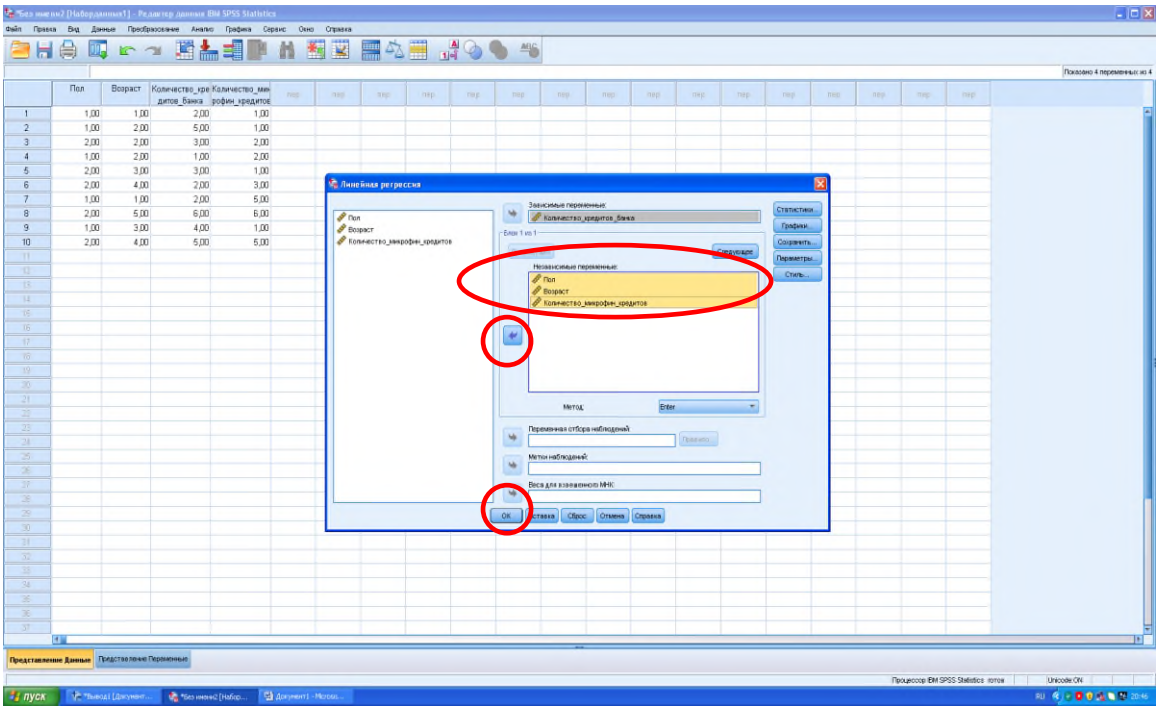
Выбрав линейный тип регрессии, попадаем в следующее окно:



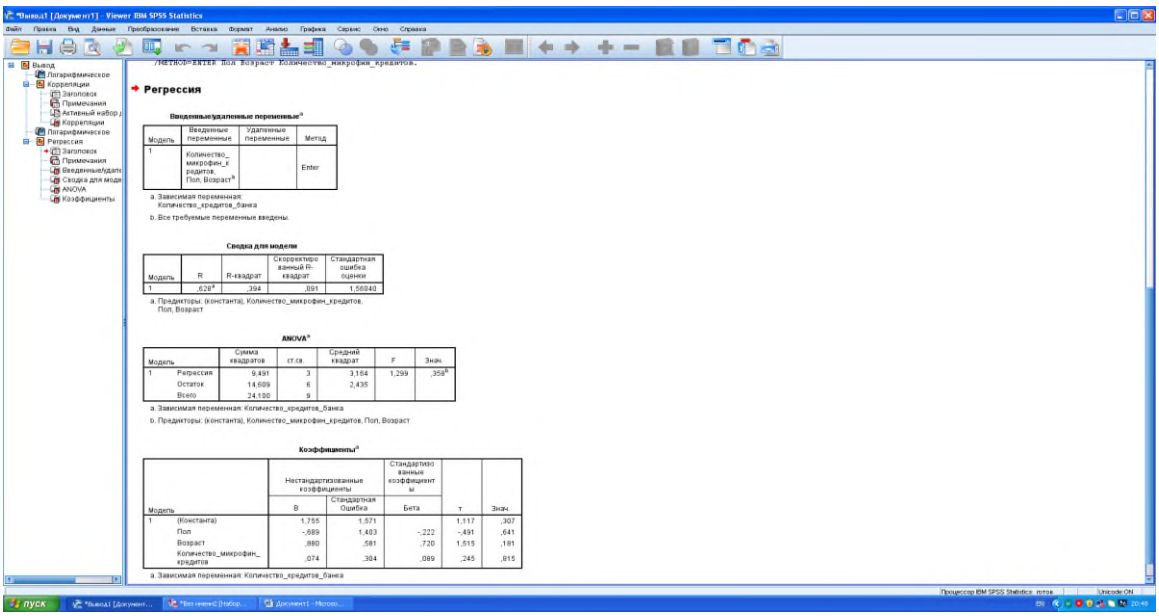
Переменную, которую хотите рассмотреть как зависимую (например, Количество_кредитов_банка в рассматриваемом примере) переносим стрелочкой в строку «Зависимые переменные»:



Остальные переменные при помощи стрелочки, расположенной ниже, переносим в окно «Независимые переменные»:



Проводим анализ нажатием кнопки «ОК». Результат появится в окне «Вывод1»:



Полученный результат необходимо скопировать для внесения в отчет к лабораторной работе (4 таблицы, показанные на рис.).

После проведения корреляционного и регрессионного анализа можно завершить работу с программой SPSS Statistics.

В отчете должны быть отражены:

1. Составленный Вами опрос по теме диссертации.

2. Таблица ответов респондентов.

3. Результаты корреляционного анализа (скопированные из программы SPSS Statistics).

4. Результаты регрессионного анализа (скопированные из программы SPSS Statistics).

5. Выводы по проведенному анализу.