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

data = pd.read_csv("diamonds.csv")

#1. Сколько в наборе данных объектов и признаков? Дать описание
каждому признаку, если оно есть.
data.shape # 53940 объектов, 10 признаков (11 колонка - просто номер)
data = data.drop(columns = 'Unnamed: 0')
data.head()
# carat: вес алмаза
# cut: качество огранки (Fair, Good, Very Good, Premium, Ideal)
# color: цвет алмаза(J I H G F E D)
# clarity: чистота алмаза (SI2, SI1, VS2, VS1, VVS2, VVS1, IF)
# depth: глубина алмаза
# table: ширина верхней части алмаза относительно его наибольшей
ширины
# price: цена алмаза
# x: длина алмаза
# y: ширина алмаза
# z: высота алмаза
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

```python
#2. Сколько категориальных признаков, какие?
# cut, color, clarity - категориальные

#3. Столбец с максимальным количеством уникальных значений
категориального признака?
# clarity - 7 значений

#4. Есть ли бинарные признаки?
# нет

#5. Какие числовые признаки?
# carat, depth, table, price, x, y, z

#6-8. Есть ли пропуски?
data.isna().sum() # пропусков нет
```

```
carat      0
cut        0
color      0
clarity    0
depth      0
```

```
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

```
#9. Есть ли на ваш взгляд выбросы, аномальные значения?
data.describe()
#выбросы могут наблюдаться в колонках carat, price, y, z, максимальные
значения сильно отличаются от среднего
```

```
               carat         depth         table         price
x  \
count  53940.000000  53940.000000  53940.000000  53940.000000
53940.000000
mean       0.797940     61.749405     57.457184   3932.799722
5.731157
std        0.474011      1.432621      2.234491   3989.439738
1.121761
min        0.200000     43.000000     43.000000    326.000000
0.000000
25%        0.400000     61.000000     56.000000    950.000000
4.710000
50%        0.700000     61.800000     57.000000   2401.000000
5.700000
75%        1.040000     62.500000     59.000000   5324.250000
6.540000
max        5.010000     79.000000     95.000000  18823.000000
10.740000

                   y             z
count  53940.000000  53940.000000
mean       5.734526      3.538734
std        1.142135      0.705699
min        0.000000      0.000000
25%        4.720000      2.910000
50%        5.710000      3.530000
75%        6.540000      4.040000
max       58.900000     31.800000
```

```
#закодируем категориальные признаки
cut_order = {'Fair': 0, 'Good': 1, 'Very Good': 2, 'Premium': 3,
'Ideal': 4}
color_order = {'J': 0, 'I': 1, 'H': 2, 'G': 3, 'F': 4, 'E': 5, 'D': 6}
clarity_order = {'I1': 0, 'SI2': 1, 'SI1': 2, 'VS2': 3, 'VS1': 4,
'VVS2': 5, 'VVS1': 6, 'IF': 7}
data['cut'] = data['cut'].map(cut_order)
data['color'] = data['color'].map(color_order)
data['clarity'] = data['clarity'].map(clarity_order)
```

```
data.head(140)
```

```
     carat  cut  color  clarity  depth  table  price     x     y     z
0     0.23    4      5        1   61.5   55.0    326  3.95  3.98  2.43
1     0.21    3      5        2   59.8   61.0    326  3.89  3.84  2.31
2     0.23    1      5        4   56.9   65.0    327  4.05  4.07  2.31
3     0.29    3      1        3   62.4   58.0    334  4.20  4.23  2.63
4     0.31    1      0        1   63.3   58.0    335  4.34  4.35  2.75
..     ...  ...    ...      ...    ...    ...    ...   ...   ...   ...
135   0.63    3      5        6   60.9   60.0   2765  5.52  5.55  3.37
136   0.71    2      4        4   60.1   62.0   2765  5.74  5.77  3.46
137   0.71    3      4        4   61.8   59.0   2765  5.69  5.73  3.53
138   0.76    4      2        2   61.2   57.0   2765  5.88  5.91  3.61
139   0.64    4      3        6   61.9   56.0   2766  5.53  5.56  3.43

[140 rows x 10 columns]
```

```
#нормализуем числовые
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
nums = ['carat', 'depth', 'table', 'price', 'x', 'y', 'z']
data[nums] = ss.fit_transform(data[nums])

data.head(150)
```

```
        carat  cut  color  clarity     depth     table     price
x  \
0   -1.198168    4      5        1 -0.174092 -1.099672 -0.904095 -
1.587837
1   -1.240361    3      5        2 -1.360738  1.585529 -0.904095 -
1.641325
2   -1.198168    1      5        4 -3.385019  3.375663 -0.903844 -
1.498691
3   -1.071587    3      1        3  0.454133  0.242928 -0.902090 -
1.364971
4   -1.029394    1      0        1  1.082358  0.242928 -0.901839 -
1.240167
..        ...  ...    ...      ...       ...       ...       ...
...
145 -0.206621    1      2        5  0.244725  2.928129 -0.292224 -
0.099093
146 -0.185524    2      3        4  1.082358  0.690462 -0.291973 -
0.188239
147 -0.143331    2      6        2 -1.081527 -0.652139 -0.291973
0.088115
148 -0.206621    2      6        2 -0.453303  0.242928 -0.291973 -
0.063434
149 -0.206621    4      5        2 -0.592908 -0.204605 -0.291973 -
0.001032
```

```
            y         z
0   -1.536196 -1.571129
1   -1.658774 -1.741175
2   -1.457395 -1.741175
3   -1.317305 -1.287720
4   -1.212238 -1.117674
..        ...       ...
145 -0.074008 -0.054888
146 -0.109030 -0.026547
147  0.118616 -0.026547
148 -0.003963 -0.083229
149  0.022304 -0.054888

[150 rows x 10 columns]
```

```
#10. Столбец с максимальным средним значением после нормировки
признаков через стандартное отклонение?
# вопрос не корректный, после нормализации через среднеквадратичное
отклонение все средние значения равны 0

#11. Столбец с целевым признаком?
# price
y = data['price']
data = data.drop(['price'], axis = 1)
data.head()
```

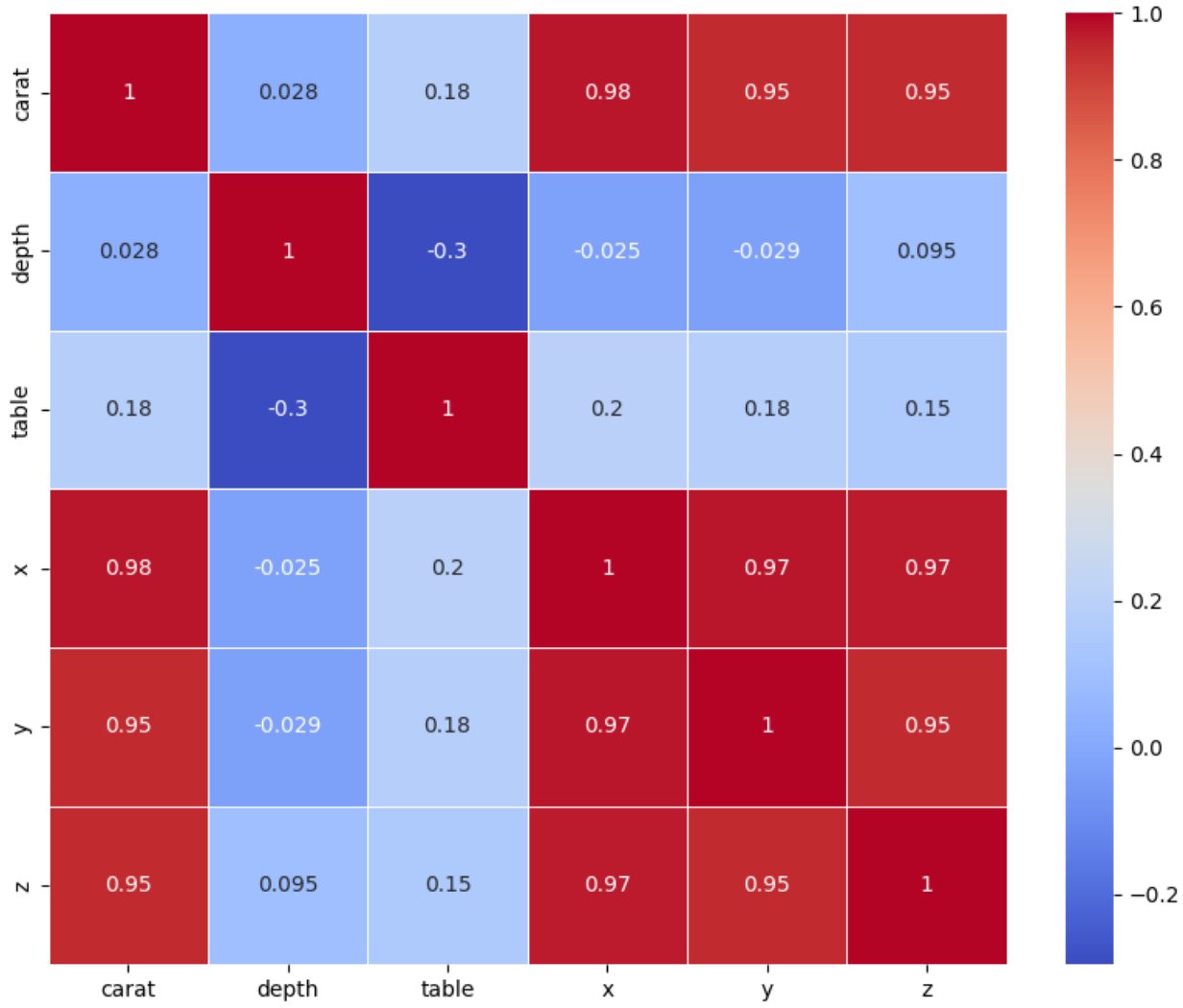
```
      carat  cut  color  clarity     depth     table         x
y  \
0 -1.198168    4      5        1 -0.174092 -1.099672 -1.587837 -
1.536196
1 -1.240361    3      5        2 -1.360738  1.585529 -1.641325 -
1.658774
2 -1.198168    1      5        4 -3.385019  3.375663 -1.498691 -
1.457395
3 -1.071587    3      1        3  0.454133  0.242928 -1.364971 -
1.317305
4 -1.029394    1      0        1  1.082358  0.242928 -1.240167 -
1.212238

          z
0 -1.571129
1 -1.741175
2 -1.741175
3 -1.287720
4 -1.117674
```

```
#12. Сколько объектов попадает в тренировочную выборку при
использовании train_test_split с параметрами test_size = 0.3,
random_state = 42?
```

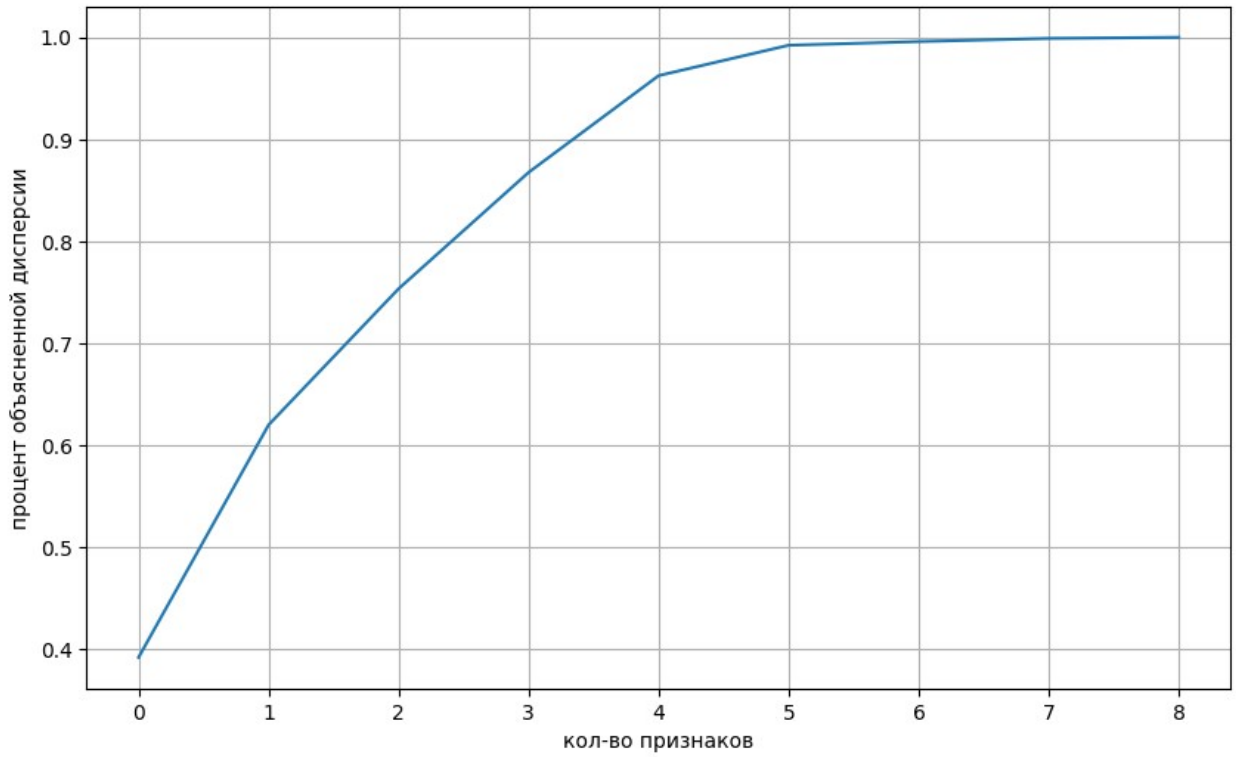
```
# всего 53940 обьектов, следовательно в тренировочной выборке будет 53940*0.7 = 37758 объектов

#13. Между какими признаками наблюдается линейная зависимость (корреляция)?
import seaborn as sns
nums = ['carat', 'depth', 'table', 'x', 'y', 'z']
corr = data[nums].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
plt.show()
```

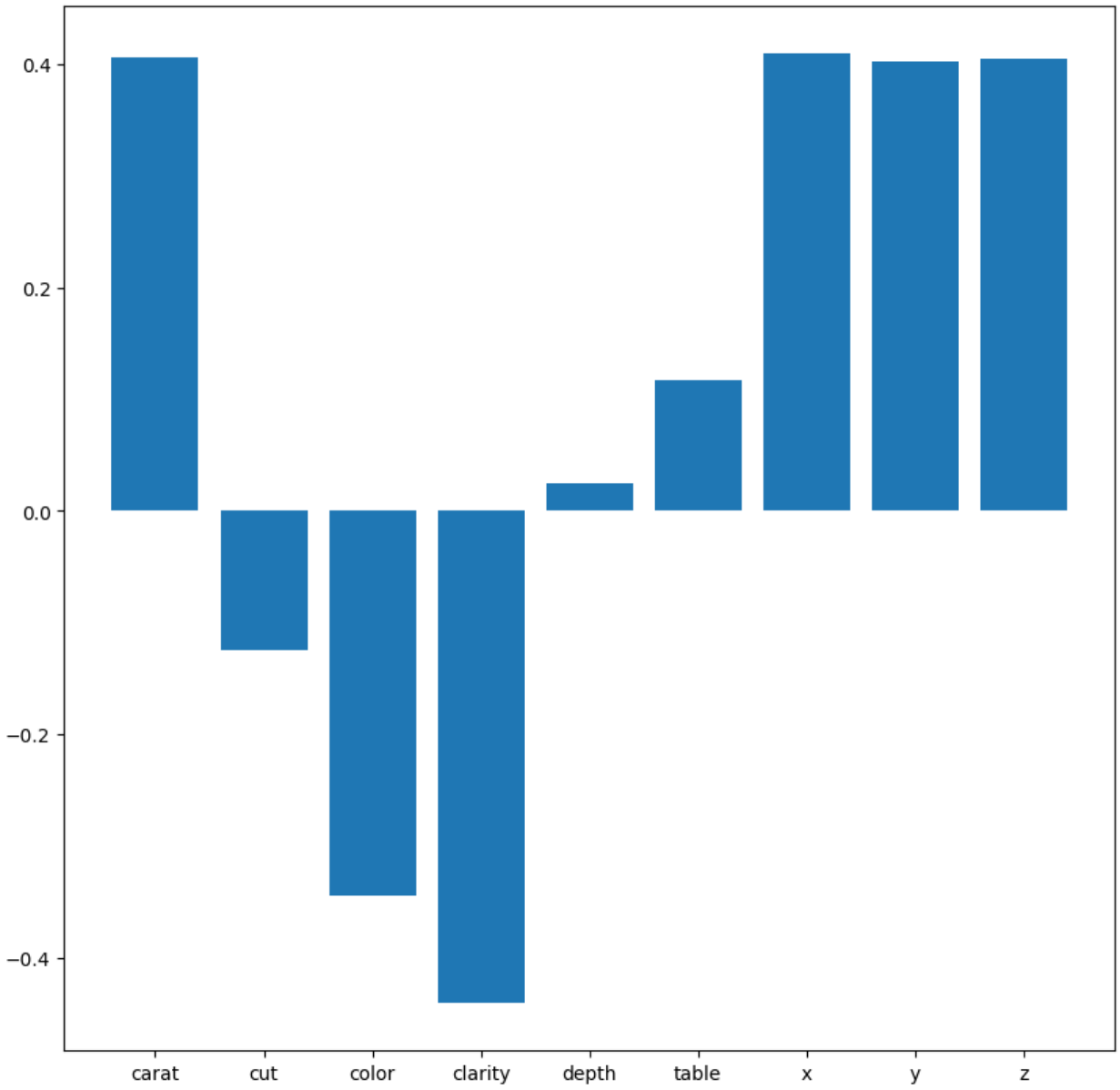



```
# видим сильную корреляцию между carat  и x, y, z. вес бриллианта коррелирует с размером
# сильная корреляция между x, y, z, размеры по разным осям коррелируют друг с другом
```

```
#14. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(data, y)
ratio = pca.explained_variance_ratio_.cumsum()
plt.figure(figsize=(10, 6))
plt.plot(ratio)
plt.xlabel('кол-во признаков')
plt.ylabel('процент объясненной дисперсии')
plt.grid(True)
plt.show()
# по графику видим, что для объяснения 90% дисперсии нужно минимум 4 компоненты
```




```
#15. Какой признак вносит наибольший вклад в первую компоненту?
plt.figure(figsize = (10,10))
plt.bar(range(len(pca.components_[0])), pca.components_[0])
plt.xticks(range(len(pca.components_[0])), data.columns)
plt.show()
```

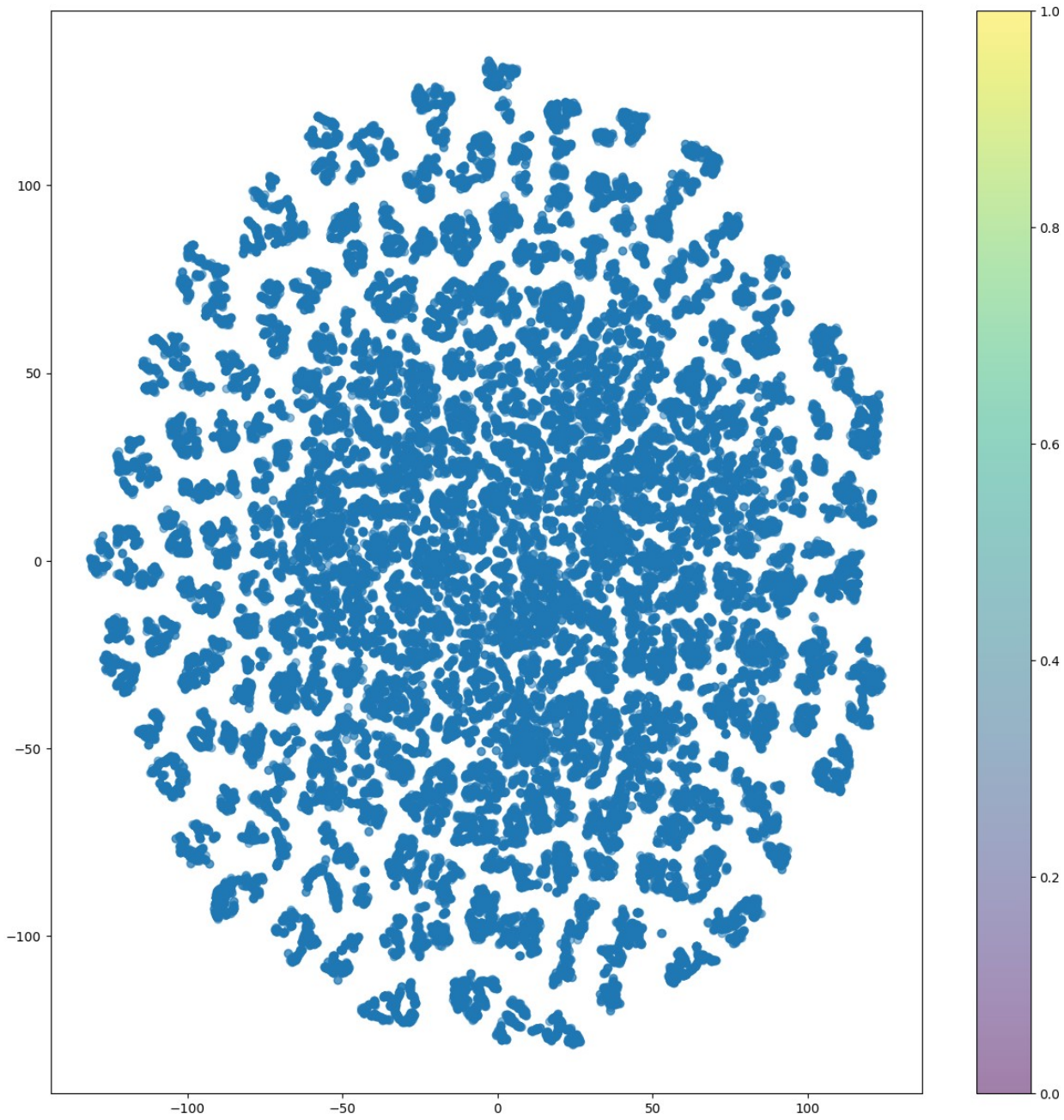



```
data.columns[pca.components_[0].argmax()]
#признак x
```

```
'x'
```

```
#16. Построить двухмерное представление данных с помощью алгоритма t-
SNE. На сколько кластеров визуально,
#на ваш взгляд, разделяется выборка? Объяснить смысл кластеров.
from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, random_state = 123)
data1 = tsne.fit_transform(data)
plt.figure(figsize = (15, 15))
plt.scatter(data1[:, 0], data1[:, 1], alpha = 0.5)
```

```
plt.colorbar()
plt.show()
```




```
#Кластеры представляют собой группы точек, которые находятся близко
друг к другу в двухмерном пространстве.
#Визуально определить количество кластеров не получается.
```