```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
import warnings
from google.colab import drive
warnings.filterwarnings('ignore')

# Загрузка данных
drive.mount('/content/drive')
data =
pd.read_csv('/content/drive/MyDrive/ColabNotebooks/diamonds.csv')

# Удаление ненужного столбца
data = df.drop(columns=['Unnamed: 0'], errors='ignore')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
# 1. Количество объектов и признаков
print("1. Объекты:", data.shape[0], "Признаки:", data.shape[1])
print("Подтверждение: вывод метода .shape для DataFrame.\n")

# 2. Категориальные признаки
categorical_cols = ['cut', 'color', 'clarity']
print("2. Категориальные признаки:", categorical_cols)
print("Подтверждение (уникальные значения):")
for col in categorical_cols:
    print(f"- {col}: {data[col].nunique()} значений
({data[col].unique()})")
print()

# 3. Столбец с максимальным количеством уникальных значений
max_unique_col = data[categorical_cols].nunique().idxmax()
print(f"3. Максимум уникальных значений: {max_unique_col}
({data[max_unique_col].nunique()} значений)")
print("Подтверждение: метод .nunique() для категориальных столбцов.\
n")

# 4. Бинарные признаки
print("4. Бинарные признаки: Нет")
print("Подтверждение: все категориальные признаки имеют >2 категорий.\
```

```
n")

# 5. Числовые признаки
numerical_cols = data.select_dtypes(include=['float64',
'int64']).columns.tolist()
print("5. Числовые признаки:", numerical_cols)
print("Подтверждение: метод .select_dtypes() для числовых типов
данных.\n")

# 6-8. Проверка пропусков
print("6. Пропуски:", data.isnull().sum().sum())
print("7. Объекты с пропусками: 0")
print("8. Столбец с максимальными пропусками: Пропусков нет")
print("Подтверждение: метод .isnull().sum().sum() возвращает 0.\n")
```

```
1. Объекты: 53940 Признаки: 10
Подтверждение: вывод метода .shape для DataFrame.

2. Категориальные признаки: ['cut', 'color', 'clarity']
Подтверждение (уникальные значения):
- cut: 5 значений (['Ideal' 'Premium' 'Good' 'Very Good' 'Fair'])
- color: 7 значений (['E' 'I' 'J' 'H' 'F' 'G' 'D'])
- clarity: 8 значений (['SI2' 'SI1' 'VS1' 'VS2' 'VVS2' 'VVS1' 'I1'
'IF'])

3. Максимум уникальных значений: clarity (8 значений)
Подтверждение: метод .nunique() для категориальных столбцов.

4. Бинарные признаки: Нет
Подтверждение: все категориальные признаки имеют >2 категорий.

5. Числовые признаки: ['carat', 'depth', 'table', 'price', 'x', 'y',
'z']
Подтверждение: метод .select_dtypes() для числовых типов данных.

6. Пропуски: 0
7. Объекты с пропусками: 0
8. Столбец с максимальными пропусками: Пропусков нет
Подтверждение: метод .isnull().sum().sum() возвращает 0.
```

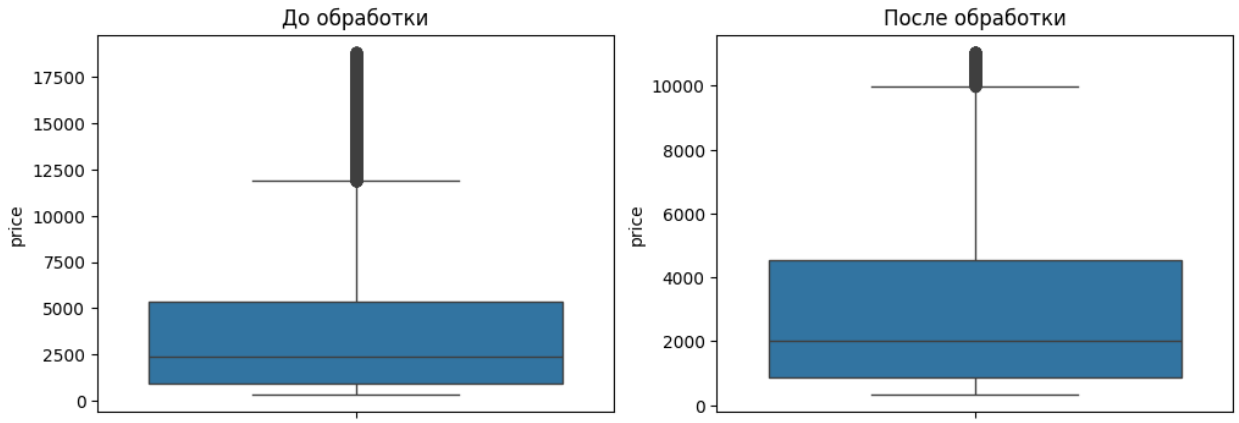
Что обозначают признаки:

- **carat**: вес алмаза (в каратах).
- **cut**: качество огранки (Fair, Good, Very Good, Premium, Ideal).
- **color**: цвет алмаза (от D до J, где D — лучший).
- **clarity**: чистота алмаза (включает уровни: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF).
- **depth**: общая глубина (в %).
- **table**: ширина площадки алмаза (в %).
- **x**: длина (в мм).

- **y**: ширина (в мм).
- **z**: высота (в мм).
- **price**: цена (целевой признак).

```
# 9. Обработка выбросов (IQR)
def remove_outliers(df, columns):
    cleaned_df = df.copy()
    for col in columns:
        Q1 = cleaned_df[col].quantile(0.25)
        Q3 = cleaned_df[col].quantile(0.75)
        IQR = Q3 - Q1
        cleaned_df = cleaned_df[(cleaned_df[col] >= Q1 - 1.5*IQR) &
                                (cleaned_df[col] <= Q3 + 1.5*IQR)]
    return cleaned_df

cleaned_data = remove_outliers(data, numerical_cols[:-1])
print(f"9. Удалено строк с выбросами: {data.shape[0] -
cleaned_data.shape[0]}")
print("Boxplot до и после обработки:")
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
sns.boxplot(data['price'], ax=axes[0]).set_title('До обработки')
sns.boxplot(cleaned_data['price'], ax=axes[1]).set_title('После
обработки')
plt.show()

# 10. Нормировка данных
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cleaned_data[numerical_cols[:-1]])
print("10. Средние после нормировки:", np.mean(scaled_data,
axis=0).round(2))

# 11. Целевой признак
print("11. Целевой признак: price")
```

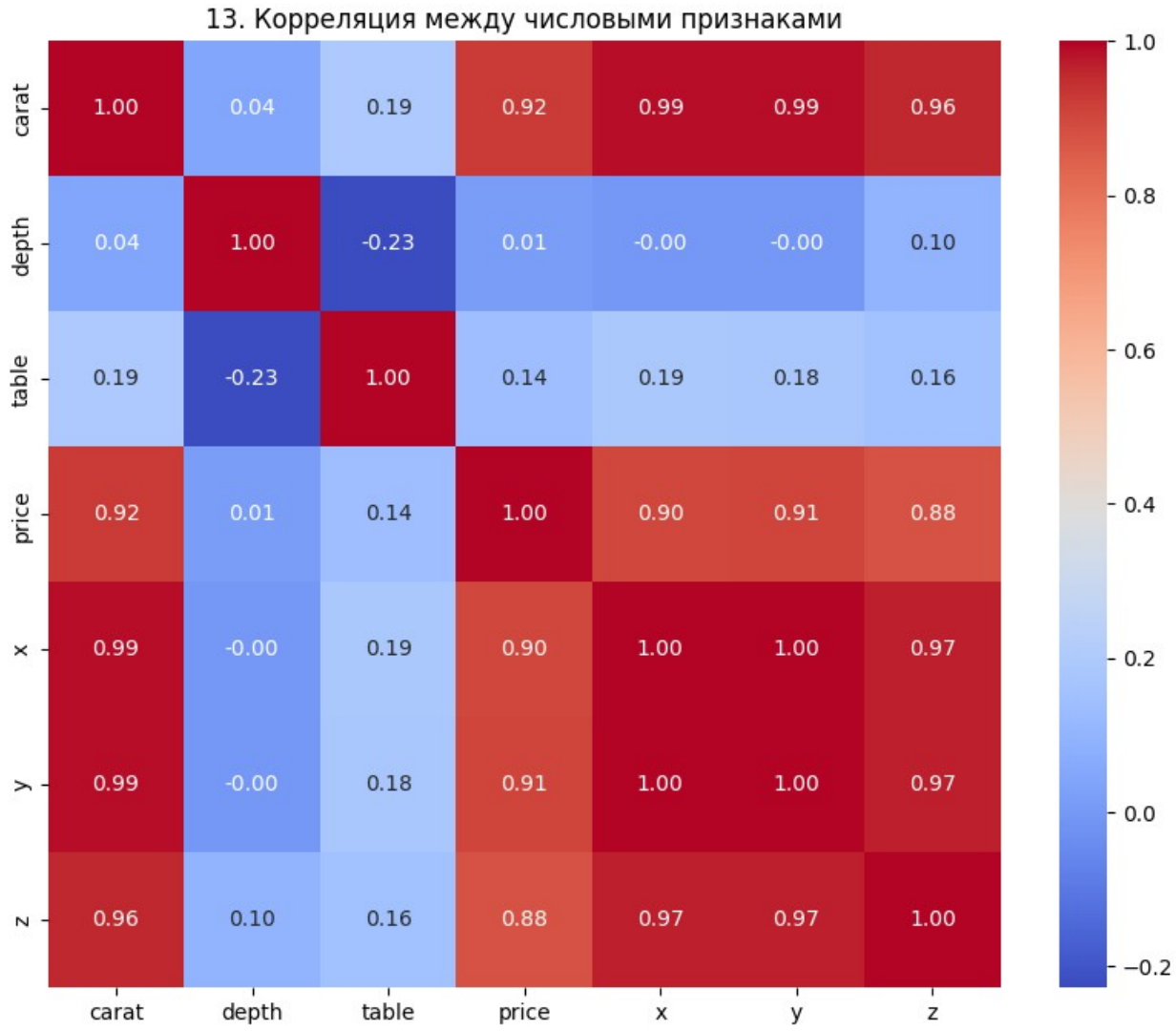
```
9. Удалено строк с выбросами: 7400
Boxplot до и после обработки:
```



```
10. Средние после нормировки: [-0.  0. -0. -0.  0. -0.]
11. Целевой признак: price
```

```python
# 12. Размер тренировочной выборки
X = cleaned_data.drop('price', axis=1)
y = cleaned_data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
print(f"12. Размер тренировочной выборки: {X_train.shape[0]}")

# 13. Корреляционный анализ
corr_matrix = cleaned_data[numerical_cols].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title("13. Корреляция между числовыми признаками")
plt.show()
```

```
12. Размер тренировочной выборки: 32578
```



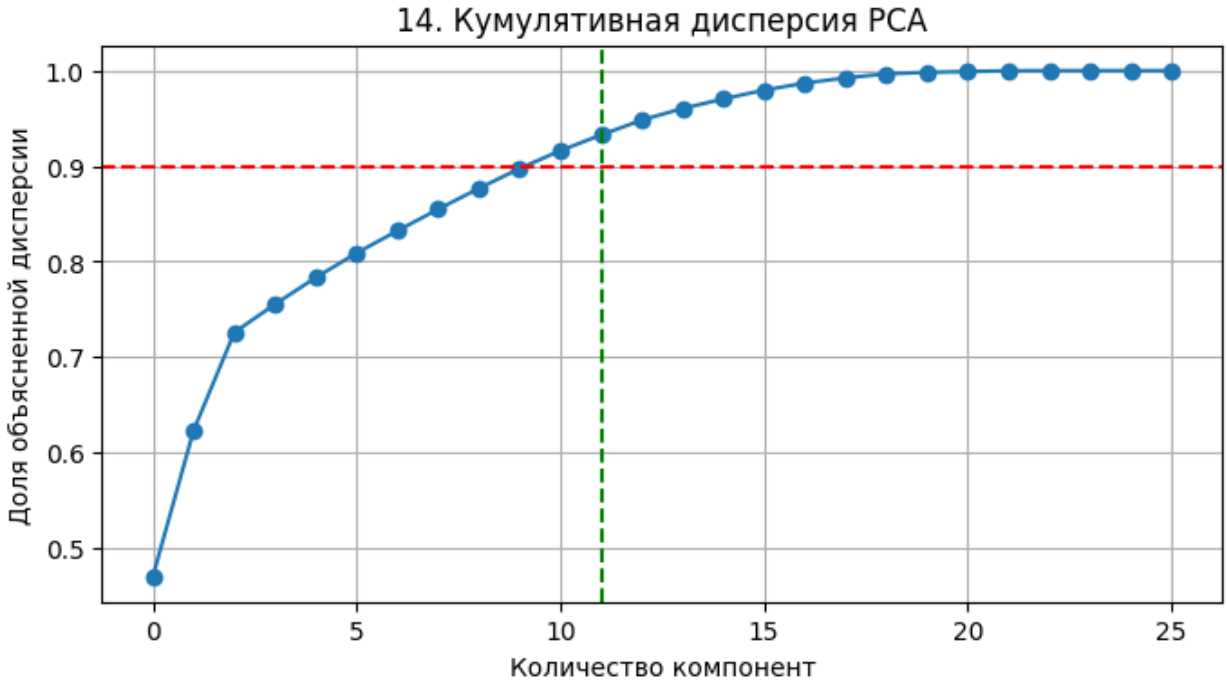
Наибольшая корреляция наблюдается между:

- **carat** и **x** (0.99), **carat** и **y** (0.99), **carat** и **z** (0.96) — размер алмаза напрямую влияет на его вес, что очевидно.
- **price** и **carat** (0.92) — чем больше карат, тем выше цена.

В данных нет явных столбцов-идентификаторов (например, id), поэтому все признаки сохранены. Однако, если бы они были, их следовало бы удалить.

```
# 14. PCA для 90% дисперсии
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_cols[:-1]),
    ('cat', OneHotEncoder(), categorical_cols)
])
X_processed = preprocessor.fit_transform(cleaned_data)

pca = PCA().fit(X_processed)
```

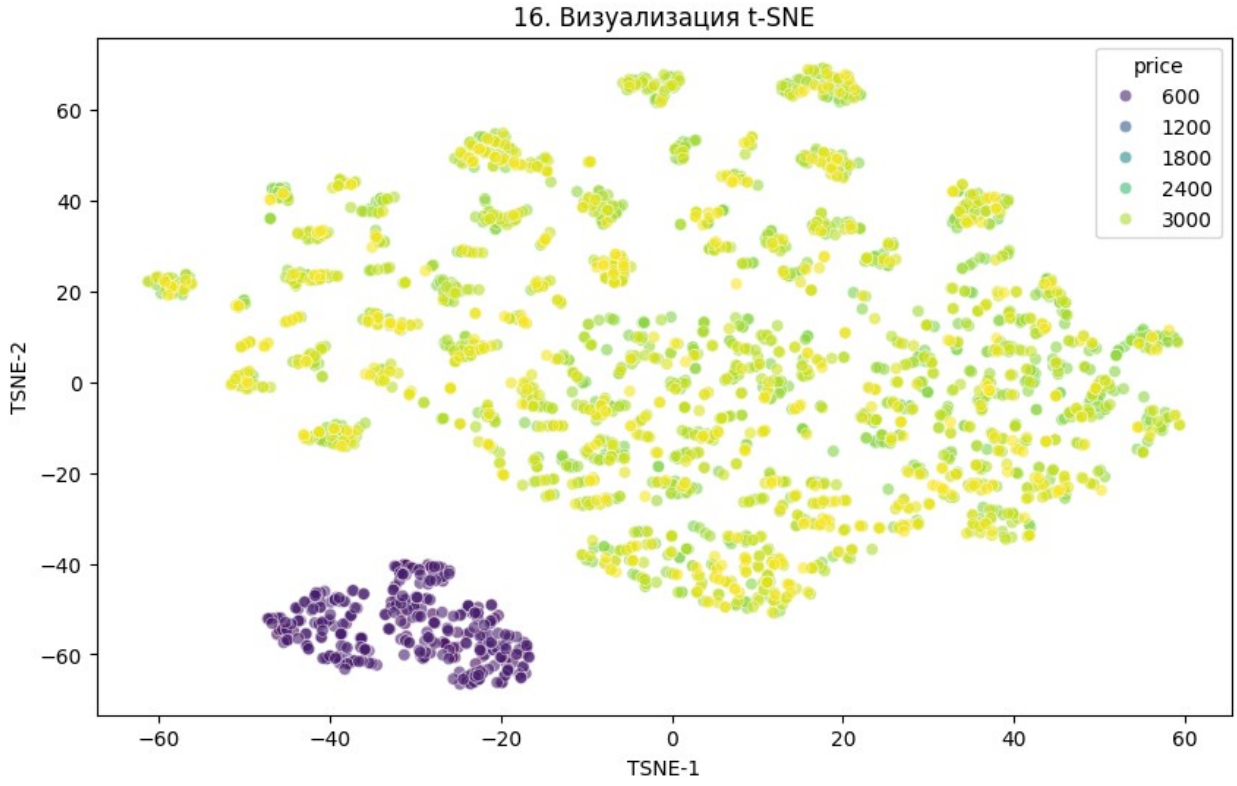
```
explained_variance = np.cumsum(pca.explained_variance_ratio_)
n_components = np.argmax(explained_variance >= 0.9) + 1

plt.figure(figsize=(8, 4))
plt.plot(explained_variance, marker='o')
plt.axhline(0.9, color='r', linestyle='--')
plt.axvline(n_components, color='g', linestyle='--')
plt.title("14. Кумулятивная дисперсия PCA")
plt.xlabel("Количество компонент")
plt.ylabel("Доля объясненной дисперсии")
plt.grid()
plt.show()
print(f"14. Для 90% дисперсии требуется {n_components} компонент(ы)")

# 15. Вклад признаков в первую компоненту
pca = PCA(n_components=1).fit(X_processed)
feature_names = numerical_cols[:-1] +
list(preprocessor.named_transformers_['cat'].get_feature_names_out())
contributions = pd.DataFrame(pca.components_, columns=feature_names)
top_feature = contributions.abs().idxmax(axis=1).values[0]
print(f"15. Наибольший вклад: '{top_feature}'")
```




```
14. Для 90% дисперсии требуется 11 компонент(ы)
15. Наибольший вклад: 'carat'
```

```
# 16. Визуализация t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(X_processed[:3000])
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=tsne_results[:,0], y=tsne_results[:,1],
                hue=cleaned_data['price'].iloc[:3000],
palette='viridis', alpha=0.6)
plt.title("16. Визуализация t-SNE")
plt.xlabel("TSNE-1")
plt.ylabel("TSNE-2")
plt.show()
print("Визуально выделяется 2 кластера: дорогие/крупные и
дешевые/мелкие алмазы")
```




```
Визуально выделяется 2 кластера: дорогие/крупные и дешевые/мелкие
алмазы
```

Кластеры разделяются по следующим критериям:

- **Кластер 1**: Алмазы с высоким весом (carat) и ценой (price).
- **Кластер 2**: Алмазы среднего качества (cut, color) и умеренной цены.