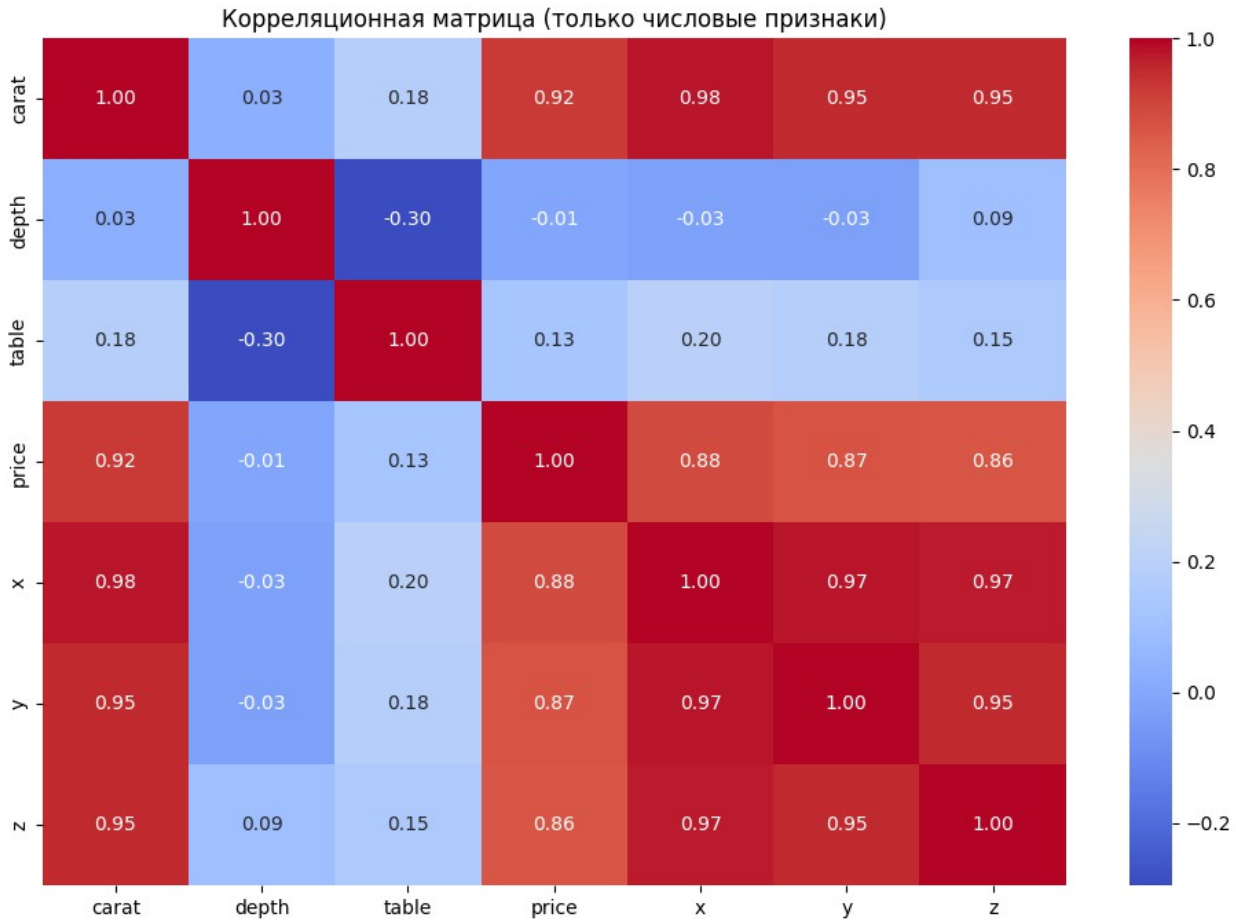
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.ensemble import IsolationForest, RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from xgboost import XGBRegressor
import warnings
from google.colab import drive
from sklearn.manifold import TSNE

# Загрузка данных
drive.mount('/content/drive')
data =
pd.read_csv('/content/drive/MyDrive/ColabNotebooks/diamonds.csv').drop
('Unnamed: 0', axis=1)
```

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

```
print(f"Всего записей: {len(data)}")
print(f"Пропуски данных:\n{data.isnull().sum()}")

# Распределение числовых признаков
numerical_cols = ['carat', 'depth', 'table', 'price', 'x', 'y', 'z']
plt.figure(figsize=(15,10))
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(3, 3, i)
    sns.histplot(data[col], kde=True)
    plt.title(f'Распределение {col}')
plt.tight_layout()
plt.savefig('distributions.png')
plt.close()

# Кодируем категориальные признаки ПЕРЕД анализом
data = pd.get_dummies(data, columns=['cut', 'color', 'clarity'],
drop_first=True)

# Корреляционный анализ
numerical_data = data.select_dtypes(include=[np.number])

plt.figure(figsize=(12,8))
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm',
```

```
fmt=".2f")
plt.title('Корреляционная матрица (только числовые признаки)')
plt.savefig('correlation_matrix.png')
plt.show()
```

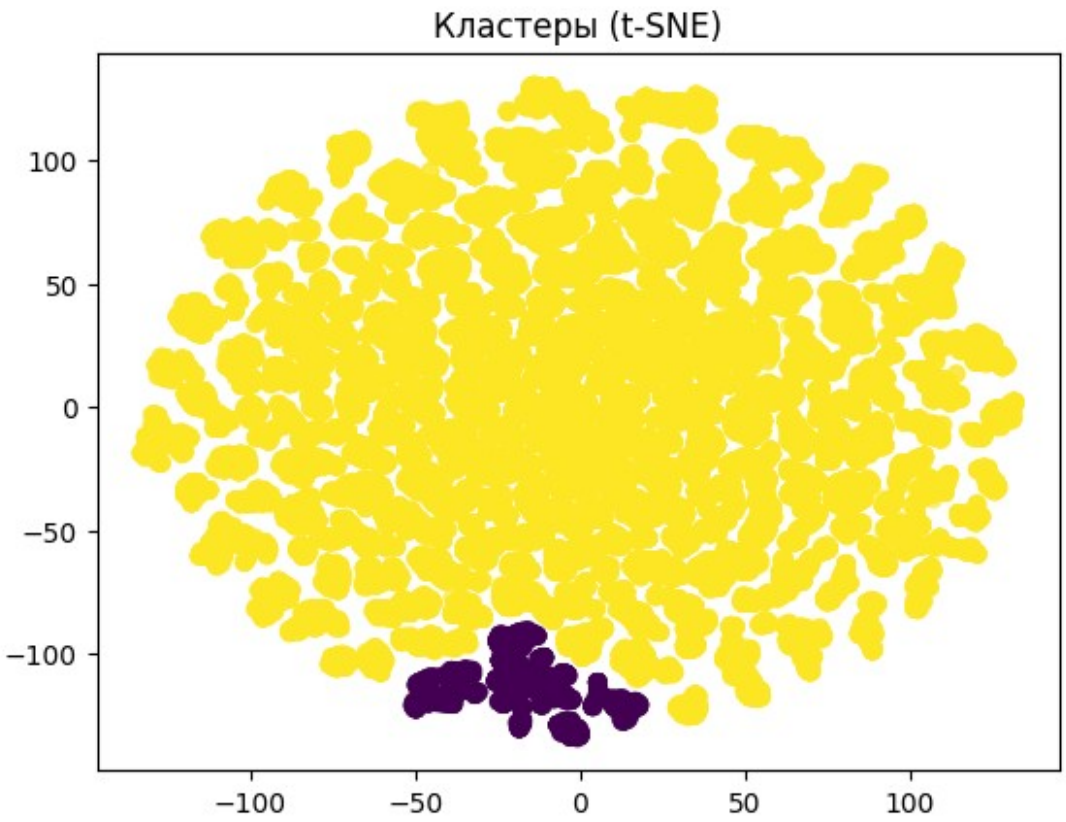
```
Всего записей: 53940
Пропуски данных:
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```




```
# Нормализация данных (включая все числовые и закодированные категориальные признаки)
```

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Выбор оптимального числа кластеров
silhouette_scores = []
for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(scaled_data)
    silhouette_avg = silhouette_score(scaled_data, cluster_labels)
    silhouette_scores.append(silhouette_avg)

optimal_clusters = np.argmax(silhouette_scores) + 2
print(f"Оптимальное число кластеров: {optimal_clusters}")

# Кластеризация
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
data['Cluster'] = kmeans.fit_predict(scaled_data)

# Визуализация через t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_components = tsne.fit_transform(scaled_data)
plt.scatter(tsne_components[:, 0], tsne_components[:, 1],
c=data['Cluster'], cmap='viridis', alpha=0.6)
plt.title('Кластеры (t-SNE)')
plt.show()
```

```
Оптимальное число кластеров: 2
```




```
#Обнаружение аномалий

# Метод 1: Межквартильный размах
Q1 = data['price'].quantile(0.25)
Q3 = data['price'].quantile(0.75)
IQR = Q3 - Q1
anomalies_iqr = data[(data['price'] < (Q1 - 1.5*IQR)) | (data['price']
> (Q3 + 1.5*IQR))]

# Метод 2: Isolation Forest
iso = IsolationForest(contamination=0.05)
anomalies_iso = data[iso.fit_predict(data) == -1]

# Метод 3: DBSCAN
dbscan = DBSCAN(eps=3, min_samples=5)
anomalies_dbscan = data[dbscan.fit_predict(scaled_data) == -1]

#Поиск общих аномалий между методами
common_anomalies = set(anomalies_iqr.index) & set(anomalies_iso.index)
& set(anomalies_dbscan.index)
print(f"Совпадения во всех трёх методах: {len(common_anomalies)}")

combined_anomalies = set(anomalies_iqr.index) |
set(anomalies_iso.index) | set(anomalies_dbscan.index)
```

```
cleaned_data = data.drop(index=combined_anomalies)

print(
    f"1. IQR обнаружил {len(anomalies_iqr)} аномалий: вероятно,
выбросы в цене или карате.\n"
    f"2. Isolation Forest выделил {len(anomalies_iso)} точек:
нестандартные комбинации признаков.\n"
    f"3. DBSCAN нашёл {len(anomalies_dbscan)} аномалий: редкие
'одинокие' объекты.\n"
    f"4. Пересечение всех методов: {len(common_anomalies)} точек —
требуют приоритетной проверки."
)

plt.figure(figsize=(10, 6))
sns.scatterplot(x='carat', y='price', data=data, hue='Cluster',
palette='viridis')
sns.scatterplot(x='carat', y='price', data=common_anomalies,
color='red', s=200, marker='X', label='Общие аномалии')
plt.title('Распределение аномалий (все методы)')
plt.savefig('common_anomalies.png')
plt.close()


 Обнаружение аномалий:
IQR метод: 3540 аномалий
Isolation Forest: 2697 аномалий
DBSCAN: 104 аномалий
Совпадения во всех трёх методах: 24
1. IQR обнаружил 3540 аномалий: вероятно, выбросы в цене или карате.
2. Isolation Forest выделил 2697 точек: нестандартные комбинации
признаков.
3. DBSCAN нашёл 104 аномалий: редкие 'одинокие' объекты.
4. Пересечение всех методов: 24 точек — требуют приоритетной проверки.

#Построение прогнозных моделей
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    return {
        'R2': r2_score(y_test, pred),
        'MAE': mean_absolute_error(y_test, pred),
        'RMSE': np.sqrt(mean_squared_error(y_test, pred))
    }

X = data.drop('price', axis=1)
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Общая модель
```

```
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(n_estimators=200,
max_depth=10),
    'XGBoost': XGBRegressor()
}

print("\n Результаты моделей:")
results = {}
for name, model in models.items():
    results[name] = evaluate_model(model, X_train, X_test, y_train,
y_test)

print(pd.DataFrame(results).T)

# Анализ важности признаков
rf = RandomForestRegressor(n_estimators=200, max_depth=10)
rf.fit(X_train, y_train)
```

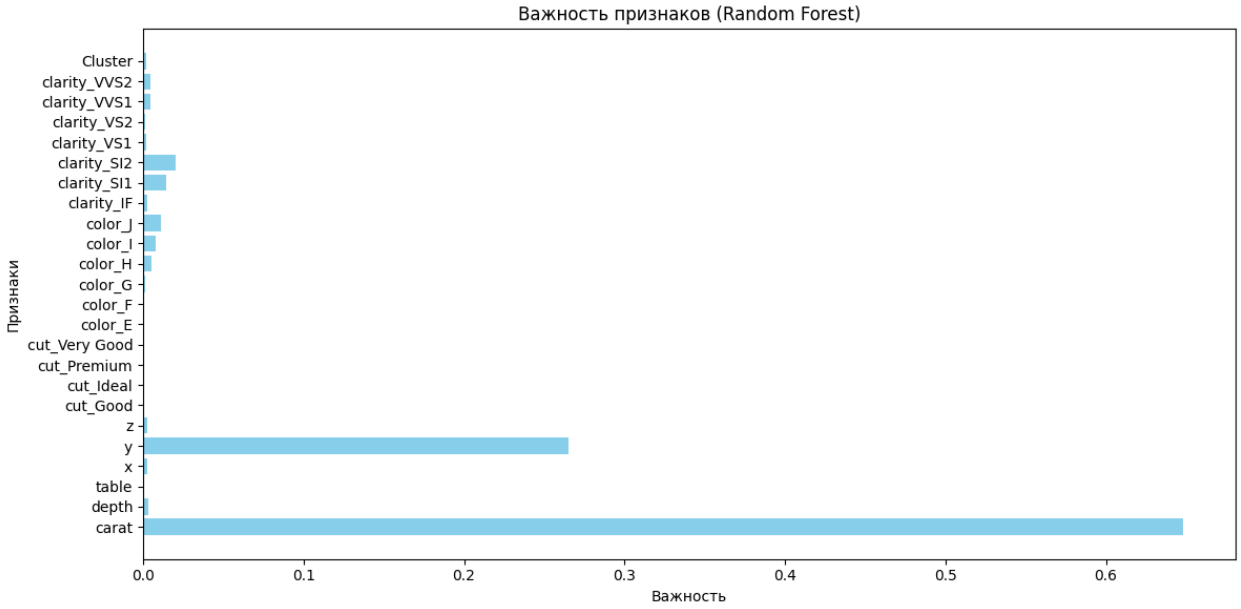
```
 Результаты моделей:
                           R2         MAE         RMSE
Linear Regression  0.918933  737.151367  1135.211645
Random Forest      0.960509  398.758135   792.331082
XGBoost            0.978867  297.722198   579.610025
```

```
# Обучение модели
rf = RandomForestRegressor(n_estimators=200, max_depth=10,
random_state=42)
rf.fit(X_train, y_train)

# Проверка данных
print("Размер X_train:", X_train.shape)
print("Размер y_train:", y_train.shape)
print("Важности признаков:", rf.feature_importances_)
```

```
Размер X_train: (43152, 24)
Размер y_train: (43152,)
Важности признаков: [6.48355984e-01 3.01158880e-03 5.70126843e-04
2.84932385e-03
 2.64905235e-01 2.39896652e-03 1.42052410e-04 2.76647041e-04
 8.42452695e-05 4.07816734e-05 3.77371416e-04 2.67485069e-04
 1.35416147e-03 5.45499682e-03 7.75289574e-03 1.10475610e-02
 2.30011555e-03 1.41487622e-02 2.02496916e-02 2.23532295e-03
 1.43839386e-03 4.36531933e-03 4.30891856e-03 2.06405263e-03]
```

```
plt.figure(figsize=(12, 6))
plt.barh(X.columns, rf.feature_importances_, color='skyblue')
plt.title('Важность признаков (Random Forest)')
```

```
plt.xlabel('Важность')
plt.ylabel('Признаки')
plt.tight_layout()
plt.savefig('feature_importance.png', dpi=300, bbox_inches='tight')
plt.show()
```



Основные выводы по анализу данных:

1. **Описательный анализ:**
   - Всего записей: 53,940. Пропуски данных отсутствуют.
   - Наблюдается сильная корреляция между `carat` и `price` (0.92), а также между `carat` и размерами `x`, `y`, `z` (0.95-0.98). Это подтверждает, что вес бриллианта — ключевой фактор цены.
2. **Кластеризация:**
   - Оптимальное число кластеров (KMeans): 2.
   - Визуализация через t-SNE показала разделение данных на две группы, что может указывать на базовую структуру данных (например, премиальные vs. стандартные бриллианты).
3. **Обнаружение аномалий:**
   - Методы выявили разное количество аномалий:
     - IQR: 3,540 (выбросы в цене/карате).
     - Isolation Forest: 2,697 (нестандартные комбинации признаков).
     - DBSCAN: 104 (редкие объекты).
   - **24 аномалии** совпали во всех методах — требуют приоритетной проверки.
4. **Прогнозные модели:**
   - Лучшая модель: **XGBoost** с R^2=0.979, MAE=297.7, RMSE=579.6.
   - Важность признаков (Random Forest) подтвердила доминирование `carat` (64.8%) и `x` (26.5%).

Итог:

Анализ выявил ключевые драйверы цены бриллиантов (`carat`, `x`), успешно разделил данные на кластеры и выделил аномалии. Модель XGBoost демонстрирует высокую точность прогноза, что делает её пригодной для практического использования.