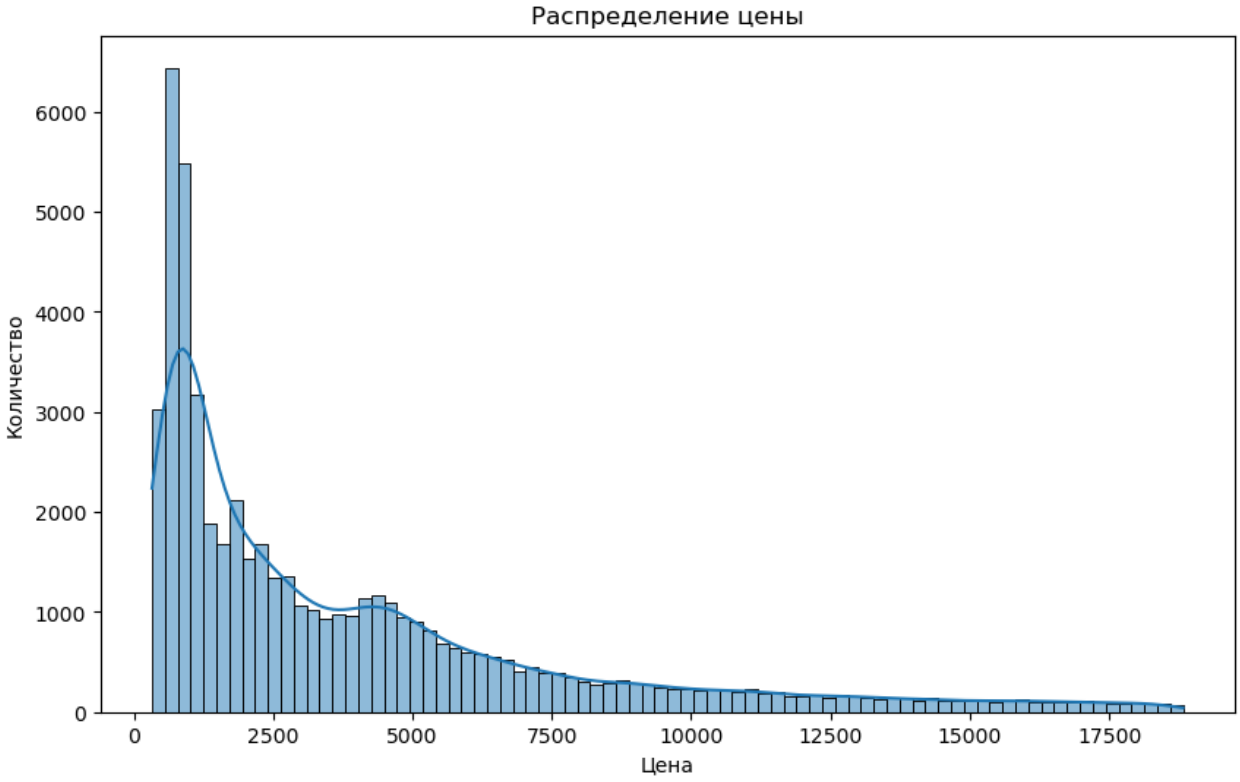
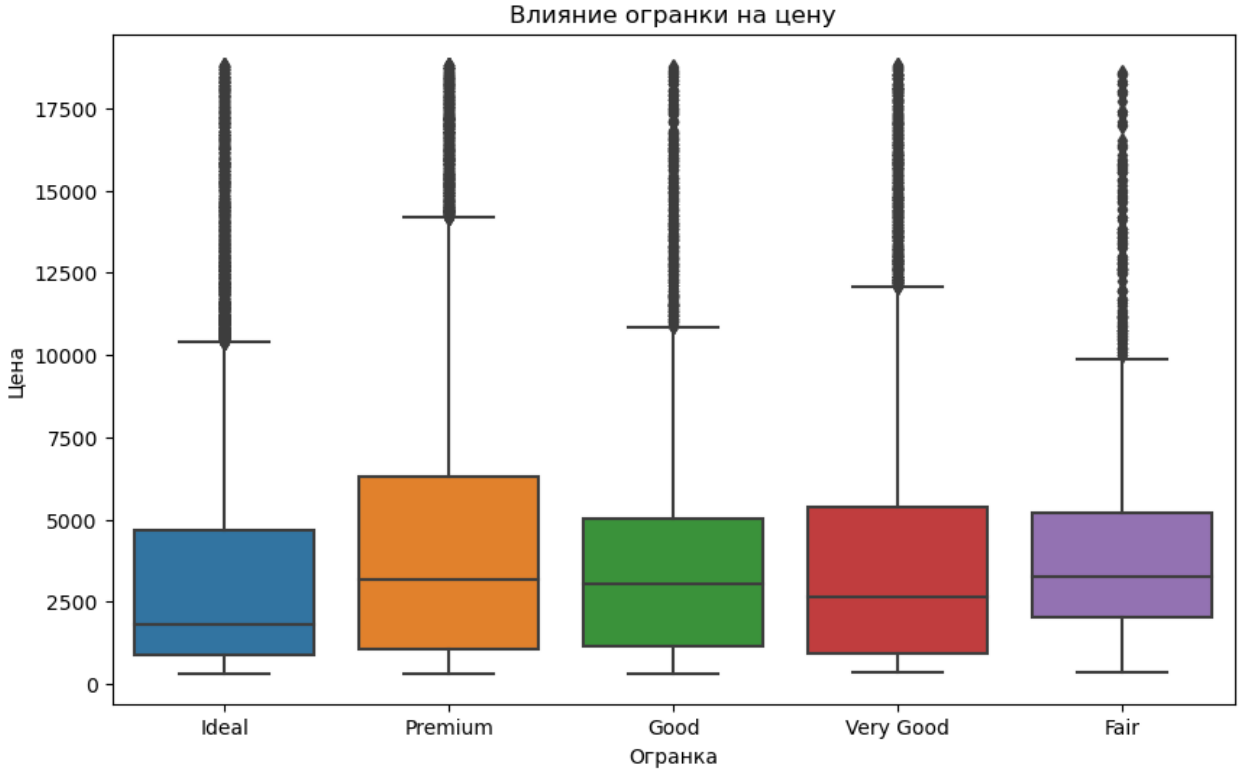
In [ ]:

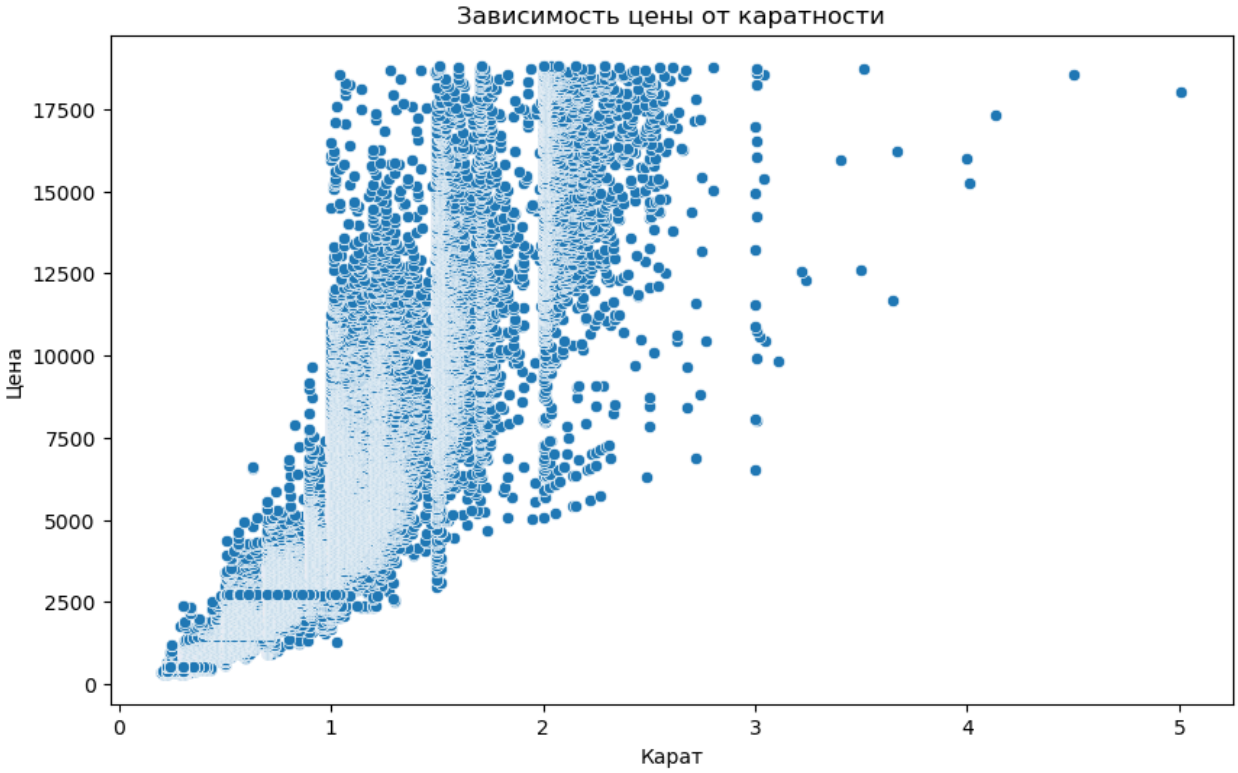
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder

# Загрузка данных
data = pd.read_csv('diamonds.csv')

# Проверка на пропущенные значения
print(data.isnull().sum())

# Описательный анализ данных
print(data.describe())

# Визуализация распределения цены
plt.figure(figsize=(10, 6))
sns.histplot(data['price'], kde=True)
plt.title('Распределение цены')
plt.xlabel('Цена')
plt.ylabel('Количество')
plt.savefig('price_distribution.pdf') # Сохранение графика в PDF
plt.show()

# Визуализация влияния качества огранки на цену до преобразования перемен
plt.figure(figsize=(10, 6))
sns.boxplot(x='cut', y='price', data=data)
plt.title('Влияние огранки на цену')
plt.xlabel('Огранка')
plt.ylabel('Цена')
plt.savefig('price_by_cut.pdf') # Сохранение графика в PDF
plt.show()

# Преобразование категориальных данных с использованием OneHotEncoder
encoder = OneHotEncoder(drop='first')
encoded_features = encoder.fit_transform(data[['cut', 'color', 'clarity']
encoded_features_df = pd.DataFrame(encoded_features.toarray(), columns=en
data_encoded = pd.concat([data.drop(['cut', 'color', 'clarity'], axis=1),

# Визуализация зависимости цены от каратности
plt.figure(figsize=(10, 6))
sns.scatterplot(x='carat', y='price', data=data_encoded)
plt.title('Зависимость цены от каратности')
plt.xlabel('Карат')
plt.ylabel('Цена')
plt.savefig('price_by_carat.pdf') # Сохранение графика в PDF
plt.show()

# Разделение данных на обучающую, валидационную и тестовую выборки
X = data_encoded.drop('price', axis=1)
y = data['price']
```

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2,
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size

# Настройка параметров моделей с использованием GridSearchCV
param_grid_lr = {'fit_intercept': [True, False]}
param_grid_rf = {'n_estimators': [100, 200], 'max_depth': [None, 10, 20,

grid_search_lr = GridSearchCV(LinearRegression(), param_grid_lr, cv=5, sc
grid_search_rf = GridSearchCV(RandomForestRegressor(random_state=42), par

grid_search_lr.fit(X_train, y_train)
grid_search_rf.fit(X_train, y_train)

# Выбор лучшей модели на основе результатов GridSearchCV
if grid_search_lr.best_score_ < grid_search_rf.best_score_:
    best_model = grid_search_lr.best_estimator_
    print("Лучшая модель: Линейная регрессия")
else:
    best_model = grid_search_rf.best_estimator_
    print("Лучшая модель: Случайный лес")

# Оценка лучшей модели на тестовой выборке
y_test_pred = best_model.predict(X_test)
print("Тестовое MSE:", mean_squared_error(y_test, y_test_pred))
```

```
Unnamed: 0    0
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
price         0
x             0
y             0
z             0
dtype: int64
         Unnamed: 0         carat         depth         table         pric
e  \
count  53940.000000  53940.000000  53940.000000  53940.000000  53940.00000
0
mean   26970.500000      0.797940     61.749405     57.457184   3932.79972
2
std    15571.281097      0.474011      1.432621      2.234491   3989.43973
8
min        1.000000      0.200000     43.000000     43.000000    326.00000
0
25%    13485.750000      0.400000     61.000000     56.000000    950.00000
0
50%    26970.500000      0.700000     61.800000     57.000000   2401.00000
0
75%    40455.250000      1.040000     62.500000     59.000000   5324.25000
0
max    53940.000000      5.010000     79.000000     95.000000  18823.00000
0

                 x             y             z
count  53940.000000  53940.000000  53940.000000
mean       5.731157      5.734526      3.538734
std        1.121761      1.142135      0.705699
min        0.000000      0.000000      0.000000
25%        4.710000      4.720000      2.910000
50%        5.700000      5.710000      3.530000
75%        6.540000      6.540000      4.040000
max       10.740000     58.900000     31.800000
```


Распределение цены
Количество
Цена
0
1000
2000
3000
4000
5000
6000
0
2500
5000
7500
10000
12500
15000
17500


Влияние огранки на цену
Цена
Огранка
0
2500
5000
7500
10000
12500
15000
17500
Ideal
Premium
Good
Very Good
Fair


Зависимость цены от каратности
Цена
Карат
0
2500
5000
7500
10000
12500
15000
17500
0
1
2
3
4
5

In [ ]: