Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Лабораторная работа №4

Выполнил:
студент группы ИУ5-63Б
Комаров Д. С.

Проверил:
Гапанюк Ю. Е.

2024 г.

Ход работы:

Загрузка датасета

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler,
StandardScaler
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor, export_graphviz,
export_text
from sklearn.svm import SVR
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV
from IPython.display import Image
from IPython.core.display import HTML
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import
OrdinalEncoder,LabelEncoder,StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.compose import make_column_transformer
from sklearn.model_selection import train_test_split

data = pd.read_csv('sample_data/cleaned_all_phones.csv')
data.head()
```

{"type":"dataframe","variable_name":"data"}

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1512 entries, 0 to 1511
Data columns (total 22 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   phone_name         1512 non-null   object
 1   brand              1512 non-null   object
 2   os                 1512 non-null   object
 3   inches             1512 non-null   float64
 4   resolution         1512 non-null   object
 5   battery            1512 non-null   int64
 6   battery_type       1512 non-null   object
 7   ram(GB)            1512 non-null   int64
 8   announcement_date  1512 non-null   object
 9   weight(g)          1512 non-null   float64
 10  storage(GB)        1512 non-null   int64
 11  video_720p         1512 non-null   bool
 12  video_1080p        1512 non-null   bool
 13  video_4K           1512 non-null   bool
 14  video_8K           1512 non-null   bool
 15  video_30fps        1512 non-null   bool
```

```
 16  video_60fps        1512 non-null   bool
 17  video_120fps       1512 non-null   bool
 18  video_240fps       1512 non-null   bool
 19  video_480fps       1512 non-null   bool
 20  video_960fps       1512 non-null   bool
 21  price(USD)         1512 non-null   float64
dtypes: bool(10), float64(3), int64(3), object(6)
memory usage: 156.6+ KB
```

Чистка данных

```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

```
phone_name           0
brand                0
os                   0
inches               0
resolution           0
battery              0
battery_type         0
ram(GB)              0
announcement_date    0
weight(g)            0
storage(GB)          0
video_720p           0
video_1080p          0
video_4K             0
video_8K             0
video_30fps          0
video_60fps          0
video_120fps         0
video_240fps         0
video_480fps         0
video_960fps         0
price(USD)           0
dtype: int64
```

```
data[['width', 'height']] = data.resolution.str.split('x', expand =
True).astype('int64')

data.head()
```

```
{"type":"dataframe","variable_name":"data"}
```

```
data['announcement_year'] = data.announcement_date.apply(lambda x:
x.split('-')[0]).astype('int64')
data.head()
```

```
{"type":"dataframe","variable_name":"data"}
```

```
data.drop(columns = ['announcement_date', 'resolution', 'phone_name'],
inplace = True)

data.describe()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"inches\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 532.5310851556795,\n
\"min\": 0.4770430982109062,\n   \"max\": 1512.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
6.4224603174603185,\n   6.5,\n   1512.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"battery\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 2148.127173043608,\n
\"min\": 784.6070221906537,\n   \"max\": 7250.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
4389.798941798942,\n   4500.0,\n   1512.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"ram(GB)\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 531.8729184957466,\n
\"min\": 1.0,\n   \"max\": 1512.0,\n   \"num_unique_values\":
7,\n   \"samples\": [\n   1512.0,\n
6.6838624338624335,\n   8.0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"weight(g)\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 482.7337788722055,\n   \"min\":
26.20011485546831,\n   \"max\": 1512.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
187.6362433862434,\n   187.0,\n   1512.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"storage(GB)\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 507.5688180604079,\n
\"min\": 1.0,\n   \"max\": 1512.0,\n   \"num_unique_values\":
7,\n   \"samples\": [\n   1512.0,\n
109.16468253968254,\n   128.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"price(USD)\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 803.235184359574,\n
\"min\": 40.0,\n   \"max\": 2300.0,\n   \"num_unique_values\":
8,\n   \"samples\": [\n   337.8470357142857,\n
260.0,\n   1512.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"width\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1117.8750058935389,\n   \"min\": 253.48894039516676,\n
\"max\": 3840.0,\n   \"num_unique_values\": 7,\n   \"samples\":
[\n   1512.0,\n   1035.212962962963,\n   1080.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"height\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1057.435936394939,\n
\"min\": 469.73457812549356,\n   \"max\": 3840.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   1512.0,\n
2207.190476190476,\n   2400.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"announcement_year\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 710.6079645347033,\n
\"min\": 1.7001902014840866,\n   \"max\": 2023.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
2020.4100529100529,\n   2021.0,\n   1512.0\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
categorical_cols = list(data.select_dtypes(include='object').columns)
onehot = OneHotEncoder(sparse_output=False)
encoded = onehot.fit_transform(data[categorical_cols])
encoded_cols = onehot.get_feature_names_out()
encoded_df = pd.DataFrame(encoded, columns=encoded_cols)

encoded_df
```

{"type":"dataframe","variable_name":"encoded_df"}

```
final_df = pd.concat([data, encoded_df],
axis=1).drop(columns=categorical_cols)

final_df
```

{"type":"dataframe","variable_name":"final_df"}

```
bool_cols = list(data.select_dtypes(include='bool').columns)
final_df[bool_cols] = final_df[bool_cols].replace({True: 1, False: 0})

final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1512 entries, 0 to 1511
Data columns (total 68 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   inches                      1512 non-null   float64
 1   battery                     1512 non-null   int64
 2   ram(GB)                     1512 non-null   int64
 3   weight(g)                   1512 non-null   float64
 4   storage(GB)                 1512 non-null   int64
 5   video_720p                  1512 non-null   int64
 6   video_1080p                 1512 non-null   int64
 7   video_4K                    1512 non-null   int64
 8   video_8K                    1512 non-null   int64
 9   video_30fps                 1512 non-null   int64
 10  video_60fps                 1512 non-null   int64
 11  video_120fps                1512 non-null   int64
 12  video_240fps                1512 non-null   int64
 13  video_480fps                1512 non-null   int64
 14  video_960fps                1512 non-null   int64
 15  price(USD)                  1512 non-null   float64
 16  width                       1512 non-null   int64
 17  height                      1512 non-null   int64
 18  announcement_year           1512 non-null   int64
 19  brand_Apple                 1512 non-null   float64
 20  brand_Google                1512 non-null   float64
 21  brand_Honor                 1512 non-null   float64
 22  brand_Huawei                1512 non-null   float64
 23  brand_LG                    1512 non-null   float64
 24  brand_Lenovo                1512 non-null   float64
 25  brand_OnePlus               1512 non-null   float64
```

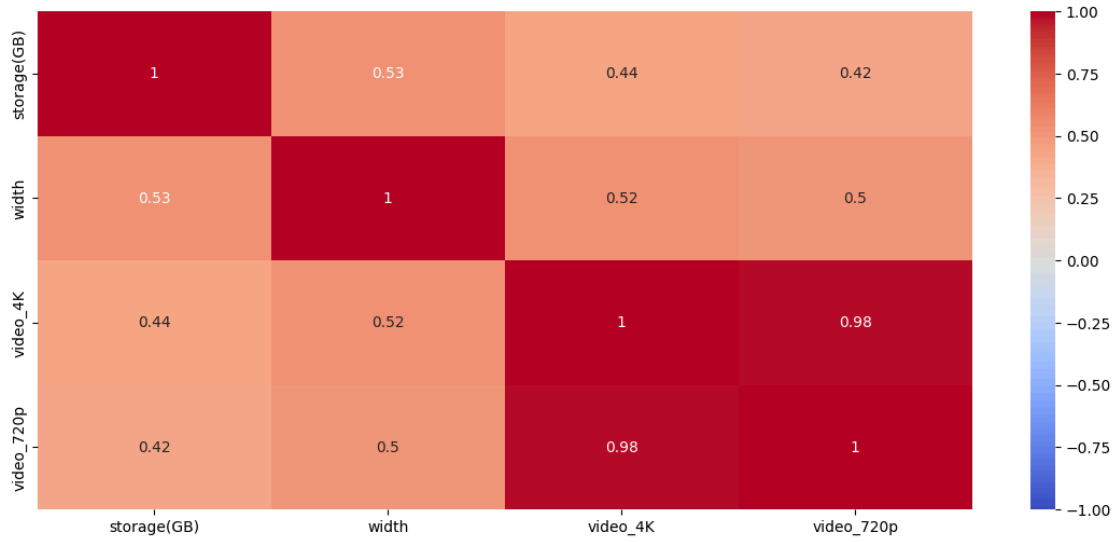
```
 26  brand_Oppo                 1512 non-null   float64
 27  brand_Realme               1512 non-null   float64
 28  brand_Samsung              1512 non-null   float64
 29  brand_Sony                 1512 non-null   float64
 30  brand_Vivo                 1512 non-null   float64
 31  brand_Xiaomi               1512 non-null   float64
 32  os_Android                 1512 non-null   float64
 33  os_Android 10              1512 non-null   float64
 34  os_Android 10/ Android 11  1512 non-null   float64
 35  os_Android 11              1512 non-null   float64
 36  os_Android 12              1512 non-null   float64
 37  os_Android 12 or 13        1512 non-null   float64
 38  os_Android 12L             1512 non-null   float64
 39  os_Android 13              1512 non-null   float64
 40  os_Android 5.1             1512 non-null   float64
 41  os_Android 6               1512 non-null   float64
 42  os_Android 6.0             1512 non-null   float64
 43  os_Android 6.0.1           1512 non-null   float64
 44  os_Android 7.0             1512 non-null   float64
 45  os_Android 7.0.1           1512 non-null   float64
 46  os_Android 7.1             1512 non-null   float64
 47  os_Android 7.1.1           1512 non-null   float64
 48  os_Android 7.1.2           1512 non-null   float64
 49  os_Android 8.0             1512 non-null   float64
 50  os_Android 8.0 Oreo        1512 non-null   float64
 51  os_Android 8.1             1512 non-null   float64
 52  os_Android 8.1 Oreo        1512 non-null   float64
 53  os_Android 9.0             1512 non-null   float64
 54  os_Android 9.0 Pie         1512 non-null   float64
 55  os_EMUI 12                 1512 non-null   float64
 56  os_EMUI 13                 1512 non-null   float64
 57  os_Tizen 3.0               1512 non-null   float64
 58  os_iOS 11                  1512 non-null   float64
 59  os_iOS 11.1.1              1512 non-null   float64
 60  os_iOS 12                  1512 non-null   float64
 61  os_iOS 13                  1512 non-null   float64
 62  os_iOS 14.1                1512 non-null   float64
 63  os_iOS 15                  1512 non-null   float64
 64  os_iOS 15.4                1512 non-null   float64
 65  os_iOS 16                  1512 non-null   float64
 66  battery_type_Li-Ion        1512 non-null   float64
 67  battery_type_Li-Po         1512 non-null   float64
dtypes: float64(52), int64(16)
memory usage: 803.4 KB
```

```
print('Признаки, имеющие максимальную по модулю корреляцию с ценой телефона')
best_params =
final_df.corr()['price(USD)'].map(abs).sort_values(ascending=False)[1:]
best_params = best_params[best_params.values > 0.3]
best_params
```

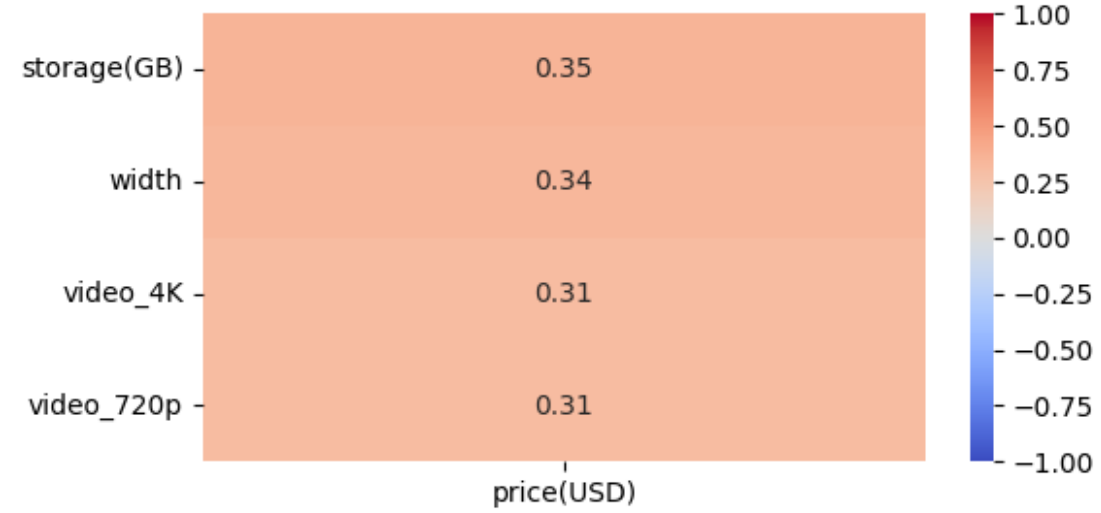
Признаки, имеющие максимальную по модулю корреляцию с ценой телефона

```
storage(GB)    0.354250
width          0.338867
video_4K       0.312411
video_720p     0.310810
Name: price(USD), dtype: float64
```

```
plt.figure(figsize=(14, 6))
sns.heatmap(final_df[best_params.index].corr(), vmin=-1, vmax=1,
cmap='coolwarm', annot=True)
plt.show()
```




```
plt.figure(figsize=(6, 3))
sns.heatmap(pd.DataFrame(data[np.append(best_params.index.values,
'price(USD)')].corr()['price(USD)'].sort_values(ascending=False)[1:]),
vmin=-1, vmax=1, cmap='coolwarm', annot=True)
plt.show()
```



Разделение выборки на обучающую и тестовую

```
y = data['price(USD)']
X = data[best_params.index]
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=3)
```

Линейная регрессия

```
def print_metrics(y_test, y_pred):
    # коэффициент детерминации
    print(f"R^2: {r2_score(y_test, y_pred)}")
    # среднеквадратичная ошибка
    print(f"MSE: {mean_squared_error(y_test, y_pred)}")
    # средняя абсолютная ошибка
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")

linear_model = LinearRegression()
linear_model.fit(x_train, y_train)
y_pred_linear = linear_model.predict(x_test)
print_metrics(y_test, y_pred_linear)
```

```
R^2: 0.13510555261347457
MSE: 66433.49914733712
MAE: 163.7848199812498
```

Полиноминальная регрессия

```
poly_model = PolynomialFeatures(degree=3)
x_train_poly = poly_model.fit_transform(x_train)
x_test_poly = poly_model.fit_transform(x_test)
linear_model = LinearRegression()
linear_model.fit(x_train_poly, y_train)
y_pred_poly = linear_model.predict(x_test_poly)
print_metrics(y_test, y_pred_poly)
```

```
R^2: -0.8520645814281242
MSE: 142259.12903350135
MAE: 179.1300123620549
```

SVM

```
scaler = StandardScaler().fit(x_train)
x_train_scaled = pd.DataFrame(scaler.transform(x_train),
columns=x_train.columns)
x_test_scaled = pd.DataFrame(scaler.transform(x_test),
columns=x_train.columns)
x_train_scaled.describe()
```


```
{"summary":"{\n  \"name\": \"x_train_scaled\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"storage(GB)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
373.8174690255228,\n        \"min\": -1.4947908459995323,\n
\"max\": 1058.0,\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          1058.0,\n          -6.296160820180472e-17,\n
0.25100927754626334\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"width\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 373.64275208677526,\n        \"min\": -2.1055893753998194,\n
\"max\": 1058.0,\n        \"num_unique_values\": 7,\n        \"samples\":
```

[\n   1058.0,\n   2.5184643280721886e-16,\n
0.15328536075849372\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"video_4K\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 373.97091200805585,\n   \"min\": -1.046446130935795,\n
\"max\": 1058.0,\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   -4.53323579052994e-17,\n   0.9556153636936283,\n
1.0004729250678182\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"video_720p\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 373.97376147434903,\n   \"min\": -1.058414134902744,\n
\"max\": 1058.0,\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   -1.6453966943404966e-16,\n   0.9448097554856335,\n
1.0004729250678182\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
params = {'C': np.concatenate([np.arange(0.1, 2, 0.1), np.arange(2, 15,
1)])}
svm_model = SVR(kernel='linear')
grid_cv = GridSearchCV(estimator=svm_model, param_grid=params, cv=10,
n_jobs=-1, scoring='r2')
grid_cv.fit(x_train_scaled, y_train)
print(grid_cv.best_params_)
```

```
{'C': 8.0}
```

```
best_svm_model = grid_cv.best_estimator_
best_svm_model = SVR(kernel='linear', C=8)
best_svm_model.fit(x_train_scaled, y_train)
y_pred_svm = best_svm_model.predict(x_test_scaled)
print_metrics(y_test, y_pred_svm)
```

```
R^2: 0.09028888194733287
MSE: 69875.91719208499
MAE: 145.91318982938506
```

Дерево решений

```
params = {'min_samples_leaf': range(3, 30)}
tree = DecisionTreeRegressor(random_state=3)
grid_cv = GridSearchCV(estimator=tree, cv=5, param_grid=params, n_jobs=-1,
scoring='neg_mean_absolute_error')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'min_samples_leaf': 13}
```

```
best_tree = grid_cv.best_estimator_
best_tree.fit(x_train, y_train)
y_pred_tree = best_tree.predict(x_test)
print_metrics(y_test, y_pred_tree)
```

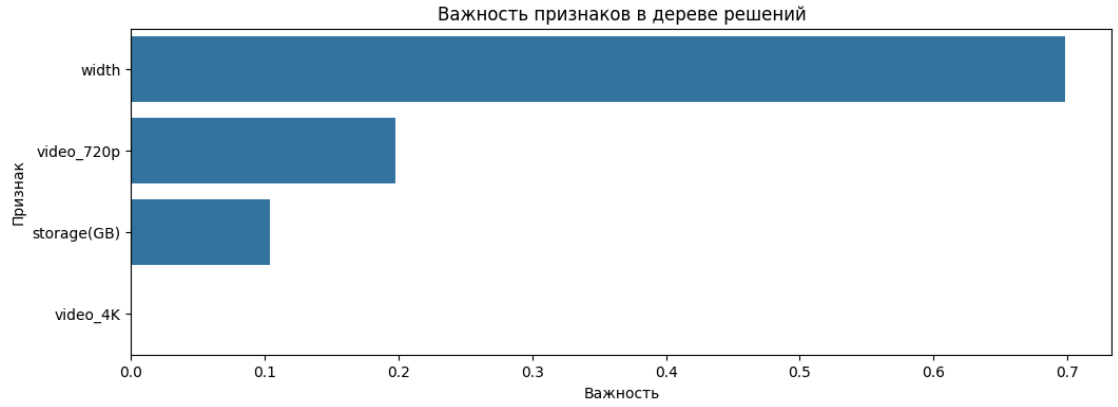
```
R^2: 0.17201798017858494
MSE: 63598.21475791498
MAE: 152.63921401566597
```

```
importances = pd.DataFrame(data=zip(x_train.columns,
best_tree.feature_importances_), columns=['Признак', 'Важность'])
print('Важность признаков в дереве решений\n')
for row in importances.sort_values(by='Важность', ascending=False).values:
    print(f'{row[0]}: {round(row[1], 3)}')
```

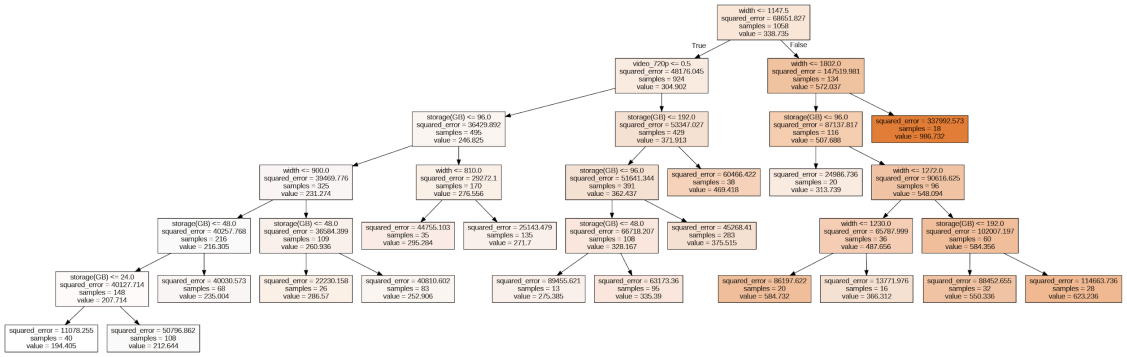
```
Важность признаков в дереве решений

width: 0.698
video_720p: 0.197
storage(GB): 0.104
video_4K: 0.0
```

```
plt.figure(figsize=(12, 4))
sns.barplot(data=importances.sort_values(by='Важность', ascending=False),
y='Признак', x='Важность', orient='h', )
plt.title('Важность признаков в дереве решений')
plt.show()
```




```
export_graphviz(best_tree, feature_names=best_params.index, filled=True,
out_file='tree.dot')
!dot -Tpng tree.dot -o tree.png
Image(filename='tree.png')
```



```
print('Линейная регрессия')
print_metrics(y_test, y_pred_linear)

print('\nПолиномиальная регрессия')
print_metrics(y_test, y_pred_poly)
```

```python
print('\nМетод опорных векторов')
print_metrics(y_test, y_pred_svm)

print('\nДерево решений')
print_metrics(y_test, y_pred_tree)
```

```
Линейная регрессия
R^2: 0.13510555261347457
MSE: 66433.49914733712
MAE: 163.7848199812498

Полиномиальная регрессия
R^2: -0.8520645814281242
MSE: 142259.12903350135
MAE: 179.1300123620549

Метод опорных векторов
R^2: 0.09028888194733287
MSE: 69875.91719208499
MAE: 145.91318982938506

Дерево решений
R^2: 0.17201798017858494
MSE: 63598.21475791498
MAE: 152.63921401566597
```