**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 ________________Системы обработки информации и управления______________

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ*

## *НА ТЕМУ:*

***Предсказание стоимости ноутбука на основе характеристик***

Студент __ИУ5-33М____ (Группа) ________________________ (Подпись, дата) _____Зорин А.А.____ (И.О.Фамилия)

Руководитель ________________________ (Подпись, дата) _____Гапанюк Ю.Е._ (И.О.Фамилия)

Консультант ________________________ (Подпись, дата) ___________________ (И.О.Фамилия)

*2021 г.*

# Введение

Последние всемирные тренды показывают все высшую популярность эксплуатации ноутбуков. За последнее время число купленных ноутбуков превышает количество проданных стационарных компьютеров. Этот факт невозможно было себе представить еще пару лет ранее, однако сегодняшние события вносят свои коррективы. Теперь мы следим за развитием и новинками ноутбуков — это уже стало входить в наш привычный образ жизни. В данной курсовой работе будут определены характеристики современных ноутбуков, которые влияют на итоговую стоимость. На основе датасета будет построена модель, которая предсказывает стоимость ноутбука на основе характеристик.

# Исследование

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso, LinearRegression
from sklearn.feature_selection import SelectFromModel
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
import scipy.stats as stats
from supervised.automl import AutoML

import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['figure.dpi'] = 300
import seaborn as sns
sns.set_palette('husl')

data = pd.read_csv('../datasets/laptop/laptop_price.csv', encoding =
"ISO-8859-1")
data = data.set_index('laptop_ID')
data.head()
```

```
          Company      Product   TypeName  Inches  \
laptop_ID
1           Apple  MacBook Pro  Ultrabook    13.3
2           Apple  Macbook Air  Ultrabook    13.3
3              HP       250 G6   Notebook    15.6
4           Apple  MacBook Pro  Ultrabook    15.4
5           Apple  MacBook Pro  Ultrabook    13.3

                             ScreenResolution
Cpu  \
laptop_ID

1          IPS Panel Retina Display 2560x1600        Intel Core i5
2.3GHz
2                                    1440x900        Intel Core i5
1.8GHz
3                           Full HD 1920x1080  Intel Core i5 7200U
2.5GHz
4          IPS Panel Retina Display 2880x1800        Intel Core i7
```

```
2.7GHz
5         IPS Panel Retina Display 2560x1600       Intel Core i5
3.1GHz

           Ram              Memory                           Gpu
OpsSys  \
laptop_ID

1          8GB           128GB SSD  Intel Iris Plus Graphics 640
macOS
2          8GB  128GB Flash Storage        Intel HD Graphics 6000
macOS
3          8GB           256GB SSD         Intel HD Graphics 620  No
OS
4         16GB           512GB SSD            AMD Radeon Pro 455
macOS
5          8GB           256GB SSD  Intel Iris Plus Graphics 650
macOS

          Weight  Price_euros
laptop_ID
1         1.37kg      1339.69
2         1.34kg       898.94
3         1.86kg       575.00
4         1.83kg      2537.45
5         1.37kg      1803.60
```

### Обработка нестандартных признаков

```
data["Ram"] = data["Ram"].str.replace('GB', '')
data["Weight"] = data["Weight"].str.replace('kg', '')
data['Memory'] = data['Memory'].astype(str).replace('\.0', '',
regex=True)
data["Memory"] = data["Memory"].str.replace('GB', '')
data["Memory"] = data["Memory"].str.replace('TB', '000')
new2 = data["Memory"].str.split("+", n = 1, expand = True)
data["first"]= new2[0]
data["first"]=data["first"].str.strip()
data["second"]= new2[1]
data["Layer1HDD"] = data["first"].apply(lambda x: 1 if "HDD" in x else
0)
data["Layer1SSD"] = data["first"].apply(lambda x: 1 if "SSD" in x else
0)
data["Layer1Hybrid"] = data["first"].apply(lambda x: 1 if "Hybrid" in
x else 0)
data["Layer1Flash_Storage"] = data["first"].apply(lambda x: 1 if
"Flash Storage" in x else 0)
data['first'] = data['first'].str.replace(r'\D', '')
data["second"].fillna("0", inplace = True)
data["Layer2HDD"] = data["second"].apply(lambda x: 1 if "HDD" in x
else 0)
```

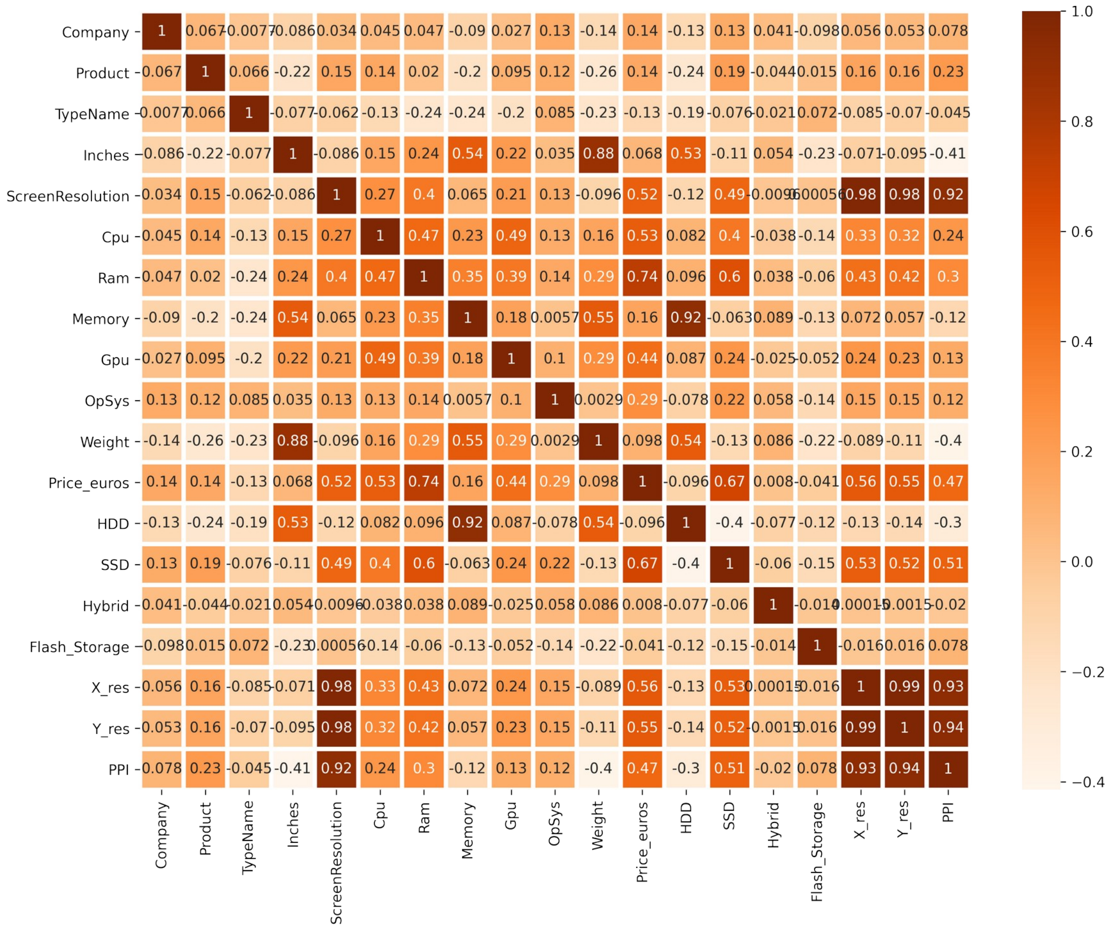
```
data["Layer2SSD"] = data["second"].apply(lambda x: 1 if "SSD" in x
else 0)
data["Layer2Hybrid"] = data["second"].apply(lambda x: 1 if "Hybrid" in
x else 0)
data["Layer2Flash_Storage"] = data["second"].apply(lambda x: 1 if
"Flash Storage" in x else 0)
data['second'] = data['second'].str.replace(r'\D', '')
data["first"] = data["first"].astype(int)
data["second"] = data["second"].astype(int)
data["Total_Memory"]=(data["first"]*(data["Layer1HDD"]
+data["Layer1SSD"]+data["Layer1Hybrid"]+data["Layer1Flash_Storage"])
+data["second"]*(data["Layer2HDD"]+data["Layer2SSD"]
+data["Layer2Hybrid"]+data["Layer2Flash_Storage"]))
data["Memory"]=data["Total_Memory"]
data["HDD"]=(data["first"]*data["Layer1HDD"]
+data["second"]*data["Layer2HDD"])
data["SSD"]=(data["first"]*data["Layer1SSD"]
+data["second"]*data["Layer2SSD"])
data["Hybrid"]=(data["first"]*data["Layer1Hybrid"]
+data["second"]*data["Layer2Hybrid"])
data["Flash_Storage"]=(data["first"]*data["Layer1Flash_Storage"]
+data["second"]*data["Layer2Flash_Storage"])
new = data["ScreenResolution"].str.split("x", n = 1, expand = True)
data["X_res"]= new[0]
data["Y_res"]= new[1]
data["Y_res"]= pd.to_numeric(data["Y_res"])
data["Y_res"]= data["Y_res"].astype(float)
data["X_res"]=(data['X_res'].str.replace(',','').str.findall(r'(\
d+\.?\d+)').apply(lambda x: pd.Series(x).astype(int)).mean(1))
data["X_res"]=pd.to_numeric(data["X_res"])
data["PPI"]=(((data["X_res"]**2+data["Y_res"]**2)**(1/2))/data["Inches
"]).astype(float)
data["ScreenResolution"]=(data["X_res"]*data["Y_res"]).astype(float)
data["Ram"] = data["Ram"].astype(int)
data["Weight"] = data["Weight"].astype(float)
data=data.drop(['first','second','Layer1HDD','Layer1SSD','Layer1Hybrid
','Layer1Flash_Storage','Layer2HDD','Layer2SSD','Layer2Hybrid','Layer2
Flash_Storage','Total_Memory'],axis=1)

data['Weight'], _ = stats.yeojohnson(data['Weight'])

cat_cols = ['Company', 'Product', 'TypeName', 'Cpu', 'Gpu', 'OpSys']
for col in cat_cols:
    data[col] = LabelEncoder().fit_transform(data[col])

data.head()
```

```
           Company  Product  TypeName  Inches  ScreenResolution  Cpu
Ram  \
laptop_ID
```

```
1                  1      300          4    13.3         4096000.0   65
8
2                  1      301          4    13.3         1296000.0   63
8
3                  7       50          3    15.6         2073600.0   74
8
4                  1      300          4    15.4         5184000.0   85
16
5                  1      300          4    13.3         4096000.0   67
8

           Memory  Gpu  OpSys    Weight  Price_euros  HDD  SSD  Hybrid
\
laptop_ID

1             128   58      8  0.706773      1339.69    0  128       0

2             128   51      8  0.698321       898.94    0    0       0

3             256   53      4  0.825642       575.00    0  256       0

4             512    9      8  0.819252      2537.45    0  512       0

5             256   59      8  0.706773      1803.60    0  256       0


           Flash_Storage   X_res   Y_res         PPI
laptop_ID
1                      0  2560.0  1600.0  226.983005
2                    128  1440.0   900.0  127.677940
3                      0  1920.0  1080.0  141.211998
4                      0  2880.0  1800.0  220.534624
5                      0  2560.0  1600.0  226.983005
```

```
plt.figure(figsize=(13,10))
sns.heatmap(data.corr(), cmap="Oranges", annot=True, linewidths=3)
```

```
<AxesSubplot:>
```




```python
def to_df(scaled, col_list):
    d = pd.DataFrame(scaled, columns=col_list)
    return d

def kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))

    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)

    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()

# Разделим выборку на обучающую и тестовую
X_train, X_test, y_train, y_test =
train_test_split(data.drop('Price_euros', axis=1),
data['Price_euros'],
                                                                    test_size=0.2,
                                                                    random_state=1)

# Преобразуем массивы в DataFrame
```

```
X_train_df = to_df(X_train, data.drop('Price_euros', axis=1).columns)
X_test_df = to_df(X_test, data.drop('Price_euros', axis=1).columns)

X_train_df.shape, X_test_df.shape
```

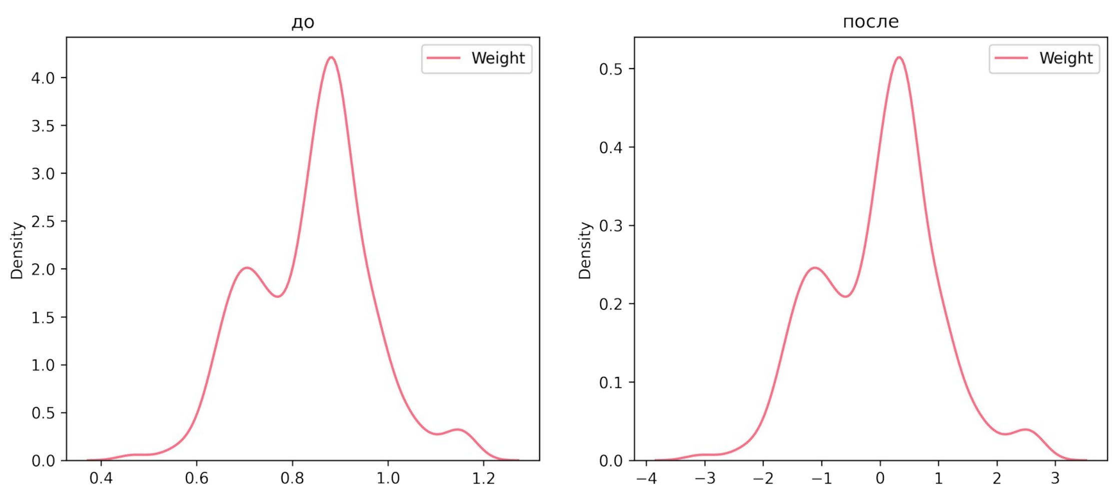
```
((1042, 18), (261, 18))
```

## Масштабирование признаков

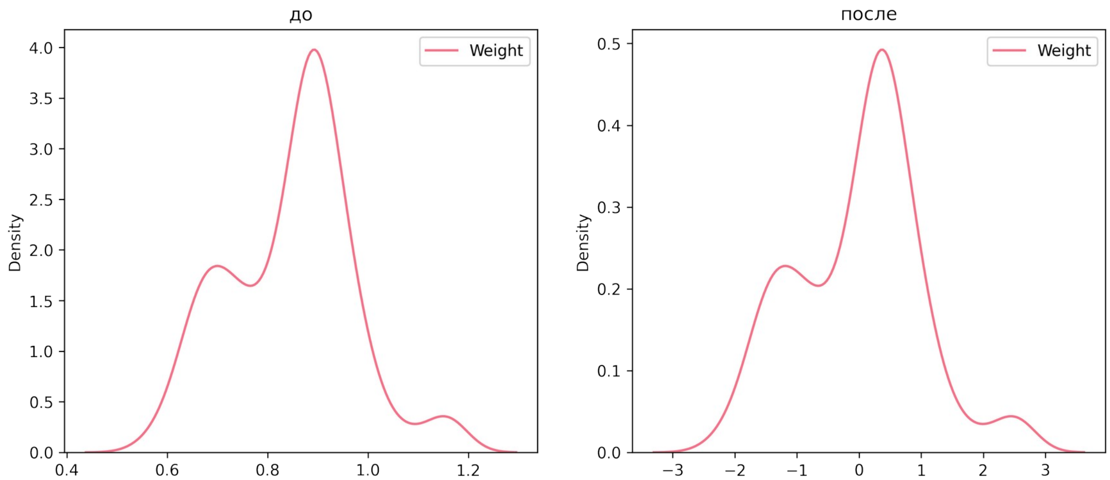
```
scale_cols = ["Weight"]
```

### 1 способ (Z-оценки)

```
# Обучаем StandardScaler на всей выборке и масштабируем
weight_standard_scaler =
StandardScaler().fit_transform(X_train[scale_cols])
# формируем DataFrame на основе массива
weight_standard_scaler =  to_df(weight_standard_scaler, scale_cols)
kde(scale_cols, X_train, weight_standard_scaler, 'до', 'после')
```

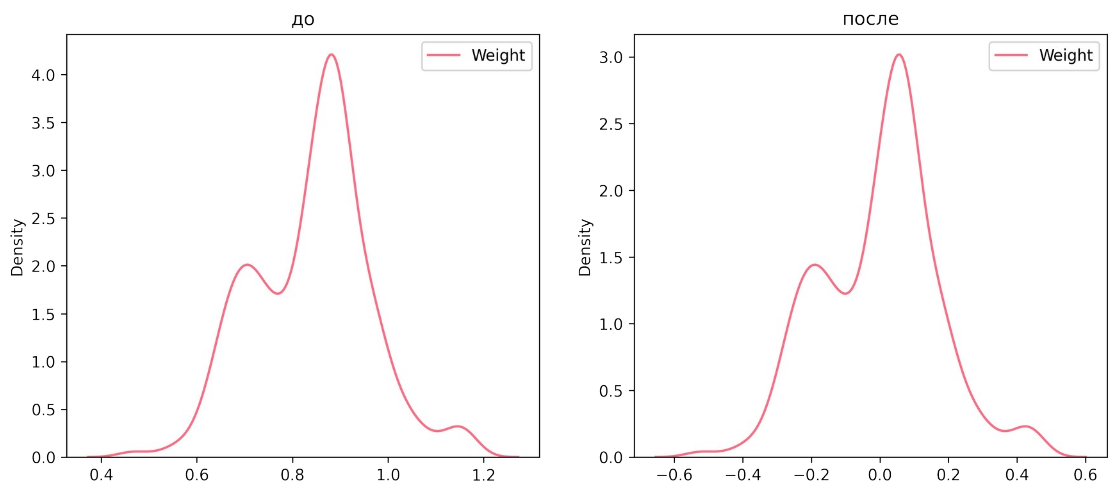



```
# Обучаем StandardScaler на всей выборке и масштабируем
weight_standard_scaler =
StandardScaler().fit_transform(X_test[scale_cols])
# формируем DataFrame на основе массива
weight_standard_scaler =  to_df(weight_standard_scaler, scale_cols)
kde(scale_cols, X_test, weight_standard_scaler, 'до', 'после')
```

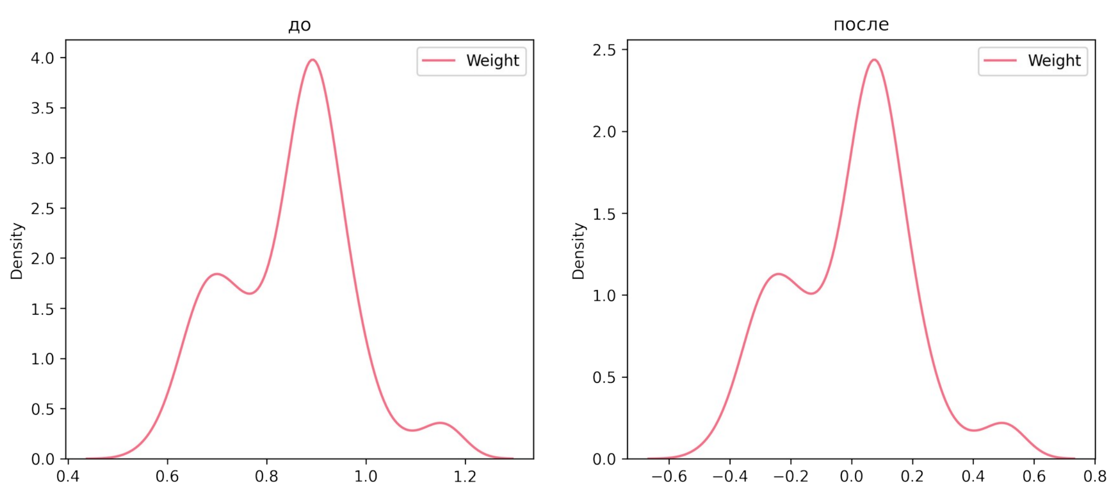



## 2 способ (Mean Normalisation)

```
class MeanNormalisation:

    def fit(self, param_df):
        self.means = param_df.mean(axis=0)
        maxs = param_df.max(axis=0)
        mins = param_df.min(axis=0)
        self.ranges = maxs - mins

    def transform(self, param_df):
        param_df_scaled = (param_df - self.means) / self.ranges
        return param_df_scaled

    def fit_transform(self, param_df):
        self.fit(param_df)
        return self.transform(param_df)

weight_standard_scaler =
MeanNormalisation().fit_transform(X_train[scale_cols])
weight_standard_scaler =  to_df(weight_standard_scaler, scale_cols)
kde(scale_cols, X_train, weight_standard_scaler, 'до', 'после')
```

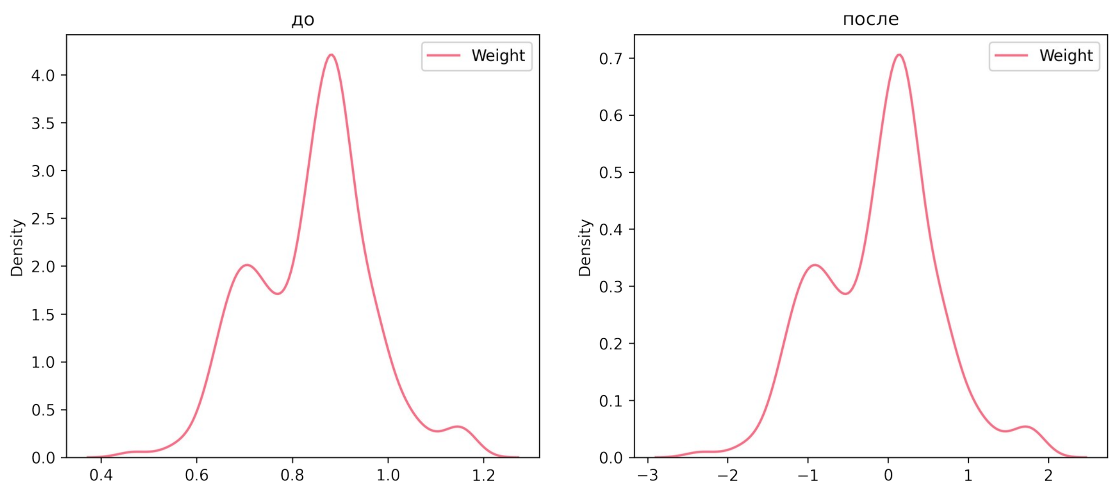



```
weight_standard_scaler =
MeanNormalisation().fit_transform(X_test[scale_cols])
weight_standard_scaler =  to_df(weight_standard_scaler, scale_cols)
kde(scale_cols, X_test, weight_standard_scaler, 'до', 'после')
```

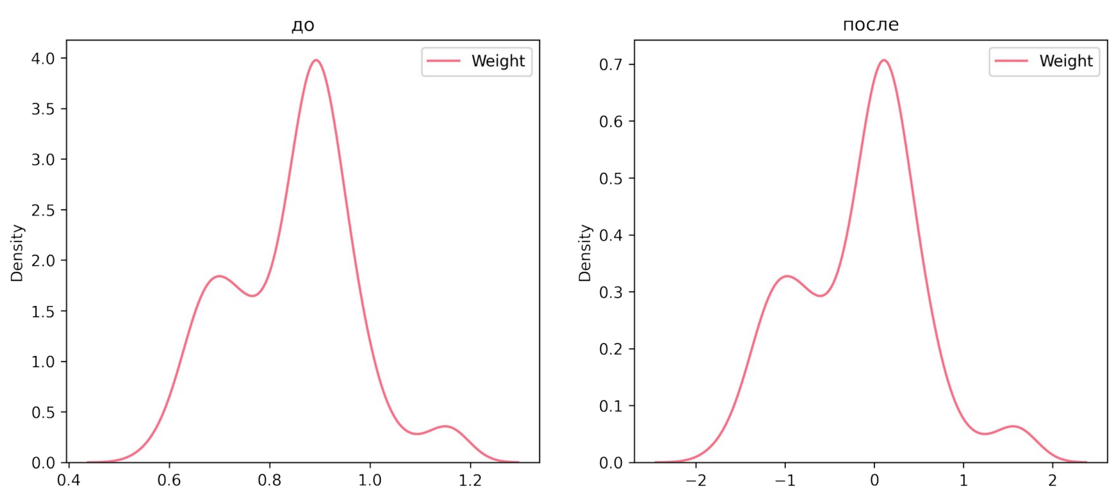



### 3 способ (по медиане)

```
weight_standard_scaler =
RobustScaler().fit_transform(X_train[scale_cols])
weight_standard_scaler =  to_df(weight_standard_scaler, scale_cols)
kde(scale_cols, X_train, weight_standard_scaler, 'до', 'после')
```




```
weight_standard_scaler =
RobustScaler().fit_transform(X_test[scale_cols])
weight_standard_scaler =  to_df(weight_standard_scaler, scale_cols)
kde(scale_cols, X_test, weight_standard_scaler, 'до', 'после')
```



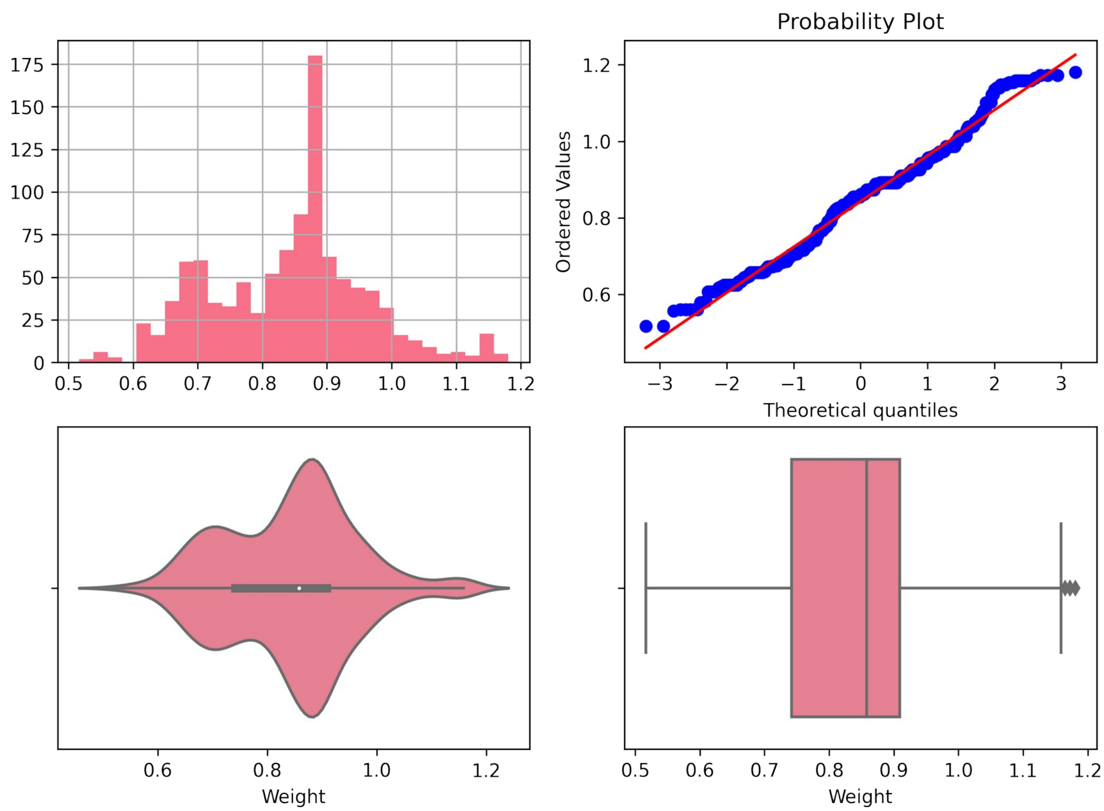

## Обработка выбросов для числовых признаков

### Удаление выбросов

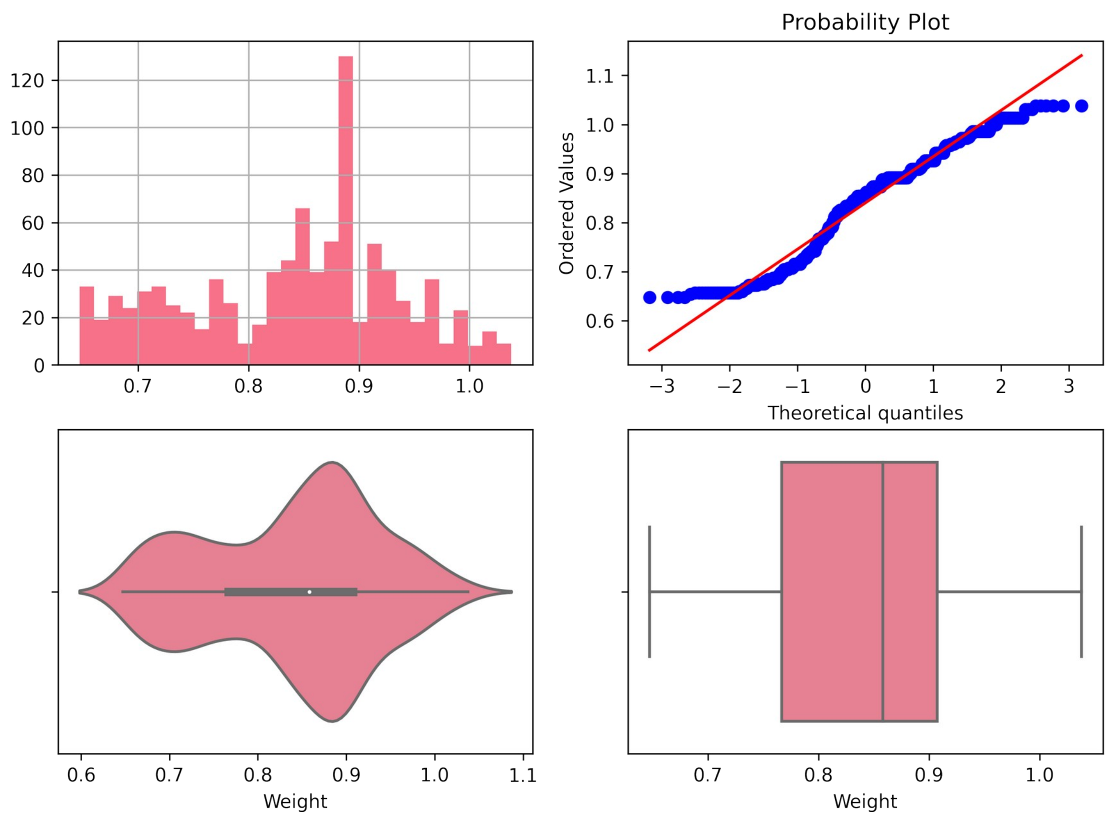
```
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3

def get_outlier_boundaries(df, col, outlier_boundary_type:
OutlierBoundaryType):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())
```

```
    elif outlier_boundary_type == OutlierBoundaryType.QUANTILE:
        lower_boundary = df[col].quantile(0.05)
        upper_boundary = df[col].quantile(0.95)

    elif outlier_boundary_type == OutlierBoundaryType.IRQ:
        K2 = 1.5
        IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
        lower_boundary = df[col].quantile(0.25) - (K2 * IQR)
        upper_boundary = df[col].quantile(0.75) + (K2 * IQR)

    else:
        raise NameError('Unknown Outlier Boundary Type')

    return lower_boundary, upper_boundary

def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # ящик с усами
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()

for col in scale_cols:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary =
get_outlier_boundaries(X_train, col, obt)
        # Флаги для удаления выбросов
        outliers_temp = np.where(X_train[col] > upper_boundary, True,
                                 np.where(X_train[col] <
lower_boundary, True, False))
        # Удаление данных на основе флага
        data_trimmed = X_train.loc[~(outliers_temp), ]
        title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt,
data_trimmed.shape[0])
        diagnostic_plots(data_trimmed, col, title)
```

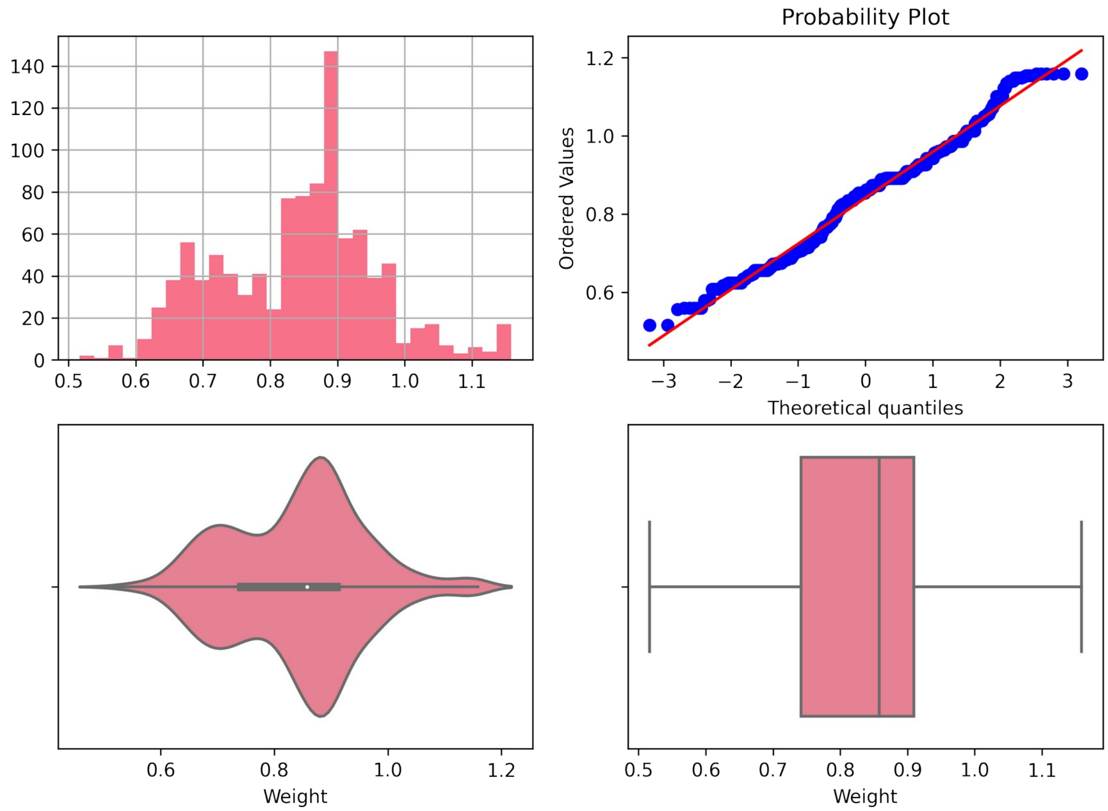
Поле-Weight, метод-OutlierBoundaryType.SIGMA, строк-1038



Поле-Weight, метод-OutlierBoundaryType.QUANTILE, строк-942



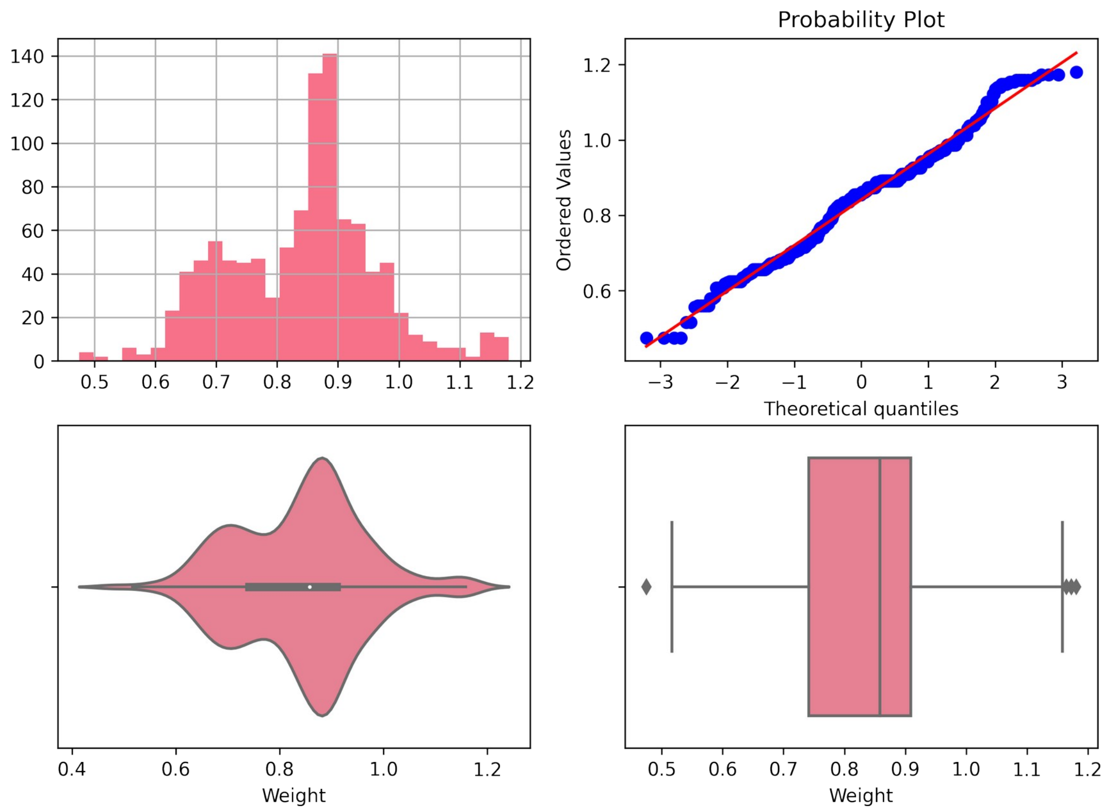
Поле-Weight, метод-OutlierBoundaryType.IRQ, строк-1033



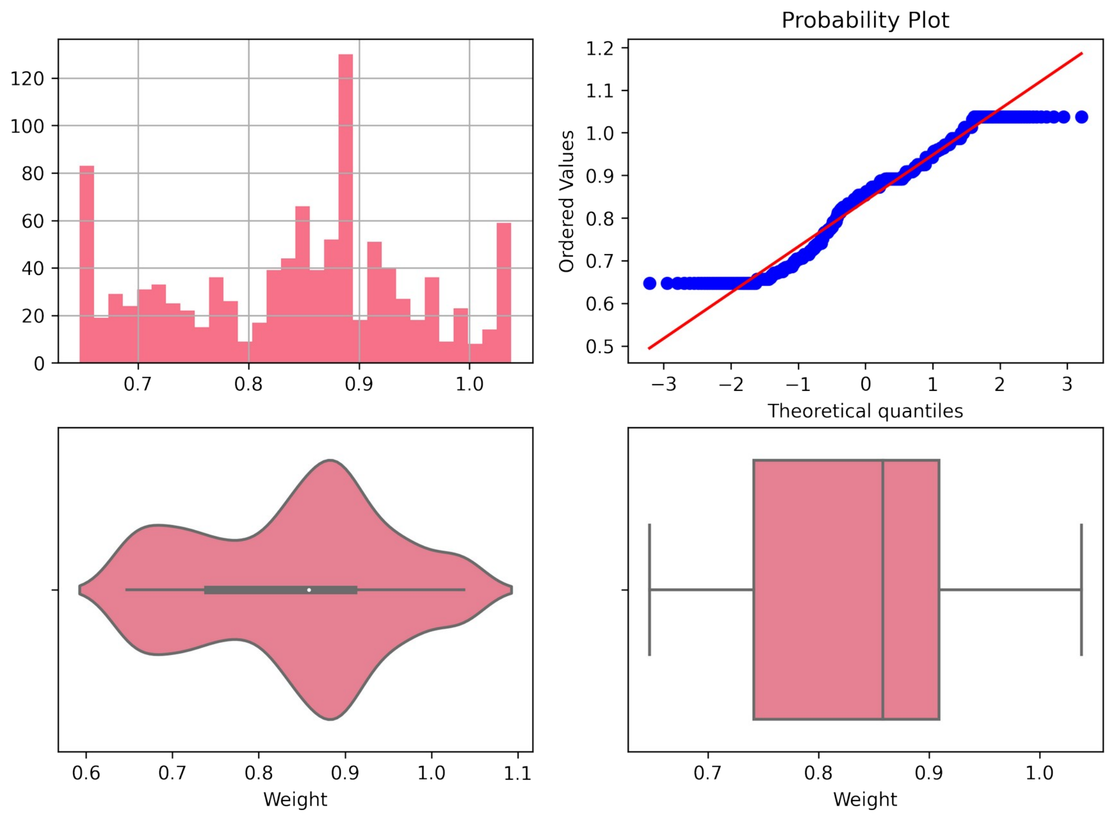

### Замена выбросов

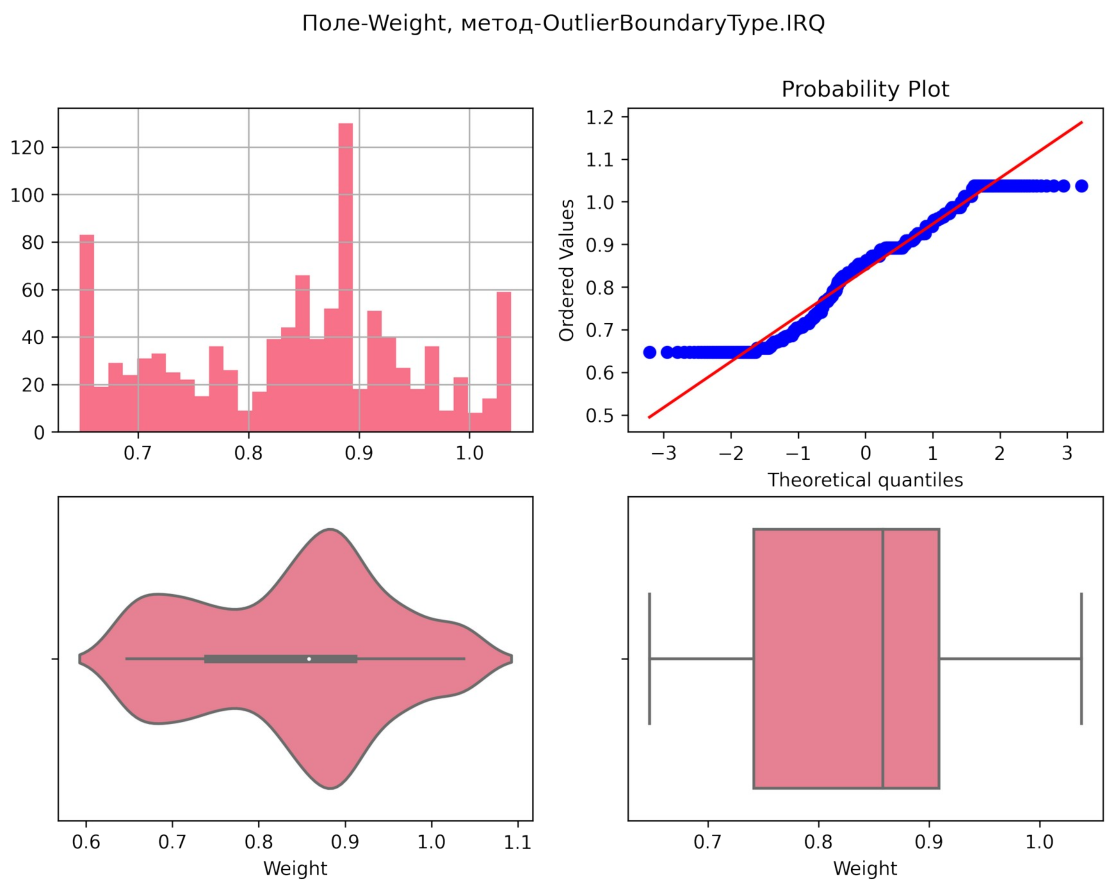
```
for col in scale_cols:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
        lower_boundary, upper_boundary = 
get_outlier_boundaries(X_train, col, obt)
        # Изменение данных
        X_train[col] = np.where(X_train[col] > upper_boundary, 
upper_boundary, 
                                 np.where(X_train[col] < 
lower_boundary, lower_boundary, X_train[col]))
        title = 'Поле-{}, метод-{}'.format(col, obt)
        diagnostic_plots(X_train, col, title)
```

Поле-Weight, метод-OutlierBoundaryType.SIGMA



Поле-Weight, метод-OutlierBoundaryType.QUANTILE






## Отбор признаков

### Метод из группы методов фильтрации

```python
# Формирование DataFrame с сильными корреляциями
def make_corr_df(df):
    cr = data.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.8]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr

make_corr_df(data)
```

| | f1 | f2 | corr |
|---|---|---|---|
| 0 | Y_res | X_res | 0.994219 |
| 1 | X_res | Y_res | 0.994219 |
| 2 | X_res | ScreenResolution | 0.982492 |
| 3 | ScreenResolution | X_res | 0.982492 |
| 4 | Y_res | ScreenResolution | 0.979294 |
| 5 | ScreenResolution | Y_res | 0.979294 |

```
6                 PPI              Y_res  0.939363
7               Y_res                PPI  0.939363
8                 PPI              X_res  0.931217
9               X_res                PPI  0.931217
10             Memory                HDD  0.920622
11                HDD             Memory  0.920622
12                PPI  ScreenResolution  0.920342
13   ScreenResolution                PPI  0.920342
14             Inches             Weight  0.879499
15             Weight             Inches  0.879499
```

```
# Обнаружение групп коррелирующих признаков
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            # находим коррелирующие признаки
            correlated_block = cr[cr['f1'] == feature]
            cur_dups = list(correlated_block['f2'].unique()) +
[feature]
            grouped_feature_list = grouped_feature_list + cur_dups
            correlated_groups.append(cur_dups)
    return correlated_groups

corr_groups(make_corr_df(data))
```

```
[['X_res', 'ScreenResolution', 'PPI', 'Y_res'],
 ['HDD', 'Memory'],
 ['Weight', 'Inches']]
```

**Метод из группы методов обертывания**

```
sfs = SFS(LinearRegression(),
          k_features=5,
          forward=True,
          floating=False,
          scoring = 'r2',
          cv = 0)

sfs.fit(X_train, y_train)
sfs.k_feature_names_
```

```
('Cpu', 'Ram', 'OpSys', 'SSD', 'Y_res')
```

**Метод из группы методов вложений**

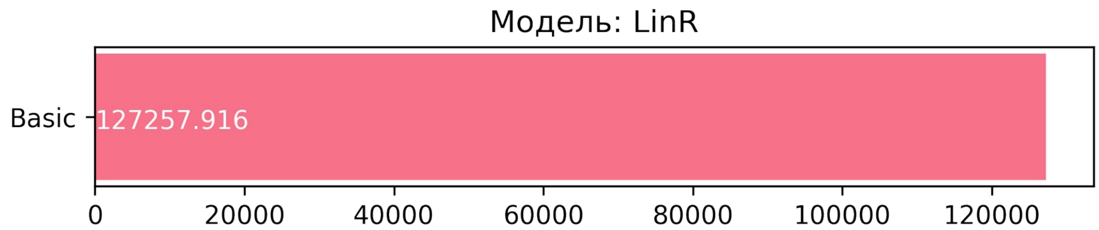
```
# Используем L1-регуляризацию
e_ls1 = Lasso(random_state=1)
e_ls1.fit(X_train, y_train)
# Коэффициенты регрессии
list(zip(data.drop('Price_euros', axis=1).columns, e_ls1.coef_))
```

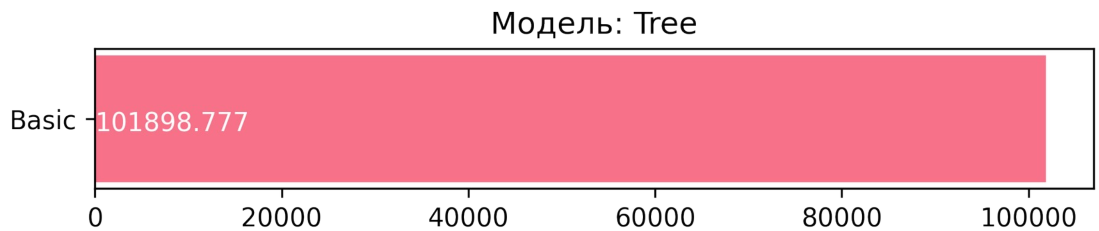
```
sel_e_ls1 = SelectFromModel(e_ls1)
sel_e_ls1.fit(X_train, y_train)
list(zip(data.drop('Price_euros', axis=1).columns,
sel_e_ls1.get_support()))
```

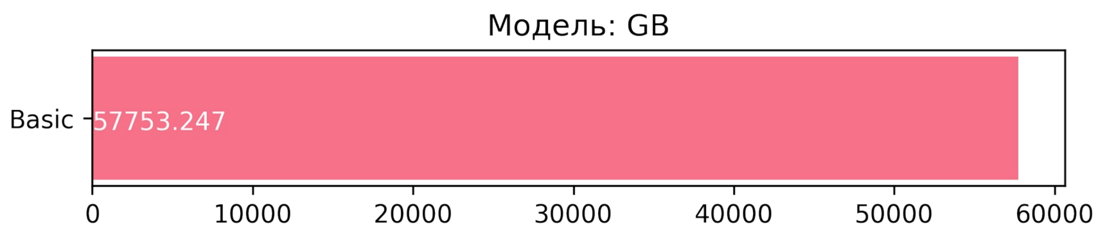
```
[('Company', True),
 ('Product', True),
 ('TypeName', True),
 ('Inches', True),
 ('ScreenResolution', True),
 ('Cpu', True),
 ('Ram', True),
 ('Memory', True),
 ('Gpu', True),
 ('OpSys', True),
 ('Weight', False),
 ('HDD', True),
 ('SSD', True),
 ('Hybrid', True),
 ('Flash_Storage', True),
 ('X_res', True),
 ('Y_res', True),
 ('PPI', True)]
```

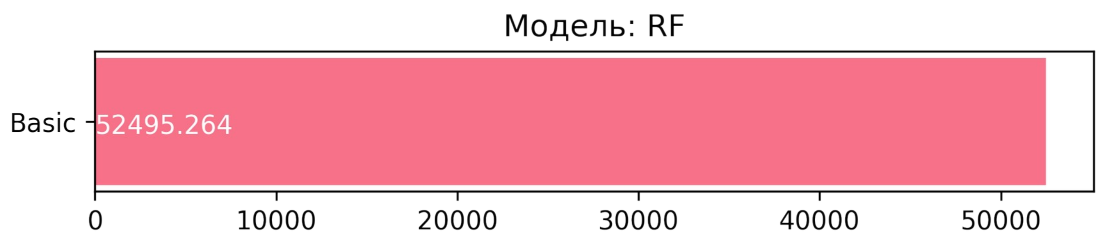
```
class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено

self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)
].index, inplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
```

```
        temp_data_2 = temp_data.sort_values(by='value',
ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5,
5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric,
ascending)
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        plt.show()

clas_models_dict = {'LinR': LinearRegression(),
                    'Tree':DecisionTreeRegressor(random_state=1),
                    'GB': GradientBoostingRegressor(random_state=1),
                    'RF':RandomForestRegressor(n_estimators=50,
random_state=1)}

X_data_dict = {'Basic': (X_train_df, X_test_df)}

def test_models(clas_models_dict, X_train, X_test, y_train, y_test):

    logger = MetricLogger()

    for model_name, model in clas_models_dict.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        logger.add(model_name, 'Basic', mse)

    return logger



logger = test_models(clas_models_dict, X_train_df, X_test_df, y_train,
y_test)

# Построим графики метрик качества модели
for model in clas_models_dict:
    logger.plot('Модель: ' + model, model, figsize=(7, 1))
```




### AutoML

```
!pip3 install --user mljar-supervised
!pip3 install delayed
```

```
Requirement already satisfied: mljar-supervised in
/usr/local/lib/python3.9/site-packages (0.10.4)
Requirement already satisfied: xgboost==1.3.3 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.3.3)
Requirement already satisfied: scikit-plot==0.3.7 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (0.3.7)
Requirement already satisfied: optuna==2.7.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (2.7.0)
Requirement already satisfied: tabulate==0.8.7 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (0.8.7)
Requirement already satisfied: cloudpickle==1.3.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.3.0)
Requirement already satisfied: category-encoders==2.2.2 in
```

```
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (2.2.2)
Requirement already satisfied: dtreeviz==1.3 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.3)
Requirement already satisfied: scikit-learn==0.24.2 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised)
(0.24.2)
Requirement already satisfied: seaborn==0.11.1 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised)
(0.11.1)
Requirement already satisfied: wordcloud==1.8.1 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.8.1)
Requirement already satisfied: joblib==1.0.1 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.0.1)
Requirement already satisfied: shap==0.36.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised)
(0.36.0)
Requirement already satisfied: markdown in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (3.3.4)
Requirement already satisfied: matplotlib>=3.2.2 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (3.4.2)
Requirement already satisfied: catboost==0.24.4 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised)
(0.24.4)
Requirement already satisfied: numpy>=1.20.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised)
(1.20.3)
Requirement already satisfied: pyarrow>=2.0.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (4.0.1)
Requirement already satisfied: scipy==1.6.1 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.6.1)
Requirement already satisfied: lightgbm==3.0.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (3.0.0)
Requirement already satisfied: pandas==1.2.0 in
/usr/local/lib/python3.9/site-packages (from mljar-supervised) (1.2.0)
Requirement already satisfied: six in
/usr/local/Cellar/protobuf/3.15.3/libexec/lib/python3.9/site-packages
(from catboost==0.24.4->mljar-supervised) (1.15.0)
Requirement already satisfied: graphviz in
/usr/local/lib/python3.9/site-packages (from catboost==0.24.4->mljar-
supervised) (0.16)
Requirement already satisfied: plotly in
/usr/local/lib/python3.9/site-packages (from catboost==0.24.4->mljar-
supervised) (4.14.3)
Requirement already satisfied: statsmodels>=0.9.0 in
/usr/local/lib/python3.9/site-packages (from category-encoders==2.2.2-
>mljar-supervised) (0.12.2)
Requirement already satisfied: patsy>=0.5.1 in
/usr/local/lib/python3.9/site-packages (from category-encoders==2.2.2-
>mljar-supervised) (0.5.1)
Requirement already satisfied: pytest in
```

```
/usr/local/lib/python3.9/site-packages (from dtreeviz==1.3->mljar-
supervised) (6.2.4)
Requirement already satisfied: colour in
/usr/local/lib/python3.9/site-packages (from dtreeviz==1.3->mljar-
supervised) (0.1.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.9/site-packages (from optuna==2.7.0->mljar-
supervised) (20.9)
Requirement already satisfied: cmaes>=0.8.2 in
/usr/local/lib/python3.9/site-packages (from optuna==2.7.0->mljar-
supervised) (0.8.2)
Requirement already satisfied: colorlog in
/usr/local/lib/python3.9/site-packages (from optuna==2.7.0->mljar-
supervised) (5.0.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/site-
packages (from optuna==2.7.0->mljar-supervised) (4.61.0)
Requirement already satisfied: alembic in
/usr/local/lib/python3.9/site-packages (from optuna==2.7.0->mljar-
supervised) (1.6.5)
Requirement already satisfied: sqlalchemy>=1.1.0 in
/usr/local/lib/python3.9/site-packages (from optuna==2.7.0->mljar-
supervised) (1.4.17)
Requirement already satisfied: cliff in /usr/local/lib/python3.9/site-
packages (from optuna==2.7.0->mljar-supervised) (3.8.0)
Requirement already satisfied: pytz>=2017.3 in
/usr/local/lib/python3.9/site-packages (from pandas==1.2.0->mljar-
supervised) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.9/site-packages (from pandas==1.2.0->mljar-
supervised) (2.8.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.9/site-packages (from scikit-learn==0.24.2-
>mljar-supervised) (2.1.0)
Requirement already satisfied: slicer in
/usr/local/lib/python3.9/site-packages (from shap==0.36.0->mljar-
supervised) (0.0.7)
Requirement already satisfied: numba in /usr/local/lib/python3.9/site-
packages (from shap==0.36.0->mljar-supervised) (0.53.1)
Requirement already satisfied: pillow in
/usr/local/lib/python3.9/site-packages (from wordcloud==1.8.1->mljar-
supervised) (8.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.9/site-packages (from matplotlib>=3.2.2->mljar-
supervised) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.9/site-packages (from matplotlib>=3.2.2->mljar-
supervised) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.9/site-packages (from matplotlib>=3.2.2->mljar-
supervised) (0.10.0)
```

```
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.9/site-packages (from sqlalchemy>=1.1.0-
>optuna==2.7.0->mljar-supervised) (1.1.0)
Requirement already satisfied: python-editor>=0.3 in
/usr/local/lib/python3.9/site-packages (from alembic->optuna==2.7.0-
>mljar-supervised) (1.0.4)
Requirement already satisfied: Mako in /usr/local/lib/python3.9/site-
packages (from alembic->optuna==2.7.0->mljar-supervised) (1.1.4)
Requirement already satisfied: cmd2>=1.0.0 in
/usr/local/lib/python3.9/site-packages (from cliff->optuna==2.7.0-
>mljar-supervised) (1.5.0)
Requirement already satisfied: pbr!=2.1.0,>=2.0.0 in
/usr/local/lib/python3.9/site-packages (from cliff->optuna==2.7.0-
>mljar-supervised) (5.6.0)
Requirement already satisfied: PrettyTable>=0.7.2 in
/usr/local/lib/python3.9/site-packages (from cliff->optuna==2.7.0-
>mljar-supervised) (2.1.0)
Requirement already satisfied: PyYAML>=3.12 in
/usr/local/lib/python3.9/site-packages (from cliff->optuna==2.7.0-
>mljar-supervised) (5.4.1)
Requirement already satisfied: stevedore>=2.0.1 in
/usr/local/lib/python3.9/site-packages (from cliff->optuna==2.7.0-
>mljar-supervised) (3.3.0)
Requirement already satisfied: wcwidth>=0.1.7 in
/usr/local/lib/python3.9/site-packages (from cmd2>=1.0.0->cliff-
>optuna==2.7.0->mljar-supervised) (0.2.5)
Requirement already satisfied: colorama>=0.3.7 in
/usr/local/lib/python3.9/site-packages (from cmd2>=1.0.0->cliff-
>optuna==2.7.0->mljar-supervised) (0.4.4)
Requirement already satisfied: attrs>=16.3.0 in
/usr/local/lib/python3.9/site-packages (from cmd2>=1.0.0->cliff-
>optuna==2.7.0->mljar-supervised) (21.2.0)
Requirement already satisfied: pyperclip>=1.6 in
/usr/local/lib/python3.9/site-packages (from cmd2>=1.0.0->cliff-
>optuna==2.7.0->mljar-supervised) (1.8.2)
Requirement already satisfied: MarkupSafe>=0.9.2 in
/usr/local/lib/python3.9/site-packages (from Mako->alembic-
>optuna==2.7.0->mljar-supervised) (2.0.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.9/site-packages (from numba->shap==0.36.0-
>mljar-supervised) (53.0.0)
Requirement already satisfied: llvmlite<0.37,>=0.36.0rc1 in
/usr/local/lib/python3.9/site-packages (from numba->shap==0.36.0-
>mljar-supervised) (0.36.0)
Requirement already satisfied: retrying>=1.3.3 in
/usr/local/lib/python3.9/site-packages (from plotly->catboost==0.24.4-
>mljar-supervised) (1.3.3)
Requirement already satisfied: pluggy<1.0.0a1,>=0.12 in
/usr/local/lib/python3.9/site-packages (from pytest->dtreeviz==1.3-
>mljar-supervised) (0.13.1)
```

```
Requirement already satisfied: iniconfig in
/usr/local/lib/python3.9/site-packages (from pytest->dtreeviz==1.3-
>mljar-supervised) (1.1.1)
Requirement already satisfied: py>=1.8.2 in
/usr/local/lib/python3.9/site-packages (from pytest->dtreeviz==1.3-
>mljar-supervised) (1.10.0)
Requirement already satisfied: toml in /usr/local/lib/python3.9/site-
packages (from pytest->dtreeviz==1.3->mljar-supervised) (0.10.2)

WARNING: You are using pip version 21.0.1; however, version 21.1.2 is
available.
You should consider upgrading via the
'/usr/local/opt/python@3.9/bin/python3.9 -m pip install --upgrade pip'
command.
Collecting delayed
  Downloading delayed-0.11.0b1-py2.py3-none-any.whl (19 kB)
Collecting hiredis
  Downloading hiredis-2.0.0-cp39-cp39-macosx_10_9_x86_64.whl (24 kB)
Collecting redis
  Downloading redis-3.5.3-py2.py3-none-any.whl (72 kB)
WARNING: You are using pip version 21.0.1; however, version 21.1.2 is
available.
You should consider upgrading via the
'/usr/local/opt/python@3.9/bin/python3.9 -m pip install --upgrade pip'
command.
```

```
train = data
train.head()
```

```
automl = AutoML()
automl.fit(train[train.columns[2:-3]], train['Price_euros'])
```

```
AutoML directory: AutoML_1
The task is regression with evaluation metric rmse
AutoML will use algorithms: ['Baseline', 'Linear', 'Decision Tree',
'Random Forest', 'Xgboost', 'Neural Network']
AutoML will ensemble availabe models
AutoML steps: ['simple_algorithms', 'default_algorithms', 'ensemble']
* Step simple_algorithms will try to check up to 3 models
1_Baseline rmse 757.134561 trained in 0.46 seconds
2_DecisionTree rmse 165.431313 trained in 40.07 seconds
3_Linear rmse 1808.228924 trained in 7.19 seconds
* Step default_algorithms will try to check up to 3 models
4_Default_Xgboost rmse 48.712868 trained in 5.95 seconds
5_Default_NeuralNetwork rmse 115.24174 trained in 0.92 seconds
6_Default_RandomForest rmse 104.533224 trained in 5.1 seconds
* Step ensemble will try to check up to 1 model
Ensemble rmse 37.127632 trained in 0.26 seconds
AutoML fit time: 95.75 seconds
AutoML best model: Ensemble
```

# Вывод

В данной курсовой работе была произведена подготовка и анализ характеристик, влияющих на стоимость современных ноутбуков. На основе полученных результатов была построена модель, позволяющая предсказывать стоимость ноутбуков в зависимости от их характеристик.