Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчёт по курсовой работе
По курсу «Методы машинного обучения»

«Классификация звёздных типов»

ИСПОЛНИТЕЛЬ:

Лосева Светлана Сергеевна
Группа ИУ5-34М
____

Решение задачи:

1. Поиск и выбор набора данных для построения модели машинного обучения. На основе выбранного набора данных строится модель для задачи классификации.
2. Для выбранного датасета решить следующие задачи:
   - устранение пропусков в данных;
   - кодирование категориальных признаков;
   - нормализацию числовых признаков;
   - масштабирование признаков;
   - обработку выбросов для числовых признаков;
   - обработку нестандартных признаков (которые не является числовым или категориальным);
   - отбор признаков, наиболее подходящих для построения модели;
3. Обучить модель и оценить метрики качества для двух выборок :
   - исходная выборка, которая содержит только минимальную предобработку данных, необходимую для построения модели (например, кодирование категориальных признаков).
   - улучшенная выборка, полученная в результате полной предобработки данных в пункте 2.
4. Построить модель с использованием произвольной библиотеки AutoML.
5. Сравнить метрики для трех полученных моделей.

```
!pip install numpy pandas scikit-surprise sklearn seaborn matplotlib automl mljar-supervis
```

```
    Using cached mljar-supervised-0.7.1.tar.gz (69 kB)
  Collecting scipy>=1.0.0
    Downloading scipy-1.4.1-cp37-cp37m-manylinux1_x86_64.whl (26.1 MB)
       |████████████████████████████████| 26.1 MB 115.1 MB/s
  Collecting scikit-learn
    Using cached scikit_learn-0.23.2-cp37-cp37m-manylinux1_x86_64.whl (6.8 MB)
  Collecting xgboost==1.2.0
    Using cached xgboost-1.2.0-py3-none-manylinux2010_x86_64.whl (148.9 MB)
  Collecting mljar-supervised
    Using cached mljar-supervised-0.7.0.tar.gz (69 kB)
    Using cached mljar-supervised-0.6.1.tar.gz (65 kB)
  Collecting scikit-learn
    Using cached scikit_learn-0.22.2-cp37-cp37m-manylinux1_x86_64.whl (7.1 MB)
  Collecting xgboost==1.0.2
    Using cached xgboost-1.0.2-py3-none-manylinux1_x86_64.whl (109.7 MB)
  Collecting mljar-supervised
    Using cached mljar-supervised-0.6.0.tar.gz (61 kB)
    Using cached mljar-supervised-0.5.5.tar.gz (58 kB)
    Using cached mljar-supervised-0.5.4.tar.gz (58 kB)
    Using cached mljar-supervised-0.5.3.tar.gz (57 kB)
    Using cached mljar-supervised-0.5.2.tar.gz (55 kB)
    Using cached mljar-supervised-0.5.1.tar.gz (55 kB)
    Using cached mljar-supervised-0.5.0.tar.gz (55 kB)
    Using cached mljar-supervised-0.4.1.tar.gz (52 kB)
    Using cached mljar-supervised-0.4.0.tar.gz (52 kB)
    Using cached mljar-supervised-0.3.5.tar.gz (43 kB)
    Using cached mljar-supervised-0.3.4.tar.gz (43 kB)
    Using cached mljar-supervised-0.3.3.tar.gz (43 kB)
    Using cached mljar-supervised-0.3.2.tar.gz (43 kB)
    Using cached mljar-supervised-0.3.1.tar.gz (43 kB)
    Using cached mljar-supervised-0.3.0.tar.gz (43 kB)
    Using cached mljar-supervised-0.2.8.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.7.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.6.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.5.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.4.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.3.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.2.tar.gz (37 kB)
    Using cached mljar-supervised-0.2.1.tar.gz (36 kB)
  WARNING: Discarding https://files.pythonhosted.org/packages/2d/af/f9471b6e5c9e4fb6
    Using cached mljar-supervised-0.2.0.tar.gz (36 kB)
  WARNING: Discarding https://files.pythonhosted.org/packages/4f/54/0905eff999200251
    Using cached mljar-supervised-0.1.7.tar.gz (25 kB)
  Collecting xgboost==0.80
    Using cached xgboost-0.80-py2.py3-none-manylinux1_x86_64.whl (15.8 MB)
  Collecting mljar-supervised
    Using cached mljar-supervised-0.1.6.tar.gz (25 kB)
    Using cached mljar-supervised-0.1.5.tar.gz (25 kB)
    Using cached mljar-supervised-0.1.4.tar.gz (25 kB)
    Using cached mljar-supervised-0.1.3.tar.gz (25 kB)
    Using cached mljar-supervised-0.1.2.tar.gz (24 kB)
    Using cached mljar-supervised-0.1.1.tar.gz (23 kB)
    Using cached mljar-supervised-0.1.0.tar.gz (21 kB)
  INFO: pip is looking at multiple versions of threadpoolctl to determine which vers
  Collecting threadpoolctl>=2.0.0
    Using cached threadpoolctl-3.0.0-py3-none-any.whl (14 kB)
```

```
!pip install --upgrade numpy
!pip install --upgrade pandas
!pip install --upgrade lightgbm
!pip install --upgrade scikit-learn
!pip install --upgrade scipy
!pip install --upgrade tabulate
```

```
    Found existing installation: numpy 1.19.5
    Uninstalling numpy-1.19.5:
      Successfully uninstalled numpy-1.19.5
ERROR: pip's dependency resolver does not currently take into account all the package
yellowbrick 1.3.post1 requires numpy<1.20,>=1.16.0, but you have numpy 1.21.4 which i
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompa
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.2.9 which
Successfully installed numpy-1.21.4
WARNING: The following packages were previously imported in this runtime:
  [numpy]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.1
```

```
!pip install mljar-supervised
```

```
Collecting stevedore>=2.0.1
  Downloading https://files.pythonhosted.org/packages/d4/49/b602307aeac3df3384ff1fcd
     |████████████████████████████████| 51kB 6.3MB/s
Requirement already satisfied: PrettyTable>=0.7.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: PyYAML>=3.12 in /usr/local/lib/python3.7/dist-packages
Collecting pbr!=2.1.0,>=2.0.0
  Downloading https://files.pythonhosted.org/packages/18/e0/1d4702dd81121d04a477c272
     |████████████████████████████████| 112kB 47.0MB/s
Collecting Mako
  Downloading https://files.pythonhosted.org/packages/f3/54/dbc07fbb20865d3b78fdb7cf
     |████████████████████████████████| 81kB 8.5MB/s
Collecting python-editor>=0.3
  Downloading https://files.pythonhosted.org/packages/c6/d3/201fc3abe391bbae6606e6f1
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in /u
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Collecting colorama>=0.3.7
  Downloading https://files.pythonhosted.org/packages/44/98/5b86278fbbf250d239ae0ecb
Collecting pyperclip>=1.6
  Downloading https://files.pythonhosted.org/packages/a7/2c/4c64579f847bd5d539803c8b
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.7/dist-pac
Building wheels for collected packages: mljar-supervised, dtreeviz, shap, pyperclip
  Building wheel for mljar-supervised (setup.py) ... done
  Created wheel for mljar-supervised: filename=mljar_supervised-0.10.4-cp37-none-any
  Stored in directory: /root/.cache/pip/wheels/a3/ea/35/583dcb9528d9a561e490f431abea
  Building wheel for dtreeviz (setup.py) ... done
  Created wheel for dtreeviz: filename=dtreeviz-1.3-cp37-none-any.whl size=66642 sha
  Stored in directory: /root/.cache/pip/wheels/60/36/b1/188ee35c677e48463f6482d580f8
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.36.0-cp37-cp37m-linux_x86_64.whl size=4576
  Stored in directory: /root/.cache/pip/wheels/fb/15/e1/8f61106790da27e0765aaa6e6645
  Building wheel for pyperclip (setup.py) ... done
  Created wheel for pyperclip: filename=pyperclip-1.8.2-cp37-none-any.whl size=11107
  Stored in directory: /root/.cache/pip/wheels/25/af/b8/3407109267803f4015e1ee2ff23b
Successfully built mljar-supervised dtreeviz shap pyperclip
ERROR: google-colab 1.0.0 has requirement pandas~=1.1.0; python_version >= "3.0", bu
ERROR: albumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll have i
Installing collected packages: pandas, scipy, xgboost, lightgbm, catboost, tabulate,
  Found existing installation: pandas 1.2.4
    Uninstalling pandas-1.2.4:
      Successfully uninstalled pandas-1.2.4
  Found existing installation: scipy 1.6.3
    Uninstalling scipy-1.6.3:
      Successfully uninstalled scipy-1.6.3
  Found existing installation: xgboost 0.90
    Uninstalling xgboost-0.90:
      Successfully uninstalled xgboost-0.90
  Found existing installation: lightgbm 3.2.1
    Uninstalling lightgbm-3.2.1:
      Successfully uninstalled lightgbm-3.2.1
  Found existing installation: tabulate 0.8.9
    Uninstalling tabulate-0.8.9:
      Successfully uninstalled tabulate-0.8.9
  Found existing installation: wordcloud 1.5.0
    Uninstalling wordcloud-1.5.0:
      Successfully uninstalled wordcloud-1.5.0
Successfully installed Mako-1.1.4 alembic-1.6.5 catboost-0.24.4 category-encoders-2.
```

```
from sklearn.model_selection import train_test_split
from supervised.automl import AutoML
```

```
pandas.util.testing is deprecated. Use the functions in the public API at pandas.tes
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MaxAbsScaler
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.impute import KNNImputer
import scipy.stats as stats
import sklearn
from sklearn.svm import SVR
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import Lasso
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
from sklearn.feature_selection import SelectKBest, SelectPercentile
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.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from IPython.display import Image
%matplotlib inline
sns.set(style="ticks")


def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    # первый график
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    # второй график
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()


def impute_column(dataset, column, strategy_param, fill_value_param=None):

    temp_data = dataset[[column]].values
    size = temp_data.shape[0]
    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                            fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data

def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive
```

```
df = pd.read_csv('/content/gdrive/My Drive/MMO/stars.csv')
df.head(15)
```

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spectr_class
0	3068	NaN	0.170	16.120	Red	M
1	3042	0.00050	NaN	16.600	Red	M
11	3129	0.01220	NaN	11.790	Red	M

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Temperature      240 non-null    int64  
 1   Otnosit_yarkost  237 non-null    float64
 2   Otnosit_radius   228 non-null    float64
 3   Abs_Velichina    237 non-null    float64
 4   Color            233 non-null    object 
 5   Spectr_class     240 non-null    object 
 6   Type             240 non-null    int64  
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

Устранение пропусков

Методы заполнения медианой и заполнения наиболее распространенным значением категории

Устранение пропусков с использованием метода заполнения медианой

```
median_oy = df['Otnosit_yarkost'].median()
median_or = df['Otnosit_radius'].median()
median_av = df['Abs_Velichina'].median()

df['Otnosit_yarkost'] = df['Otnosit_yarkost'].fillna(median_oy)
df['Otnosit_radius'] = df['Otnosit_radius'].fillna(median_or)
df['Abs_Velichina'] = df['Abs_Velichina'].fillna(median_av)
```

Устранение пропусков с использованием метода заполнения наиболее распространенным значением категории

```
Color_new, _, _ = impute_column(df, 'Color', 'most_frequent')
df['Color'] = Color_new
```

```
df.head(15)
```

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spectr_class
0	3068	0.15300	0.170	16.120	Red	M
1	3042	0.00050	0.945	16.600	Red	M
2	2600	0.00030	0.102	6.228	Red	M
3	2800	0.00020	0.945	16.650	Red	M
4	1939	0.15300	0.103	20.060	Red	M
5	2840	0.00065	0.110	16.980	Red	M
6	2637	0.00073	0.127	17.220	Red	M
7	2600	0.00040	0.945	17.400	Red	M
8	2650	0.00069	0.110	17.450	Red	M
9	2700	0.00018	0.945	16.050	Red	M
10	3600	0.00290	0.945	10.690	Red	M
11	3129	0.01220	0.945	11.790	Red	M
12	3134	0.00040	0.196	13.210	Red	M

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Temperature      240 non-null    int64  
 1   Otnosit_yarkost  240 non-null    float64
 2   Otnosit_radius   240 non-null    float64
 3   Abs_Velichina    240 non-null    float64
 4   Color            240 non-null    object 
 5   Spectr_class     240 non-null    object 
 6   Type             240 non-null    int64  
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

Кодирование категориальных признаков

```
df['Spectr_class'].unique()
```

```
array(['M', 'B', 'A', 'F', 'O', 'K', 'G'], dtype=object)
```

```
df.loc[df['Spectr_class'] == 'M', 'Spectr_class'] = 1
df.loc[df['Spectr_class'] == 'B', 'Spectr_class'] = 2
df.loc[df['Spectr_class'] == 'A', 'Spectr_class'] = 3
df.loc[df['Spectr_class'] == 'F', 'Spectr_class'] = 4
df.loc[df['Spectr_class'] == 'O', 'Spectr_class'] = 5
df.loc[df['Spectr_class'] == 'K', 'Spectr_class'] = 6
df.loc[df['Spectr_class'] == 'G', 'Spectr_class'] = 7
df['Spectr_class'] = pd.to_numeric(df['Spectr_class'])
```

```
df['Spectr_class'].unique()
```

```
     array([1, 2, 3, 4, 5, 6, 7])
```

```
df['Color'].unique()
```

```
     array(['Red', 'Blue White', 'White', 'Yellowish White',
            'Pale yellow orange', 'Blue', 'Blue-white', 'yellow-white',
            'Whitish', 'Orange', 'White-Yellow', 'white', 'yellowish',
            'Yellowish', 'Orange-Red', 'Blue white', 'Blue-White'],
           dtype=object)
```

```
df['Color'] = df['Color'].replace(['Blue-white', 'Blue white', 'Blue-White'], 'Blue White'
df['Color'] = df['Color'].replace(['Yellowish White', 'yellow-white', 'White-Yellow'], 'Ye
df['Color'] = df['Color'].replace(['white', 'Whitish'], 'White')
df['Color'] = df['Color'].replace(['yellowish'], 'Yellowish')
df['Color'] = df['Color'].replace(['Pale yellow orange', 'Orange', 'Orange-Red'], 'Yellow
```

```
df['Color'].unique()
```

```
     array(['Red', 'Blue White', 'White', 'Yellow White', 'Yellow Red', 'Blue',
            'Yellowish'], dtype=object)
```

```
df.loc[df['Color'] == 'Red', 'Color'] = 1
df.loc[df['Color'] == 'Blue White', 'Color'] = 2
df.loc[df['Color'] == 'White', 'Color'] = 3
df.loc[df['Color'] == 'Yellow White', 'Color'] = 4
df.loc[df['Color'] == 'Yellow Red', 'Color'] = 5
df.loc[df['Color'] == 'Blue', 'Color'] = 6
df.loc[df['Color'] == 'Yellowish', 'Color'] = 7
df['Color'] = pd.to_numeric(df['Color'])
```

```
df['Color'].unique()
```

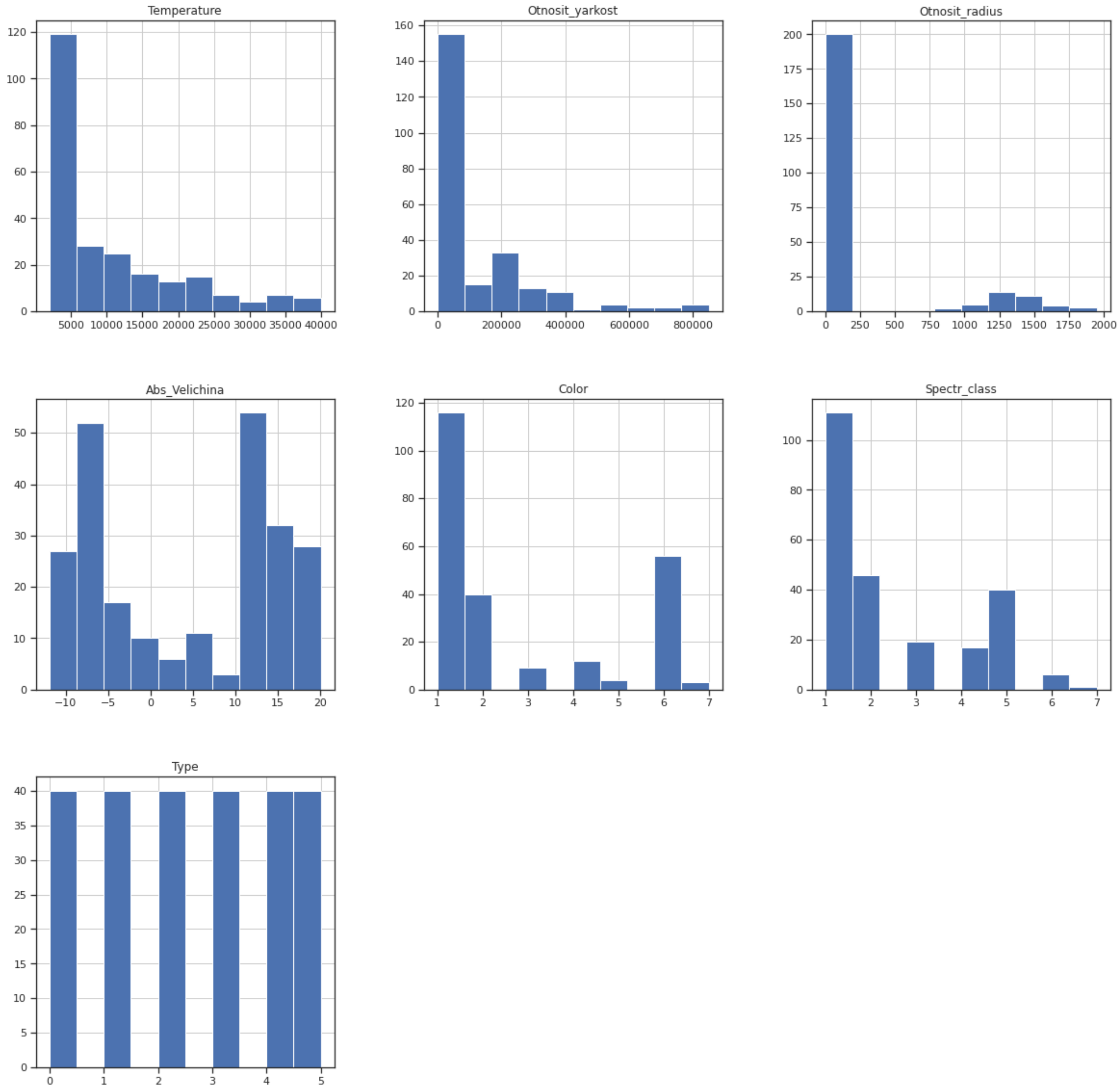
```
     array([1, 2, 3, 4, 5, 6, 7])
```

```
df.head(15)
```

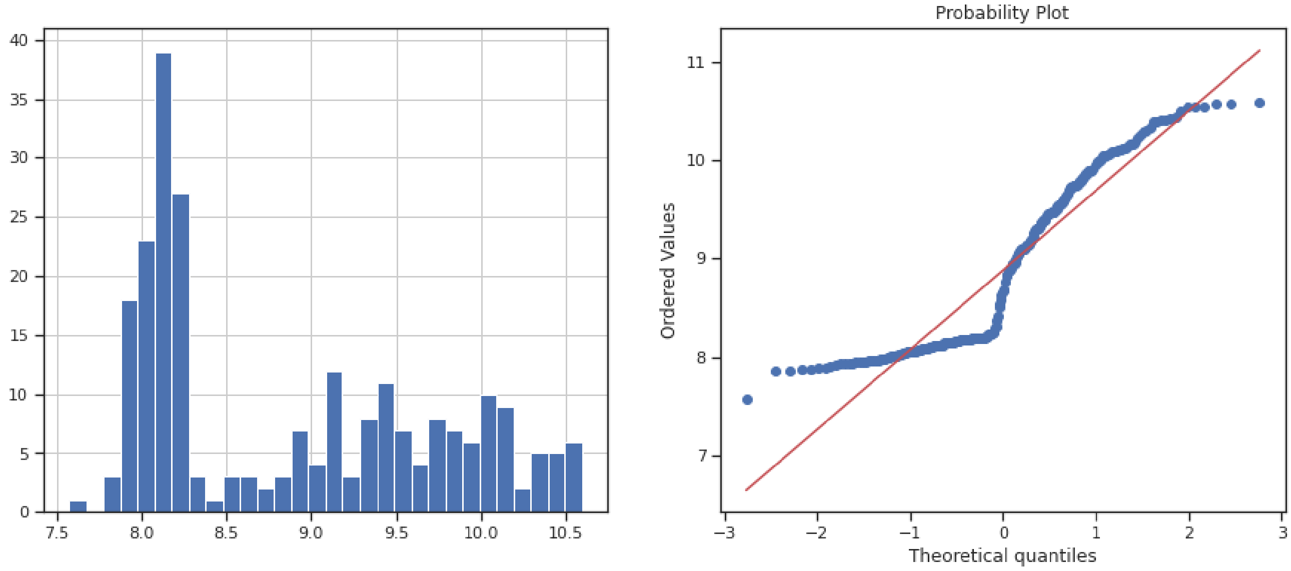
	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spectr_class
0	3068	0.15300	0.170	16.120	1	1
1	3042	0.00050	0.945	16.600	1	1
2	2600	0.00030	0.102	6.228	1	1
3	2800	0.00020	0.945	16.650	1	1
4	1939	0.15300	0.103	20.060	1	1
5	2840	0.00065	0.110	16.980	1	1
6	2637	0.00073	0.127	17.220	1	1
7	2600	0.00040	0.945	17.400	1	1
8	2650	0.00069	0.110	17.450	1	1
9	2700	0.00018	0.945	16.050	1	1
10	3600	0.00290	0.945	10.690	1	1

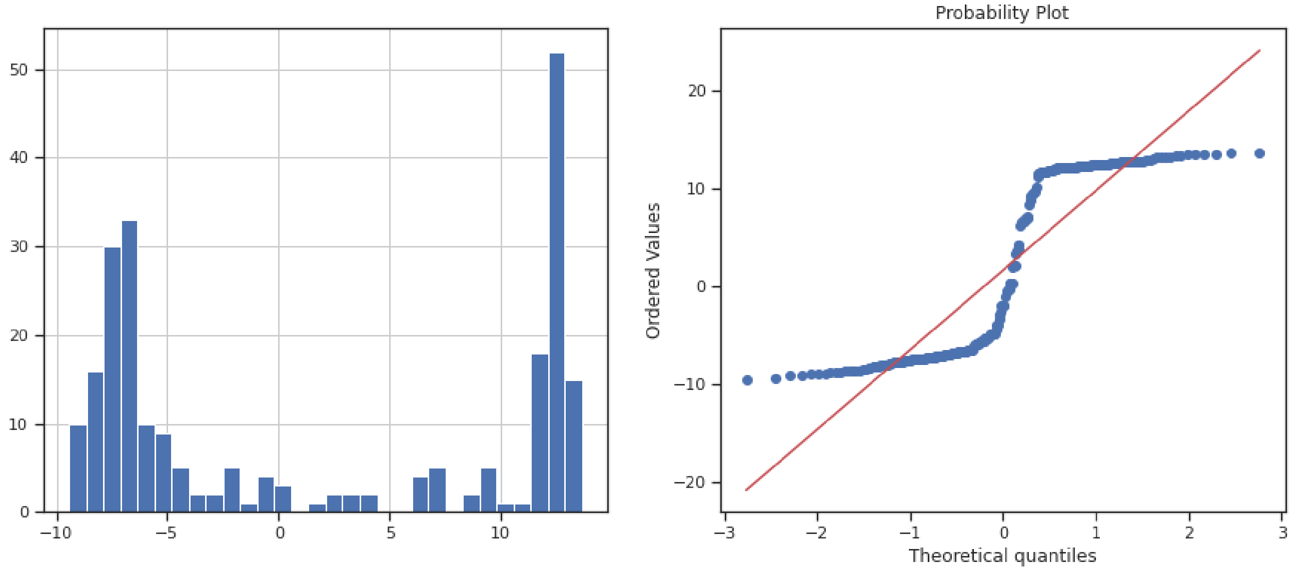
Нормализация числовых признаков

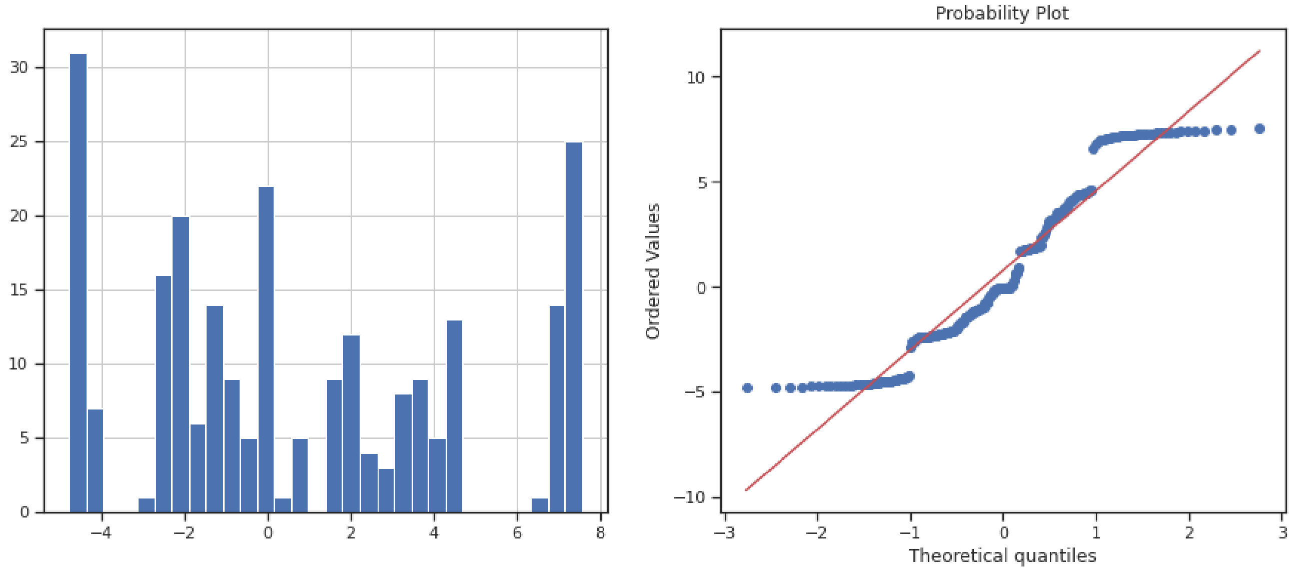
```
df_new = df.copy()
df_new.hist(figsize=(20,20))
plt.show()
```



```
df_new['Temperature'] = np.log(df_new['Temperature'])
df_new['Otnosit_yarkost'] = np.log(df_new['Otnosit_yarkost'])
df_new['Otnosit_radius'] = np.log(df_new['Otnosit_radius'])

diagnostic_plots(df_new, 'Temperature')
diagnostic_plots(df_new, 'Otnosit_yarkost')
diagnostic_plots(df_new, 'Otnosit_radius')
```







Масштабирование признаков

```
# Нужно ли масштабирование
df_new.describe()
```

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spect
count	240.000000	240.000000	240.000000	240.000000	240.000000	24
mean	8.880989	1.684978	0.790533	4.272204	2.700000	
std	0.857104	9.088714	3.892261	10.461630	2.108362	
min	7.569928	-9.433484	-4.779524	-11.920000	1.000000	
25%	8.114998	-7.038446	-2.207275	-6.232500	1.000000	
50%	8.661458	-1.877317	-0.056570	6.228000	2.000000	
75%	9.619463	12.196275	3.754918	13.564250	5.000000	
max	10.596635	13.652309	7.574815	20.060000	7.000000	

```
# DataFrame не содержащий целевой признак. Здесь введём и далее будем использовать новую в
# для дальнейшего сравнения обучения моделей
df_ne_cel = df_new.drop('Type', axis=1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-a29967b4704c> in <module>()
      1 # DataFrame не содержащий целевой признак. Здесь введём и далее будем
использовать новую выборку
      2 # для дальнейшего сравнения обучения моделей
----> 3 df_ne_cel = df_new.drop('Type', axis=1)

NameError: name 'df_new' is not defined
```

SEARCH STACK OVERFLOW

```
# Функция для восстановления датафрейма на основе масштабированных данных
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=df_ne_cel.columns)
    return res
```

```
# Деление выборки на обучающую и тестовую
X_train, X_test, y_train, y_test = train_test_split(
    df_ne_cel, df_new['Type'], test_size= 0.2, random_state= 1)
```

```
# Размер обучающей выборки
X_train.shape, y_train.shape
```

```
((192, 6), (192,))
```

```
# Размер тестовой выборки
X_test.shape, y_test.shape
```

```
    ((48, 6), (48,))
```

```
# Преобразуем массивы в DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

```
    ((192, 6), (48, 6))
```

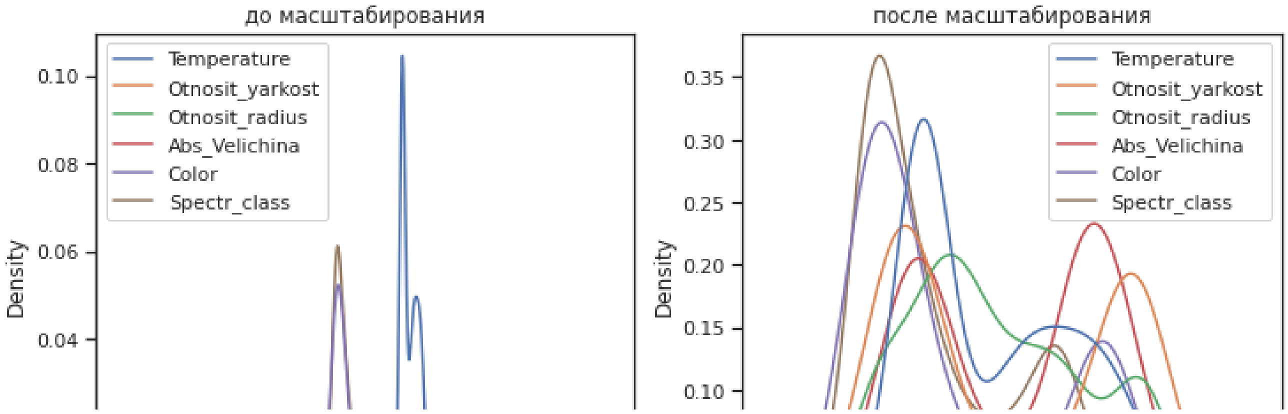
Min-Max Масштабирование

```
# Обучаем StandardScaler на всей выборке и масштабируем
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(df_ne_cel)
# формируем DataFrame на основе массива
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

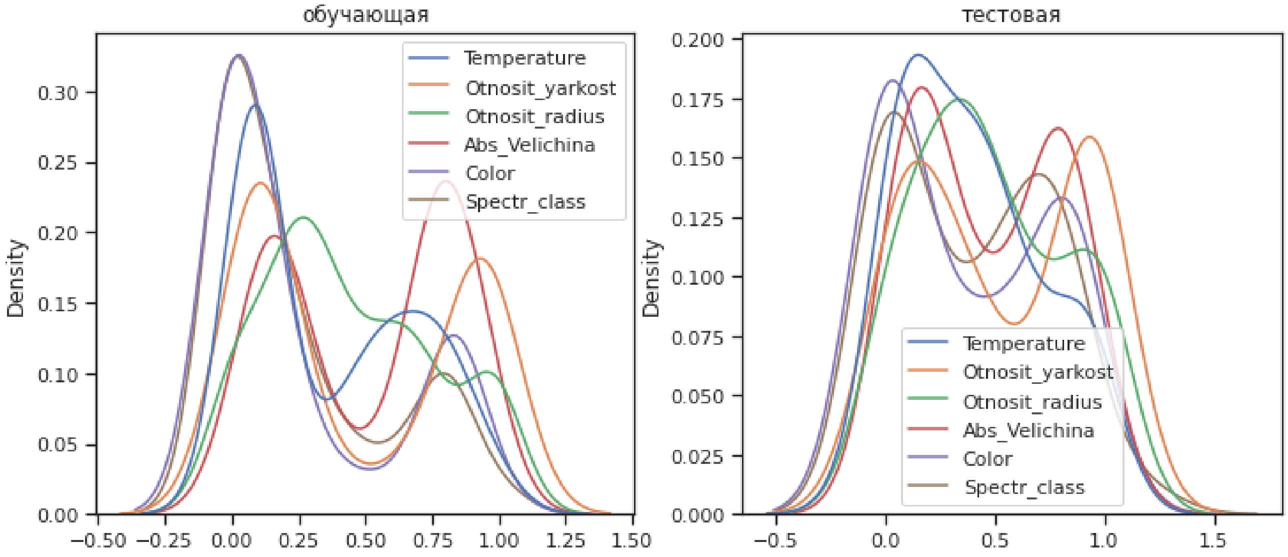
	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spect
count	240.000000	240.000000	240.000000	240.000000	240.000000	24
mean	0.433164	0.481615	0.450858	0.506323	0.283333	
std	0.283180	0.393693	0.315052	0.327130	0.351394	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.180087	0.103745	0.208206	0.177846	0.000000	
50%	0.360633	0.327308	0.382291	0.567480	0.166667	
75%	0.677150	0.936929	0.690805	0.796881	0.666667	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

```
cs32 = MinMaxScaler()
cs32.fit(X_train)
data_cs32_scaled_train_temp = cs32.transform(X_train)
data_cs32_scaled_test_temp = cs32.transform(X_test)
# формируем DataFrame на основе массива
data_cs32_scaled_train = arr_to_df(data_cs32_scaled_train_temp)
data_cs32_scaled_test = arr_to_df(data_cs32_scaled_test_temp)
```

```
draw_kde(['Temperature', 'Otnosit_yarkost', 'Otnosit_radius', 'Abs_Velichina', 'Color', 'S
```



```
draw_kde(['Temperature', 'Otnosit_yarkost', 'Otnosit_radius', 'Abs_Velichina', 'Color', 'S
```

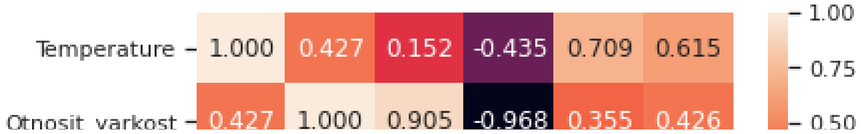


Отбор признаков

Filter methods

```
sns.heatmap(df_ne_cel.corr(), annot=True, fmt='.3f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c12d6eb50>
```



```
def make_corr_df(df):
    cr = df.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.5]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
```

```
make_corr_df(df_ne_cel)
```

	f1	f2	corr
0	Otnosit_yarkost	Abs_Velichina	0.968208
1	Abs_Velichina	Otnosit_yarkost	0.968208
2	Otnosit_yarkost	Otnosit_radius	0.905371
3	Otnosit_radius	Otnosit_yarkost	0.905371
4	Otnosit_radius	Abs_Velichina	0.894831
5	Abs_Velichina	Otnosit_radius	0.894831
6	Spectr_class	Color	0.800388
7	Color	Spectr_class	0.800388
8	Temperature	Color	0.709379
9	Color	Temperature	0.709379
10	Temperature	Spectr_class	0.615497
11	Spectr_class	Temperature	0.615497

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
```

```
# Группы коррелирующих признаков
corr_groups(make_corr_df(df_ne_cel))
```

```
     [['Abs_Velichina', 'Otnosit_radius', 'Otnosit_yarkost'],
      ['Color', 'Temperature', 'Spectr_class']]
```

Обучение модели и оценка метрики

Для исходной выборки

```
# Разделим выборку на обучающую и тестовую
X_train_basic, X_test_basic, y_train_basic, y_test_basic = train_test_split(
    df_new, df_new['Type'], test_size=0.2, random_state=1)

# Преобразуем массивы в DataFrame
X_train_basic_df = arr_to_df(X_train_basic)
X_test_basic_df = arr_to_df(X_test_basic)

X_train_basic_df.shape, X_test_basic_df.shape
```

```
     ((192, 6), (48, 6))
```

Для улучшенной выборки

```
# Разделим выборку на обучающую и тестовую
X_train_upgrade, X_test_upgrade, y_train_upgrade, y_test_upgrade = train_test_split(
    df_ne_cel, df_new['Type'], test_size=0.2, random_state=1)

# Преобразуем массивы в DataFrame
X_train_upgrade_df_new = arr_to_df(X_train_upgrade)
X_test_upgrade_df_new = arr_to_df(X_test_upgrade)

X_train_upgrade_df_new.shape, X_test_upgrade_df_new.shape
```

```
     ((192, 6), (48, 6))
```

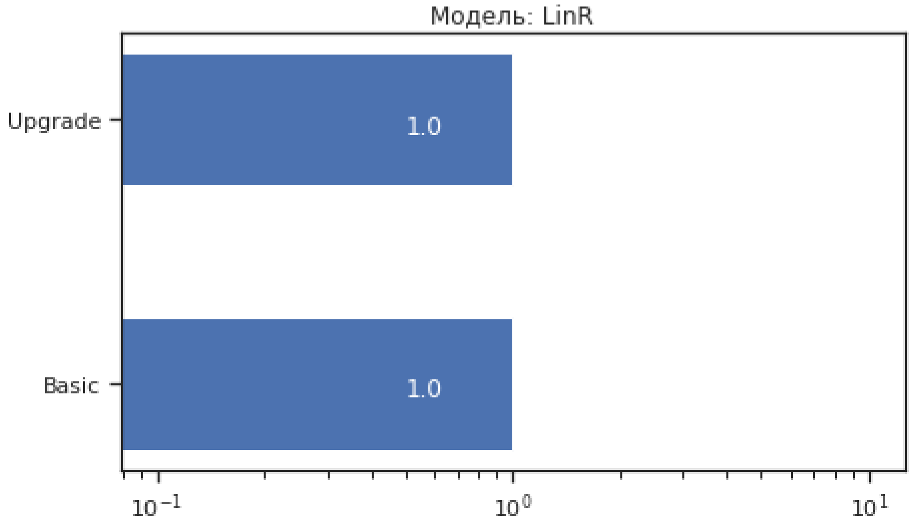
```
class MetricLogger:

    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, inp
        # 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.xscale('log')
        plt.show()


clas_models_dict = {'LinR': LogisticRegression(),
                    'KNN_5':KNeighborsClassifier(n_neighbors=5),
                    'Tree':DecisionTreeClassifier(random_state=1),
                    'GB': GradientBoostingClassifier(random_state=1),
                    'RF':RandomForestClassifier(n_estimators=20, random_state=1)}


X_data_dict = {'Basic': (X_train_basic_df, X_test_basic_df),
               'Upgrade': (X_train_upgrade_df_new, X_test_upgrade_df_new)}


from sklearn.metrics import f1_score, recall_score
def test_models(clas_models_dict, X_data_dict, y_train, y_test):

    logger = MetricLogger()

    for model_name, model in clas_models_dict.items():

        for data_name, data_tuple in X_data_dict.items():
            X_train, X_test = data_tuple

            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            # mse = mean_squared_error(y_test, y_pred)/
            f1_res = f1_score(y_test, y_pred, average='weighted')
            print(model, f1_res)
            logger.add(model_name, data_name, f1_res)

    return logger
```

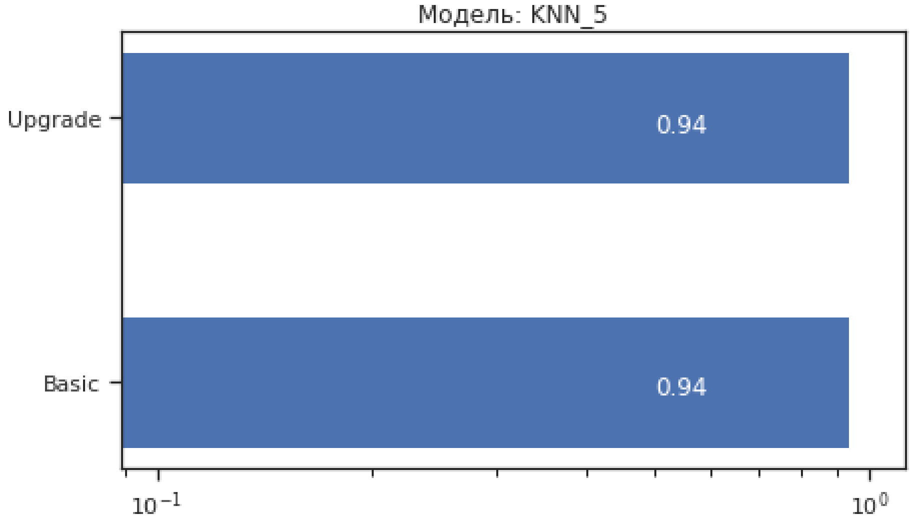
```
%%time
logger = test_models(clas_models_dict, X_data_dict, y_train_basic, y_test_basic)
```

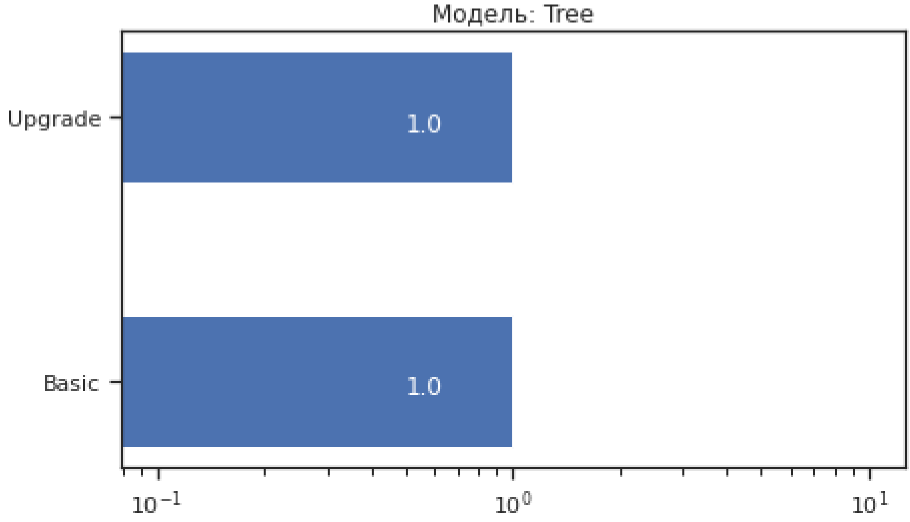
```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:765: Converg

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:765: Converg

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

LogisticRegression() 1.0
LogisticRegression() 1.0
KNeighborsClassifier() 0.9396739130434782
KNeighborsClassifier() 0.9396739130434782
DecisionTreeClassifier(random_state=1) 1.0
DecisionTreeClassifier(random_state=1) 1.0
GradientBoostingClassifier(random_state=1) 1.0
GradientBoostingClassifier(random_state=1) 1.0
RandomForestClassifier(n_estimators=20, random_state=1) 1.0
RandomForestClassifier(n_estimators=20, random_state=1) 1.0
CPU times: user 1.32 s, sys: 2.77 ms, total: 1.32 s
Wall time: 1.33 s
```

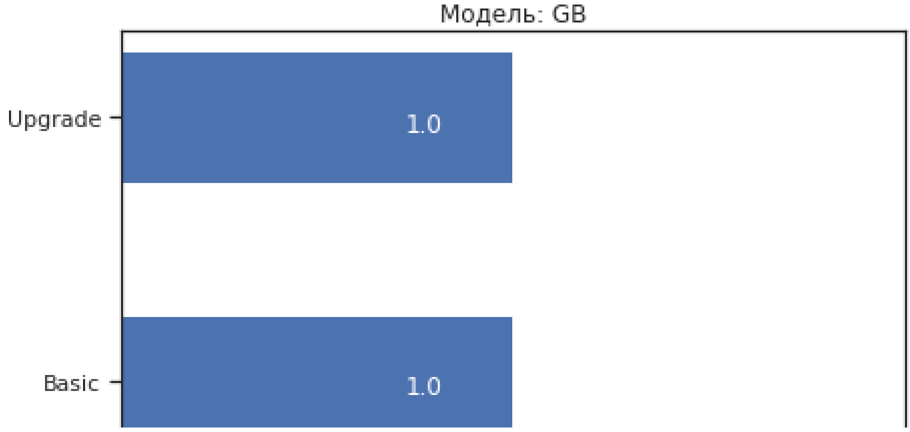
Построим графики метрик качества модели

```
for model in clas_models_dict:
    logger.plot('Модель: ' + model, model, figsize=(7, 4))
```


Модель: LinR
Upgrade
1.0
Basic
1.0
10^{-1}
10^{0}
10^{1}


Модель: KNN_5
Upgrade
0.94
Basic
0.94
10^{-1}
10^{0}


Модель: Tree
Upgrade
1.0
Basic
1.0
10^{-1}
10^{0}
10^{1}


Модель: GB
Upgrade
1.0
Basic
1.0