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

Лабораторная работа №2
По курсу «Методы машинного обучения»

«Обработка признаков. Часть 1»

ИСПОЛНИТЕЛЬ:

Лосева Светлана Сергеевна
Группа ИУ5-24М

_____

ПРОВЕРИЛ:

Гапанюк Ю.Е.

_____

Цель работы:

Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

1. Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных.
2. Для выбранного датасета на основе материалов лекций решить следующие задачи:
   - устранение пропусков в данных;
   - кодирование категориальных признаков;
   - нормализацию числовых признаков.

Описание задания:

Для выполнения лабораторной работы возьмём датасет, отображающий успеваемость студентов на экзаменах. Этот набор данных состоит из оценок, полученных студентами по различным предметам.

Выполнение работы:

1. Заполнение пропусков с использованием метода заполнения медианой для числовых признаков
2. Заполнение пропусков с использованием метода заполнения средним значением ля числовых признаков. Заполнение происходит с учётом группировки по параметру «Type», позволяя высчитывать среднее значение для каждой группы
3. Заполнение пропусков с использованием метода заполнения наиболее распространенным значением категории
4. Кодирование категориальных признаков
5. Нормализация числовых признаков. Для начала построим графики данных. По построенным графикам выберем для нормализации данные графика «Abs_Velichina» и используем функцию логарифма. По графику видно, что правая часть графика близка к нормальному распределению, тогда как левая - отклонение

Вывод:

Была проделана работа по обработке признаков для датасета по классификации звёздных типов

In [12]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from IPython.display import Image
%matplotlib inline
sns.set(style="ticks")
import scipy.stats as stats
```

In [13]:
```
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
```

In [14]:
```
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()
```

In [15]:
```
df = pd.read_csv('D:\\Ботва\\Магистратура\\2сем\\ММО\\ЛАБ2\\stars.csv')
df.head(30)
```

Out[15]:

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spectr_class	Type
0	3068	NaN	0.17000	16.120	Red	M	0
1	3042	0.000500	NaN	16.600	Red	M	0
2	2600	0.000300	0.10200	NaN	Red	M	0
3	2800	0.000200	NaN	16.650	Red	M	0

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Color	Spectr_class	Type
4	1939	NaN	0.10300	20.060	NaN	M	0
5	2840	0.000650	0.11000	16.980	Red	M	0
6	2637	0.000730	0.12700	17.220	Red	M	0
7	2600	0.000400	NaN	17.400	Red	M	0
8	2650	0.000690	0.11000	17.450	Red	M	0
9	2700	0.000180	NaN	16.050	Red	M	0
10	3600	0.002900	NaN	10.690	NaN	M	1
11	3129	0.012200	NaN	11.790	Red	M	1
12	3134	0.000400	0.19600	13.210	Red	M	1
13	3628	0.005500	NaN	10.480	Red	M	1
14	2650	0.000600	0.14000	11.782	NaN	M	1
15	3340	0.003800	0.24000	13.070	Red	M	1
16	2799	0.001800	NaN	14.790	Red	M	1
17	3692	0.003670	NaN	10.800	Red	M	1
18	3192	0.003620	0.19670	13.530	Red	M	1
19	3441	0.039000	0.35100	11.180	Red	M	1
20	25000	0.056000	0.00840	10.580	Blue White	B	2
21	7740	0.000490	NaN	14.020	NaN	A	2
22	7220	0.000170	0.01100	14.230	White	F	2
23	8500	0.000500	NaN	14.500	NaN	A	2
24	16500	NaN	0.01400	11.890	Blue White	B	2
25	12990	0.000085	0.00984	NaN	Yellowish White	F	2
26	8570	0.000810	0.00970	NaN	NaN	A	2
27	7700	0.000110	0.01280	14.470	Yellowish White	F	2
28	11790	0.000150	0.01100	12.590	Yellowish White	F	2
29	7230	0.000080	0.01300	14.080	Pale yellow orange	F	2

In [16]:
```
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
```

In [17]:
```
#Устранение пропусков с использованием метода заполнения медианой
df_chisl = df.select_dtypes(include=[np.number])
median_o = df_chisl['Otnosit_yarkost'].median()
df_chisl['Otnosit_yarkost'] = df['Otnosit_yarkost'].fillna(median_o)
```

```
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\ipykernel_launche
r.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

In [18]:
```
df_chisl.head(30)
```

Out[18]:

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Type
0	3068	0.153000	0.17000	16.120	0
1	3042	0.000500	NaN	16.600	0
2	2600	0.000300	0.10200	NaN	0
3	2800	0.000200	NaN	16.650	0
4	1939	0.153000	0.10300	20.060	0
5	2840	0.000650	0.11000	16.980	0
6	2637	0.000730	0.12700	17.220	0
7	2600	0.000400	NaN	17.400	0
8	2650	0.000690	0.11000	17.450	0
9	2700	0.000180	NaN	16.050	0
10	3600	0.002900	NaN	10.690	1
11	3129	0.012200	NaN	11.790	1
12	3134	0.000400	0.19600	13.210	1
13	3628	0.005500	NaN	10.480	1
14	2650	0.000600	0.14000	11.782	1
15	3340	0.003800	0.24000	13.070	1
16	2799	0.001800	NaN	14.790	1
17	3692	0.003670	NaN	10.800	1
18	3192	0.003620	0.19670	13.530	1
19	3441	0.039000	0.35100	11.180	1
20	25000	0.056000	0.00840	10.580	2
21	7740	0.000490	NaN	14.020	2
22	7220	0.000170	0.01100	14.230	2

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Type
23	8500	0.000500	NaN	14.500	2
24	16500	0.153000	0.01400	11.890	2
25	12990	0.000085	0.00984	NaN	2
26	8570	0.000810	0.00970	NaN	2
27	7700	0.000110	0.01280	14.470	2
28	11790	0.000150	0.01100	12.590	2
29	7230	0.000080	0.01300	14.080	2

In [19]:
```
#Устранение пропусков с использованием метода заполнения средним значением
df_sr = df.select_dtypes(include=[np.number])
name = 'Abs_Velichina'
df_sr.loc[df_sr[name].isnull(), name] = df_sr.groupby('Type')[name].transform('mean'
```

```
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\pandas\core\indexi
ng.py:1676: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

In [20]:
```
df_sr.head(30)
```

Out[20]:

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Type
0	3068	NaN	0.17000	16.120000	0
1	3042	0.000500	NaN	16.600000	0
2	2600	0.000300	0.10200	17.534359	0
3	2800	0.000200	NaN	16.650000	0
4	1939	NaN	0.10300	20.060000	0
5	2840	0.000650	0.11000	16.980000	0
6	2637	0.000730	0.12700	17.220000	0
7	2600	0.000400	NaN	17.400000	0
8	2650	0.000690	0.11000	17.450000	0
9	2700	0.000180	NaN	16.050000	0
10	3600	0.002900	NaN	10.690000	1
11	3129	0.012200	NaN	11.790000	1
12	3134	0.000400	0.19600	13.210000	1
13	3628	0.005500	NaN	10.480000	1
14	2650	0.000600	0.14000	11.782000	1
15	3340	0.003800	0.24000	13.070000	1
16	2799	0.001800	NaN	14.790000	1

	Temperature	Otnosit_yarkost	Otnosit_radius	Abs_Velichina	Type
17	3692	0.003670	NaN	10.800000	1
18	3192	0.003620	0.19670	13.530000	1
19	3441	0.039000	0.35100	11.180000	1
20	25000	0.056000	0.00840	10.580000	2
21	7740	0.000490	NaN	14.020000	2
22	7220	0.000170	0.01100	14.230000	2
23	8500	0.000500	NaN	14.500000	2
24	16500	NaN	0.01400	11.890000	2
25	12990	0.000085	0.00984	12.549211	2
26	8570	0.000810	0.00970	12.549211	2
27	7700	0.000110	0.01280	14.470000	2
28	11790	0.000150	0.01100	12.590000	2
29	7230	0.000080	0.01300	14.080000	2

In [21]:
```
#Устранение пропусков с использованием метода заполнения наиболее распространенным з
df_raspr = df[['Color', 'Spectr_class']].copy()
Color_new, _, _ = impute_column(df_raspr, 'Color', 'most_frequent')
df_raspr['Color'] = Color_new
```

In [22]:
```
df_raspr.head(30)
```

Out[22]:

	Color	Spectr_class
0	Red	M
1	Red	M
2	Red	M
3	Red	M
4	Red	M
5	Red	M
6	Red	M
7	Red	M
8	Red	M
9	Red	M
10	Red	M
11	Red	M
12	Red	M
13	Red	M
14	Red	M
15	Red	M

	Color	Spectr_class
16	Red	M
17	Red	M
18	Red	M
19	Red	M
20	Blue White	B
21	Red	A
22	White	F
23	Red	A
24	Blue White	B
25	Yellowish White	F
26	Red	A
27	Yellowish White	F
28	Yellowish White	F
29	Pale yellow orange	F

In [23]:
```
#Кодирование категориальных признаков
df_kod = df[['Color', 'Spectr_class']]
```

In [24]:
```
df_kod['Spectr_class'].unique()
```

Out[24]:
```
array(['M', 'B', 'A', 'F', 'O', 'K', 'G'], dtype=object)
```

In [25]:
```
df_kod.loc[df['Spectr_class'] == 'M', 'Spectr_class'] = 1
df_kod.loc[df['Spectr_class'] == 'B', 'Spectr_class'] = 2
df_kod.loc[df['Spectr_class'] == 'A', 'Spectr_class'] = 3
df_kod.loc[df['Spectr_class'] == 'F', 'Spectr_class'] = 4
df_kod.loc[df['Spectr_class'] == 'O', 'Spectr_class'] = 5
df_kod.loc[df['Spectr_class'] == 'K', 'Spectr_class'] = 6
df_kod.loc[df['Spectr_class'] == 'G', 'Spectr_class'] = 7
df_kod['Spectr_class'] = pd.to_numeric(df_kod['Spectr_class'])
```

```
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\pandas\core\indexi
ng.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\pandas\core\indexi
ng.py:692: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\ipykernel_launche
r.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

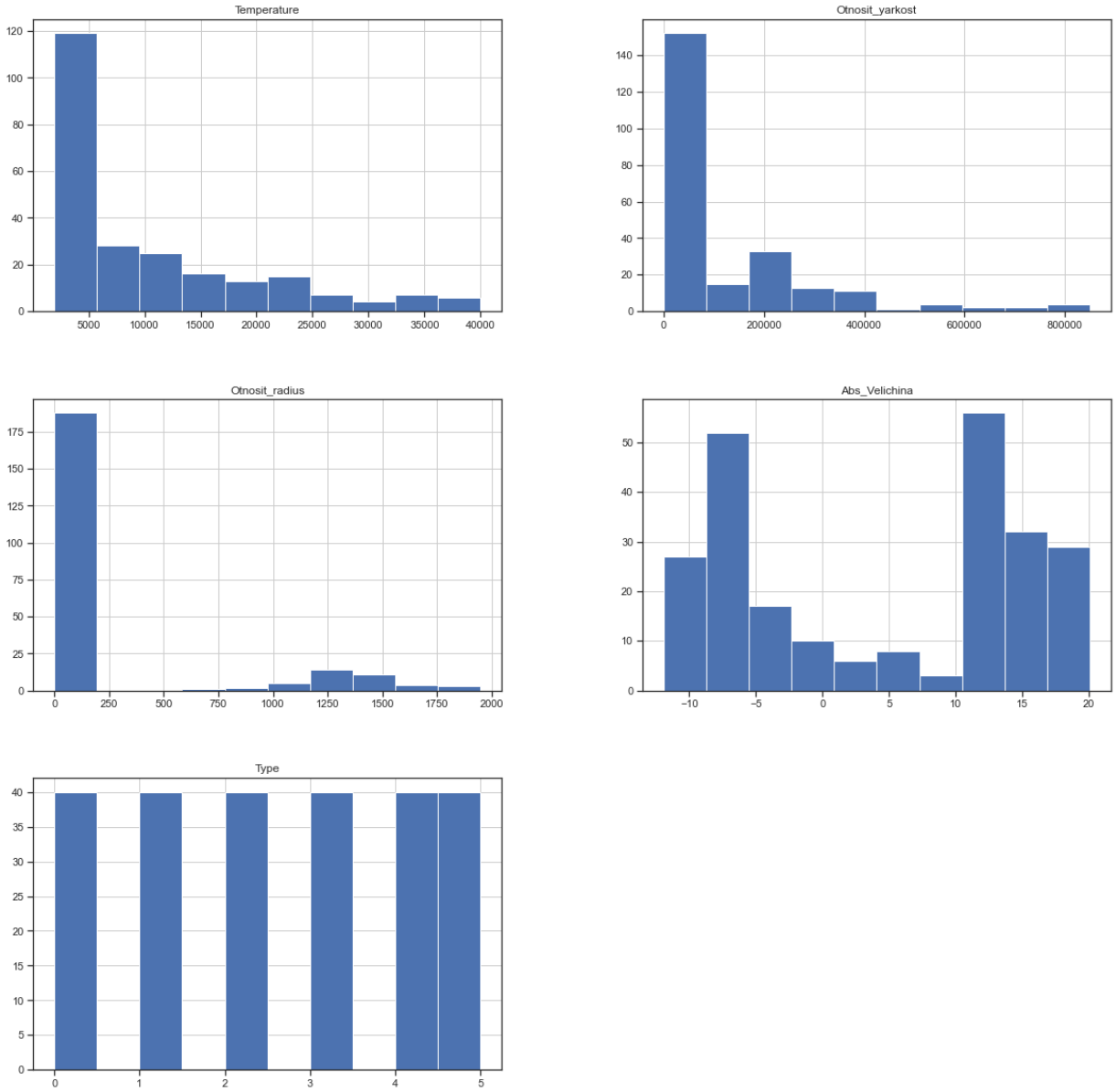
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [26]:
```
df_kod['Spectr_class'].unique()
```

Out[26]:
```
array([1, 2, 3, 4, 5, 6, 7], dtype=int64)
```

In [27]:
```
#Нормализация числовых признаков
df_sr.hist(figsize=(20,20))
plt.show()
```

```
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\pandas\plotting\_matplotlib\tools.py:400: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```

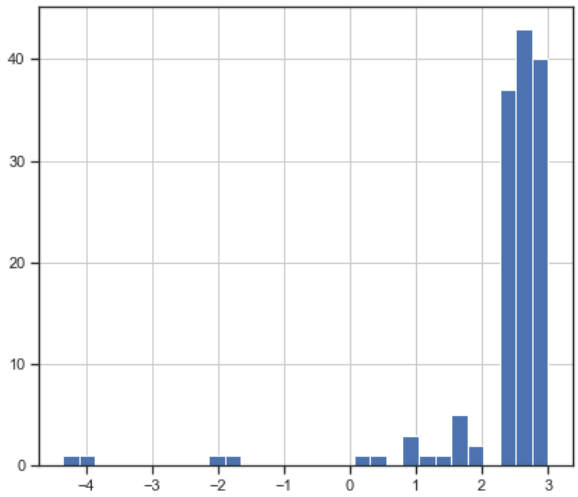


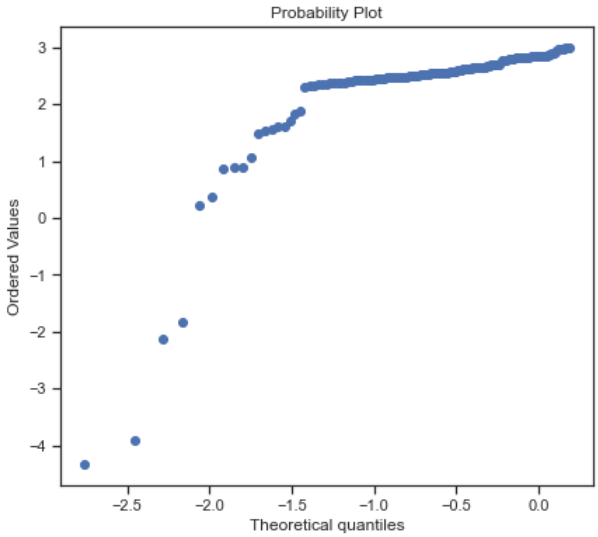
In [28]:
```
df_sr['Abs_Velichina_log'] = np.log(df_sr['Abs_Velichina'])
diagnostic_plots(df_sr, 'Abs_Velichina_log')
```

```
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\pandas\core\arraylike.py:358: RuntimeWarning: invalid value encountered in log
```

```
  result = getattr(ufunc, method)(*inputs, **kwargs)
c:\users\sveta\documents\virtualenvs\tensorflow\lib\site-packages\ipykernel_launche
r.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```





In [ ]: