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

Отчёт по лабораторной работе № 3

По дисциплине:
«Технологии машинного обучения»

По теме:
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»

Выполнил:

Студент группы ИУ5-63 _____ **Труфанов В.А.**

(Подпись, дата) (Фамилия И.О.)

Проверил:

_____ **Гапанюк Ю.Е.**

(Подпись, дата) (Фамилия И.О.)

Москва, 2020

Цель лабораторной работы

Изучение способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

1. Выбрать набор данных (датасет), содержащий категориальные признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.)
2. Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:
   A. обработку пропусков в данных;
   B. кодирование категориальных признаков;
   C. масштабирование данных.

1. Имопрт библиотек и данных

In [37]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/fifa19/data.csv
```

In [38]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [39]:
```
# Импорт данных
fifa_data = pd.read_csv('/kaggle/input/fifa19/data.csv', sep=",")
```

2. Характеристики датасета

In [40]:
```
# размер набора данных
fifa_data.shape
```

Out[40]:
```
(18207, 89)
```

In [41]:
```
# типы колонок
fifa_data.dtypes
```

Out[41]:
```
Unnamed: 0          int64
ID                  int64
Name               object
Age                 int64
Photo              object
                    ...
GKHandling        float64
GKKicking         float64
GKPositioning     float64
GKReflexes        float64
Release Clause     object
Length: 89, dtype: object
```

In [42]:
```
# Первые 5 строк датасета
fifa_data.head()
```

Out[42]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potentia
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	9
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	9
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	9
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	9
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	9

5 rows × 89 columns

In [43]:
```
# проверим есть ли пропущенные значения
fifa_data.isnull().sum()
```

Out[43]:
```
Unnamed: 0           0
ID                   0
Name                 0
Age                  0
Photo                0
                   ...
GKHandling          48
GKKicking           48
GKPositioning       48
GKReflexes          48
Release Clause    1564
Length: 89, dtype: int64
```

3. Обработка пропусков в числовых данных

3.1. Подбор и преобразование колонки

In [44]:
```
# Выбранная колонка Release Clause
fifa_data['Release Clause']
```

Out[44]:
```
0        €226.5M
1        €127.1M
2        €228.1M
3        €138.6M
4        €196.4M
          ...
18202      €143K
18203      €113K
18204      €165K
18205      €143K
18206      €165K
Name: Release Clause, Length: 18207, dtype: object
```

In [45]:
```
# первые строки с неопределенными значениями
fifa_data[fifa_data['Release Clause'].isnull()].head()
```

Out[45]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Pote
28	28	198710	J. Rodríguez	26	https://cdn.sofifa.org/players/4/19/198710.png	Colombia	https://cdn.sofifa.org/flags/56.png	88	
38	38	167664	G. Higuaín	30	https://cdn.sofifa.org/players/4/19/167664.png	Argentina	https://cdn.sofifa.org/flags/52.png	88	
91	91	187961	Paulinho	29	https://cdn.sofifa.org/players/4/19/187961.png	Brazil	https://cdn.sofifa.org/flags/54.png	85	
166	166	212523	Anderson Talisca	24	https://cdn.sofifa.org/players/4/19/212523.png	Brazil	https://cdn.sofifa.org/flags/54.png	83	
176	176	207410	M. Kovačić	24	https://cdn.sofifa.org/players/4/19/207410.png	Croatia	https://cdn.sofifa.org/flags/10.png	83	

5 rows × 89 columns

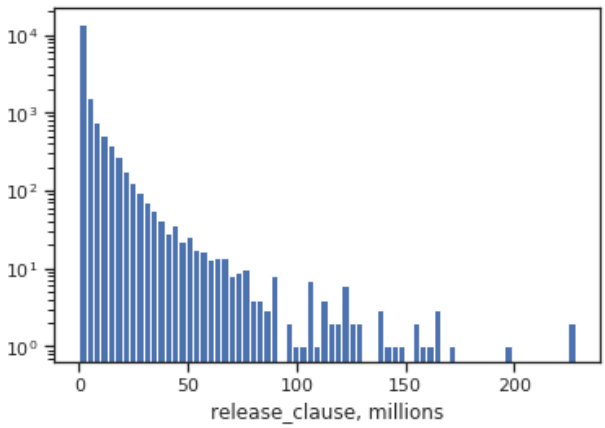
In [46]:
```
# Индекс пропущенных значений
fifa_data[fifa_data['Release Clause'].isnull()].index
```

Out[46]:
```
Int64Index([   28,    38,    91,   166,   176,   332,   354,   357,   427,
              434,
            ...
            17634, 17672, 17726, 17752, 17978, 17979, 18026, 18031, 18056,
            18183],
           dtype='int64', length=1564)
```

In [47]:
```
# Перевод значений признака в чиловой вид
# Сначала перевод NaN в 0
# Затем перевод строкового формата в числовой
release_clause = fifa_data['Release Clause'].fillna('0').replace
({'€':'','K': '*1e3', 'M': '*1e6'}, regex=True)\
                                    .map(pd.eval).astype(int)
release_clause.rename('Release Clause')
release_clause.rename_axis("animal")
release_clause
```

Out[47]:
```
0        226500000
1        127100000
2        228100000
3        138600000
4        196400000
           ...
18202       143000
18203       113000
18204       165000
18205       143000
18206       165000
Name: Release Clause, Length: 18207, dtype: int64
```

In [49]:
```
# Гистограмма признака в лограифмическом масштабе
plt.hist(release_clause.divide(other=1e6), 70,log=True)
plt.xlabel("release_clause, millions")
#xint = range(release_clause.min(), release_clause.max())
plt.show()
```



3.2 Заполнение пропусков с SimpleImputer:

In [50]:
```
# Импорт из библиотек
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [51]:
```
#Функция для применения различных стратегий импьютации для колонки
def test_num_impute(strategy_param, column, missing_val=np.nan):
    # Фильтр для проверки заполнения пустых значений
    indicator = MissingIndicator(missing_values=missing_val)
    mask_missing_values_only = indicator.fit_transform(column)
    #Simple Imputer
    imp_num = SimpleImputer(strategy=strategy_param,missing_values=missi
ng_val)
    data_num_imp = imp_num.fit_transform(column)
    return data_num_imp[mask_missing_values_only]
```

In [52]:
```
# Функция для печати результатов вставки пропусков, используя список стр
атегий
def test_strategies(strategies,df,missing_val=np.nan):
    for index, strategy in enumerate(strategies):
        print(df.iloc[:,0].name,strategy,test_num_impute(strategy,df,mis
sing_val))
```

In [53]:
```
#Выводить массив с ..., если больше 100 значений
np.set_printoptions(threshold=100)
```

In [54]:
```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator(missing_values=0)
mask_missing_values_only = indicator.fit_transform(release_clause.to_fra
me())
np.where(mask_missing_values_only)
```

Out[54]:
```
(array([   28,    38,    91, ..., 18031, 18056, 18183]),
 array([0, 0, 0, ..., 0, 0, 0]))
```

In [55]:
```
# Характеристики признака
release_clause.describe()
```

Out[55]:
```
count    1.820700e+04
mean     4.191200e+06
std      1.070778e+07
min      0.000000e+00
25%      3.920000e+05
50%      1.000000e+06
75%      3.000000e+06
max      2.281000e+08
Name: Release Clause, dtype: float64
```

In [56]:
```
# Применение стратегий и вывод значений
strategies=['mean', 'median','most_frequent']
test_strategies(strategies,release_clause.to_frame(),0)
```

```
Release Clause mean [4585060.98185423 4585060.98185423 4585060.98185423
... 4585060.98185423
 4585060.98185423 4585060.98185423]
Release Clause median [1100000. 1100000. 1100000. ... 1100000. 1100000. 1
100000.]
Release Clause most_frequent [1100000 1100000 1100000 ... 1100000 1100000
1100000]
```

4. Обработка пропусков в категориальных данных

4.1 Выбор колонки

In [57]:
```
# Поиск категориальных признаков, в которых достаточное количество пропу
сков в процентах
total_count = len(fifa_data.index)
fifa_data.select_dtypes(include=['object']).isnull().sum(axis = 0).where
(lambda x : x>0)\
         .sort_values(ascending=True).apply(lambda x: x/total_count*10
0).head(10)
```

Out[57]:
```
Preferred Foot          0.263635
Work Rate               0.263635
Body Type               0.263635
Real Face               0.263635
Weight                  0.263635
Height                  0.263635
Position                0.329544
Club                    1.323667
Contract Valid Until    1.587302
Joined                  8.529686
dtype: float64
```

In [58]:
```
# Данные в выбранной колонке
fifa_data['Club']
```

Out[58]:
```
0               FC Barcelona
1                   Juventus
2        Paris Saint-Germain
3          Manchester United
4            Manchester City
                 ...
18202        Crewe Alexandra
18203         Trelleborgs FF
18204       Cambridge United
18205        Tranmere Rovers
18206        Tranmere Rovers
Name: Club, Length: 18207, dtype: object
```

In [59]:
```
# вывод информации об уникальных значениях
print('Club:',' число уникальных значений:',fifa_data['Club'].nunique(),
      ",количество пустых значений: ",fifa_data['Club'].isnull().sum())
```

```
Club:  число уникальных значений: 651 ,количество пустых значений:  241
```

4.2 Заполнение пропусков в данных

In [60]:
```
# Индексы пропущенных значений
fifa_data[fifa_data['Club'].isnull()].index
```

Out[60]:
```
Int64Index([  452,   538,   568,   677,   874,   953,   997,  1008,  112
0,
             1271,
            ...
            16903, 16947, 16976, 17008, 17129, 17197, 17215, 17339, 1743
6,
            17539],
           dtype='int64', length=241)
```

In [61]:
```
# Применение стратегий и вывод заполненных значений
strategies = ['most_frequent','constant']
test_strategies(strategies, fifa_data[['Club']])
```

```
Club most_frequent ['AS Monaco' 'AS Monaco' 'AS Monaco' ... 'AS Monaco' '
AS Monaco'
 'AS Monaco']
Club constant ['missing_value' 'missing_value' 'missing_value' ... 'missi
ng_value'
 'missing_value' 'missing_value']
```

5. Преобразование категориальных признаков

5.1 Поиск колонки для Label encoding

In [62]:
```
# Поиск категориальных признаков, в которых мало уникальных значений
uniqueObj = fifa_data.select_dtypes(include=['object']).nunique().sort_v
alues().head(5)
print(uniqueObj)
```

```
Preferred Foot     2
Real Face          2
Work Rate          9
Body Type         10
Height            21
dtype: int64
```

In [63]:
```
# Вывод категориальных признаков с указанием уникальных значений
categoryCols = uniqueObj.index.tolist()
for col in fifa_data[categoryCols]:
    print(col," : ",fifa_data[col].unique(),",количество пустых значени
й: ",fifa_data[col].isnull().sum())
```

```
Preferred Foot  :  ['Left' 'Right' nan] ,количество пустых значений:  48
Real Face  :  ['Yes' 'No' nan] ,количество пустых значений:  48
Work Rate  :  ['Medium/ Medium' 'High/ Low' 'High/ Medium' 'High/ High' '
Medium/ High'
 'Medium/ Low' 'Low/ High' 'Low/ Medium' 'Low/ Low' nan] ,количество пуст
ых значений:  48
Body Type  :  ['Messi' 'C. Ronaldo' 'Neymar' 'Lean' 'Normal' 'Courtois' '
Stocky'
 'PLAYER_BODY_TYPE_25' 'Shaqiri' 'Akinfenwa' nan] ,количество пустых знач
ений:  48
Height  :  ["5'7" "6'2" "5'9" "6'4" "5'11" "5'8" "6'0" "5'6" "5'10" "6'6"
"6'1" "5'4"
 "6'3" "5'5" "6'5" "6'7" "5'3" "5'2" "6'8" "5'1" "6'9" nan] ,количество п
устых значений:  48
```

In [64]:
```
# Проверка на связь строк, в которых неопределенные значения, чтобы удал
ить их из датасета
fifa_data[(fifa_data[categoryCols[0]].isnull())][categoryCols].head()
```

Out[64]:

	Preferred Foot	Real Face	Work Rate	Body Type	Height
13236	NaN	NaN	NaN	NaN	NaN
13237	NaN	NaN	NaN	NaN	NaN
13238	NaN	NaN	NaN	NaN	NaN
13239	NaN	NaN	NaN	NaN	NaN
13240	NaN	NaN	NaN	NaN	NaN

In [65]:
```
#Удаляем эти строки, так как значения не определены во всех колонках
fifa_data = fifa_data[fifa_data[categoryCols[0]].notna()]
for col in fifa_data[categoryCols]:
    print(col," : ",fifa_data[col].unique(),",количество пустых значени
й: ",fifa_data[col].isnull().sum())
```

```
Preferred Foot  :  ['Left' 'Right'] ,количество пустых значений:  0
Real Face  :  ['Yes' 'No'] ,количество пустых значений:  0
Work Rate  :  ['Medium/ Medium' 'High/ Low' 'High/ Medium' 'High/ High' '
Medium/ High'
 'Medium/ Low' 'Low/ High' 'Low/ Medium' 'Low/ Low'] ,количество пустых з
начений:  0
Body Type  :  ['Messi' 'C. Ronaldo' 'Neymar' 'Lean' 'Normal' 'Courtois' '
Stocky'
 'PLAYER_BODY_TYPE_25' 'Shaqiri' 'Akinfenwa'] ,количество пустых значени
й:  0
Height  :  ["5'7" "6'2" "5'9" "6'4" "5'11" "5'8" "6'0" "5'6" "5'10" "6'6"
"6'1" "5'4"
 "6'3" "5'5" "6'5" "6'7" "5'3" "5'2" "6'8" "5'1" "6'9"] ,количество пусты
х значений:  0
```

5.2 Label encoding

In [66]:
```
# импорт из библиотек
from sklearn.preprocessing import LabelEncoder
```

In [67]:
```
#Label encoding для выбранной колонки Work Rate
le = LabelEncoder()
work_rate = fifa_data["Work Rate"]
cat_foot_le = le.fit_transform(work_rate)
print(work_rate.name,",label encoded: ",cat_foot_le)
print(work_rate.name,",unique values: ",np.unique(cat_foot_le))
print(work_rate.name,",source values: ",le.inverse_transform(cat_foot_l
e))
```

```
Work Rate ,label encoded:  [8 1 2 ... 8 8 8]
Work Rate ,unique values:  [0 1 2 3 4 5 6 7 8]
Work Rate ,source values:  ['Medium/ Medium' 'High/ Low' 'High/ Medium'
... 'Medium/ Medium'
 'Medium/ Medium' 'Medium/ Medium']
```

5.3 One hot encoding

In [68]:
```
#one-hot encoding для выбранной колонки Body Type
pd.get_dummies(fifa_data["Body Type"],dummy_na=True,prefix="Rate").head
()
```

Out[68]:

	Rate_Akinfenwa	Rate_C. Ronaldo	Rate_Courtois	Rate_Lean	Rate_Messi	Rate_Neymar	Rate_Normal	Rate_Pl
0	0	0	0	0	1	0	0	
1	0	1	0	0	0	0	0	
2	0	0	0	0	0	1	0	
3	0	0	0	1	0	0	0	
4	0	0	0	0	0	0	1	

6. Масштабирование данных

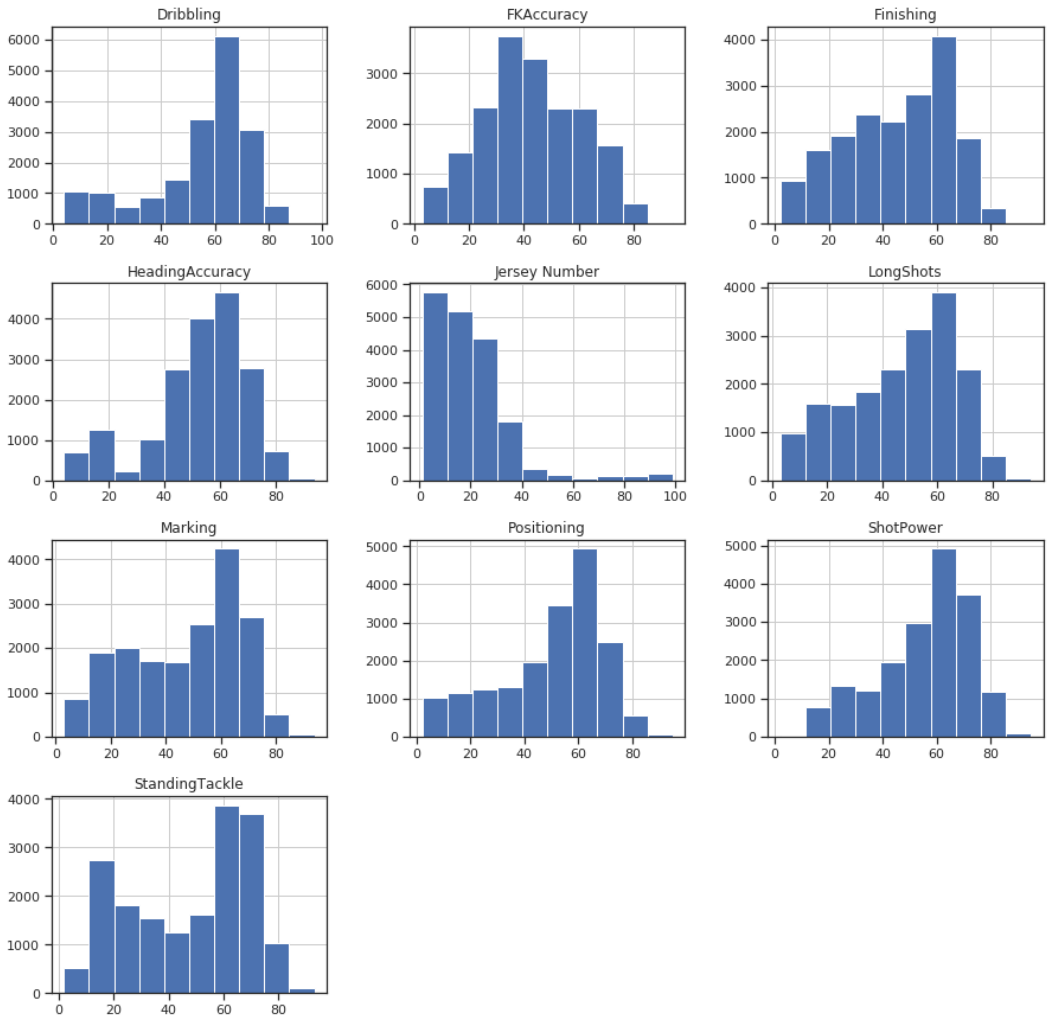
6.1 Подбор колонки

In [69]:
```
#Выборка 10 количественных признаков с количеством уникальных значений м
еньше 100
quantityCols = fifa_data.select_dtypes(exclude=['object']).nunique().whe
re(lambda x : x<100)\
                        .sort_values(ascending=False).head(10)
print(quantityCols)
```

```
Jersey Number      99.0
Dribbling          94.0
Positioning        94.0
Finishing          93.0
Marking            92.0
ShotPower          92.0
LongShots          92.0
HeadingAccuracy    91.0
StandingTackle     90.0
FKAccuracy         90.0
dtype: float64
```

In [70]:
```
# Гистограммы для этих признаков
fifa_data[quantityCols.index.tolist()].hist(figsize=(15,15))
```

Out[70]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f03de8e44d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de737d50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de75ef50
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f03de70fbd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de6cbe90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de67fb50
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f03de63ffd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de5f5ad0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de5ff650
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f03de5b3fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de52add0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f03de945590
>]],
      dtype=object)
```

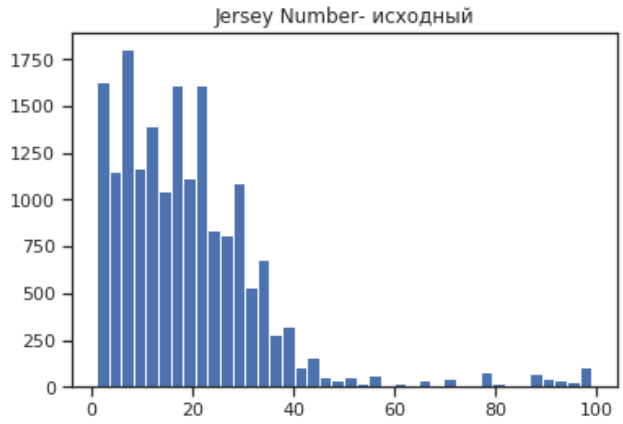


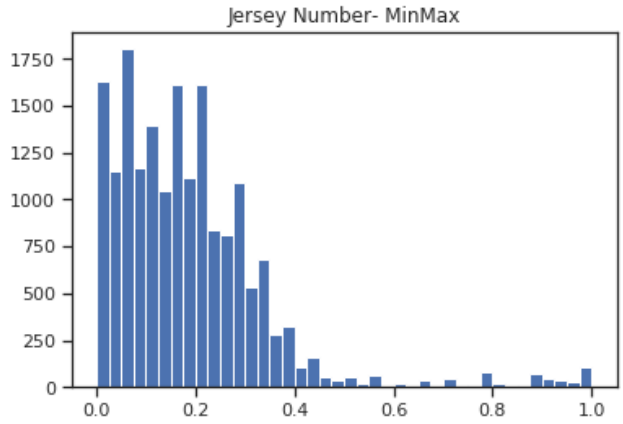
6.2 Масштабирование различными методами

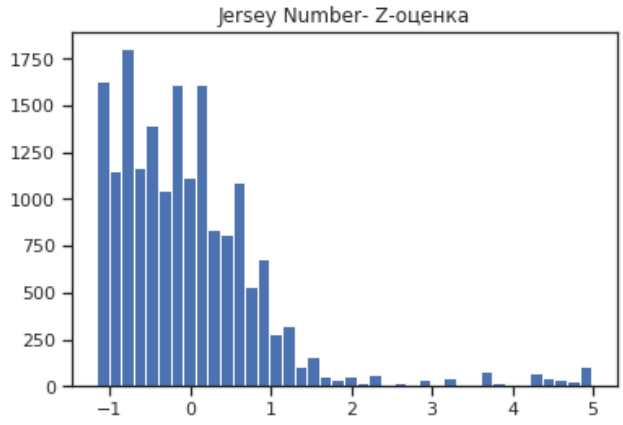
In [71]:
```
# импорт из библиотек
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

In [72]:
```
# Выбранная колонка - Jersey Number
# MinMax масштабирование
mms = MinMaxScaler()
jersey_number = fifa_data['Jersey Number']
mms_acceleration = mms.fit_transform(jersey_number.to_frame())
# Масштабирование данных на основе Z-оценки
sts = StandardScaler()
sts_acceleration = sts.fit_transform(jersey_number.to_frame())
# Построение гистограмм
plt.hist(jersey_number,40)
plt.title(jersey_number.name+"- исходный")
plt.show()
plt.hist(mms_acceleration,40)
plt.title(jersey_number.name+"- MinMax")
plt.show()
plt.hist(sts_acceleration,40)
plt.title(jersey_number.name+"- Z-оценка")
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/numpy/lib/histograms.py:839: Runti
meWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
/opt/conda/lib/python3.7/site-packages/numpy/lib/histograms.py:840: Runti
meWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```







Выводы

В результате выполнения работы были изучены методы устранения пропусков в данных, преобразования данных из категориальных в количественные, а также масштабирование данных.