Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Дисциплина «Технологии машинного обучения»

Отчёт

по лабораторной работе №3

«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»

Вариант 12

Студент:

Крюков Г. М.

Группа ИУ5-61Б

Преподаватель:

Гапанюк Ю. Е.

Москва, 2020 г.

Цель лабораторной работы:

Изучение способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

- 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.)
- Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:

    - обработку пропусков в данных;
    - кодирование категориальных признаков;
    - масштабирование данных.

Выполнение работы:

Выбранные датасеты:

https://www.kaggle.com/cjgdev/formula-1-race-data-19502017?select=results.csv

https://www.kaggle.com/rohanrao/formula-1-world-championship-1950-2020?select=drivers.csv

1. Загрузка и первичный анализ данных


```
data = pd.read_csv('/content/results.csv')


total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 23777
```

```
# Первые 5 строк датасета
data.head()
```

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId
0	1	18	1	1	22.0	1	1.0	1	1	10.0	58	34:50.6	5690616.0	39.0	2.0	01:27.5	218.3	1
1	2	18	2	2	3.0	5	2.0	2	2	8.0	58	5.478	5696094.0	41.0	3.0	01:27.7	217.586	1
2	3	18	3	3	7.0	7	3.0	3	3	6.0	58	8.163	5698779.0	41.0	5.0	01:28.1	216.719	1
3	4	18	4	4	5.0	11	4.0	4	4	5.0	58	17.181	5707797.0	58.0	7.0	01:28.6	215.464	1
4	5	18	5	1	23.0	3	5.0	5	5	4.0	58	18.014	5708630.0	43.0	1.0	01:27.4	218.385	1

```
# размер набора данных
data.shape
```

```
(23777, 18)
```

```
# типы колонок
data.dtypes
```

```
resultId             int64
raceId               int64
driverId             int64
constructorId        int64
number             float64
grid                 int64
position           float64
positionText        object
positionOrder        int64
points             float64
laps                 int64
time                object
milliseconds       float64
fastestLap         float64
rank               float64
fastestLapTime      object
fastestLapSpeed     object
statusId             int64
dtype: object
```

```
data.isnull().sum()
```

```
resultId                0
raceId                  0
driverId                0
constructorId           0
number                  6
grid                    0
position            10550
positionText            0
positionOrder           0
points                  0
laps                    0
time                17773
milliseconds        17774
fastestLap          18394
rank                18246
fastestLapTime      18394
fastestLapSpeed     18394
statusId                0
dtype: int64
```

2. Обработка пропусков в данных

2.1. Простые стратегии - удаление или заполнение нулями

```
# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

```
((23777, 18), (23777, 10))
```

```
# Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

```
((23777, 18), (2604, 18))
```

```
data.head()
```

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId
0	1	18	1	1	22.0	1	1.0	1	1	10.0	58	34:50.6	5690616.0	39.0	2.0	01:27.5	218.3	1
1	2	18	2	2	3.0	5	2.0	2	2	8.0	58	5.478	5696094.0	41.0	3.0	01:27.7	217.586	1
2	3	18	3	3	7.0	7	3.0	3	3	6.0	58	8.163	5698779.0	41.0	5.0	01:28.1	216.719	1
3	4	18	4	4	5.0	11	4.0	4	4	5.0	58	17.181	5707797.0	58.0	7.0	01:28.6	215.464	1
4	5	18	5	1	23.0	3	5.0	5	5	4.0	58	18.014	5708630.0	43.0	1.0	01:27.4	218.385	1

```
# Заполнение всех пропущенных значений нулями
data_new_3 = data.fillna(0)
data_new_3.head()
```

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId
0	1	18	1	1	22.0	1	1.0	1	1	10.0	58	34:50.6	5690616.0	39.0	2.0	01:27.5	218.3	1
1	2	18	2	2	3.0	5	2.0	2	2	8.0	58	5.478	5696094.0	41.0	3.0	01:27.7	217.586	1
2	3	18	3	3	7.0	7	3.0	3	3	6.0	58	8.163	5698779.0	41.0	5.0	01:28.1	216.719	1
3	4	18	4	4	5.0	11	4.0	4	4	5.0	58	17.181	5707797.0	58.0	7.0	01:28.6	215.464	1
4	5	18	5	1	23.0	3	5.0	5	5	4.0	58	18.014	5708630.0	43.0	1.0	01:27.4	218.385	1

2.2. "Внедрение значений" - импьютация (imputation)

```
# Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

```
Колонка number. Тип данных float64. Количество пустых значений 6, 0.03%.
Колонка position. Тип данных float64. Количество пустых значений 10550, 44.37%.
Колонка milliseconds. Тип данных float64. Количество пустых значений 17774, 74.75%.
Колонка fastestLap. Тип данных float64. Количество пустых значений 18394, 77.36%.
Колонка rank. Тип данных float64. Количество пустых значений 18246, 76.74%.
```

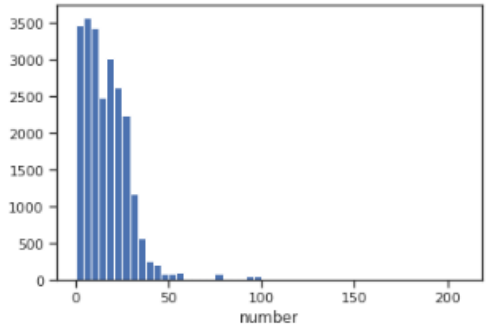
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

	number	position	milliseconds	fastestLap	rank
0	22.0	1.0	5690616.0	39.0	2.0
1	3.0	2.0	5696094.0	41.0	3.0
2	7.0	3.0	5698779.0	41.0	5.0
3	5.0	4.0	5707797.0	58.0	7.0
4	23.0	5.0	5708630.0	43.0	1.0
...	...	...	...	...	...
23772	10.0	16.0	NaN	33.0	16.0
23773	9.0	17.0	NaN	36.0	15.0
23774	18.0	18.0	NaN	52.0	6.0
23775	55.0	NaN	NaN	26.0	14.0
23776	3.0	NaN	NaN	13.0	12.0

23777 rows × 5 columns

```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:839: RuntimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:840: RuntimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```




```
# Фильтр по пустым значениям поля salary
data[data['position'].isnull()]
```

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId
8	9	18	9	2	4.0	2	NaN	R	9	0.0	47	NaN	NaN	15.0	9.0	01:28.8	215.1	4
9	10	18	10	7	12.0	18	NaN	R	10	0.0	43	NaN	NaN	23.0	13.0	01:29.6	213.166	3
10	11	18	11	8	18.0	19	NaN	R	11	0.0	32	NaN	NaN	24.0	15.0	01:30.9	210.038	7
11	12	18	12	4	6.0	20	NaN	R	12	0.0	30	NaN	NaN	20.0	16.0	01:31.4	208.907	8
12	13	18	13	6	2.0	4	NaN	R	13	0.0	29	NaN	NaN	23.0	6.0	01:28.2	216.51	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
23754	23759	987	839	10	31.0	10	NaN	R	18	0.0	0	NaN	NaN	NaN	0.0	NaN	NaN	3
23755	23760	987	838	1	2.0	12	NaN	R	19	0.0	0	NaN	NaN	NaN	0.0	NaN	NaN	3
23756	23761	987	825	210	20.0	13	NaN	R	20	0.0	0	NaN	NaN	NaN	0.0	NaN	NaN	3
23775	23780	988	832	4	55.0	12	NaN	R	19	0.0	31	NaN	NaN	26.0	14.0	01:43.4	193.41	36
23776	23781	988	817	9	3.0	4	NaN	R	20	0.0	20	NaN	NaN	13.0	12.0	01:42.8	194.579	9

10550 rows × 18 columns

```
# Запоминаем индексы строк с пустыми значениями
flt_index = data[data['position'].isnull()].index
flt_index
```

```
Int64Index([    8,     9,    10,    11,    12,    13,    14,    15,    16,
               17,
            ...
            23733, 23734, 23735, 23736, 23753, 23754, 23755, 23756, 23775,
            23776],
           dtype='int64', length=10550)
```

```
# Проверяем что выводятся нужные строки
data[data.index.isin(flt_index)]
```

	resultId	raceId	driverId	constructorId	number	grid	position	positionText	positionOrder	points	laps	time	milliseconds	fastestLap	rank	fastestLapTime	fastestLapSpeed	statusId
8	9	18	9	2	4.0	2	NaN	R	9	0.0	47	NaN	NaN	15.0	9.0	01:28.8	215.1	4
9	10	18	10	7	12.0	18	NaN	R	10	0.0	43	NaN	NaN	23.0	13.0	01:29.6	213.166	3
10	11	18	11	8	18.0	19	NaN	R	11	0.0	32	NaN	NaN	24.0	15.0	01:30.9	210.038	7
11	12	18	12	4	6.0	20	NaN	R	12	0.0	30	NaN	NaN	20.0	16.0	01:31.4	208.907	8
12	13	18	13	6	2.0	4	NaN	R	13	0.0	29	NaN	NaN	23.0	6.0	01:28.2	216.51	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
23754	23759	987	839	10	31.0	10	NaN	R	18	0.0	0	NaN	NaN	NaN	0.0	NaN	NaN	3
23755	23760	987	838	1	2.0	12	NaN	R	19	0.0	0	NaN	NaN	NaN	0.0	NaN	NaN	3
23756	23761	987	825	210	20.0	13	NaN	R	20	0.0	0	NaN	NaN	NaN	0.0	NaN	NaN	3
23775	23780	988	832	4	55.0	12	NaN	R	19	0.0	31	NaN	NaN	26.0	14.0	01:43.4	193.41	36
23776	23781	988	817	9	3.0	4	NaN	R	20	0.0	20	NaN	NaN	13.0	12.0	01:42.8	194.579	9

10550 rows × 18 columns

```
# фильтр по колонке
data_num[data_num.index.isin(flt_index)]['position']
```

```
8       NaN
9       NaN
10      NaN
11      NaN
12      NaN
         ..
23754   NaN
23755   NaN
23756   NaN
23775   NaN
23776   NaN
Name: position, Length: 10550, dtype: float64
```

```
data_num_position = data_num[['position']]
data_num_position.head()
```

	position
0	1.0
1	2.0
2	3.0
3	4.0
4	5.0

```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_position)
mask_missing_values_only
```

```
array([[False],
       [False],
       [False],
       ...,
       [False],
       [ True],
       [ True]])
```

```
strategies=['mean', 'median','most_frequent']
```

```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_position)
    return data_num_imp[mask_missing_values_only]
```

```
strategies[0], test_num_impute(strategies[0])
```

```
('mean',
 array([7.78226355, 7.78226355, 7.78226355, ..., 7.78226355, 7.78226355,
        7.78226355]))
```

```
strategies[1], test_num_impute(strategies[1])
```

```
('median', array([7., 7., 7., ..., 7., 7., 7.]))
```

```
strategies[2], test_num_impute(strategies[2])
```

```
('most_frequent', array([3., 3., 3., ..., 3., 3., 3.]))
```

```
# Более сложная функция, которая позволяет задавать колонку и вид импьютации
def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]

    return column, strategy_param, filled_data.size, filled_data[0], filled_data[filled_data.size-1]
```

```
data[['position']].describe()
```

	position
count	13227.000000
mean	7.782264
std	4.745105
min	1.000000
25%	4.000000
50%	7.000000
75%	11.000000
max	33.000000

```
test_num_impute_col(data, 'position', strategies[0])
```

```
('position', 'mean', 10550, 7.782263551825811, 7.782263551825811)
```

```
test_num_impute_col(data, 'position', strategies[1])
```

```
('position', 'median', 10550, 7.0, 7.0)
```

```
test_num_impute_col(data, 'position', strategies[2])
```

```
('position', 'most_frequent', 10550, 3.0, 3.0)
```

3. Преобразование категориальных признаков в числовые

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
data = pd.read_csv('/content/datasets_468218_878459_drivers.csv')
data.head()
```

	driverId	driverRef	number	code	forename	surname	dob	nationality	url
0	1	hamilton	44	HAM	Lewis	Hamilton	1985-01-07	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
1	2	heidfeld	\N	HEI	Nick	Heidfeld	1977-05-10	German	http://en.wikipedia.org/wiki/Nick_Heidfeld
2	3	rosberg	6	ROS	Nico	Rosberg	1985-06-27	German	http://en.wikipedia.org/wiki/Nico_Rosberg
3	4	alonso	14	ALO	Fernando	Alonso	1981-07-29	Spanish	http://en.wikipedia.org/wiki/Fernando_Alonso
4	5	kovalainen	\N	KOV	Heikki	Kovalainen	1981-10-19	Finnish	http://en.wikipedia.org/wiki/Heikki_Kovalainen

```
le = LabelEncoder()
```

```
cat_temp_data = data[['nationality']]
cat_temp_data.head()
```

	nationality
0	British
1	German
2	German
3	Spanish
4	Finnish

```
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
```

```
cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

	c1
0	British
1	German
2	German
3	Spanish
4	Finnish
...	...
842	Monegasque
843	Russian
844	British
845	British
846	Thai

847 rows × 1 columns

3.1.Кодирование категорий целочисленными значениями - label encoding

```
cat_enc['c1'].unique()
```

```
array(['British', 'German', 'Spanish', 'Finnish', 'Japanese', 'French',
       'Polish', 'Brazilian', 'Italian', 'Australian', 'Austrian',
       'American', 'Dutch', 'Colombian', 'Portuguese', 'Canadian',
       'Indian', 'Hungarian', 'Irish', 'Danish', 'Argentine', 'Czech',
       'Malaysian', 'Swiss', 'Belgian', 'Monegasque', 'Swedish',
       'Venezuelan', 'New Zealander', 'Chilean', 'Mexican',
       'South African', 'Liechtensteiner', 'Rhodesian',
       'American-Italian', 'Uruguayan', 'Argentine-Italian', 'Thai',
       'East German', 'Russian', 'Indonesian'], dtype=object)
```

```
cat_enc_le = le.fit_transform(cat_enc['c1'])
np.unique(cat_enc_le)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40])
```

```
le.inverse_transform([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40])
```

```
array(['American', 'American-Italian', 'Argentine', 'Argentine-Italian',
       'Australian', 'Austrian', 'Belgian', 'Brazilian', 'British',
       'Canadian', 'Chilean', 'Colombian', 'Czech', 'Danish', 'Dutch',
       'East German', 'Finnish', 'French', 'German', 'Hungarian',
       'Indian', 'Indonesian', 'Irish', 'Italian', 'Japanese',
       'Liechtensteiner', 'Malaysian', 'Mexican', 'Monegasque',
       'New Zealander', 'Polish', 'Portuguese', 'Rhodesian', 'Russian',
       'South African', 'Spanish', 'Swedish', 'Swiss', 'Thai',
       'Uruguayan', 'Venezuelan'], dtype=object)
```

3.2. Кодирование категорий наборами бинарных значений - one-hot encoding

```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])
```

```
cat_enc.shape
```

```
(847, 1)
```

```
cat_enc_ohe.shape
```

```
(847, 41)
```

```
cat_enc_ohe
```

```
<847x41 sparse matrix of type '<class 'numpy.float64'>'
        with 847 stored elements in Compressed Sparse Row format>
```

```
cat_enc_ohe.todense()[0:10]
```

```
matrix([[0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 1., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.]])
```

```
cat_enc.head(10)
```

	c1
0	British
1	German
2	German
3	Spanish
4	Finnish
5	Japanese
6	French
7	Finnish
8	Polish
9	German

3.3. Pandas get_dummies - быстрый вариант one-hot кодирования

```
pd.get_dummies(cat_enc).head()
```

	c1_American	c1_American-Italian	c1_Argentine	c1_Argentine-Italian	c1_Australian	c1_Austrian	c1_Belgian	c1_Brazilian	c1_British	c1_Canadian	c1_Chilean	c1_Colombian
0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0

```
pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

	nationality_American	nationality_American-Italian	nationality_Argentine	nationality_Argentine-Italian	nationality_Australian	nationality_Austrian	nationality_Belgian
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

4. Масштабирование данных

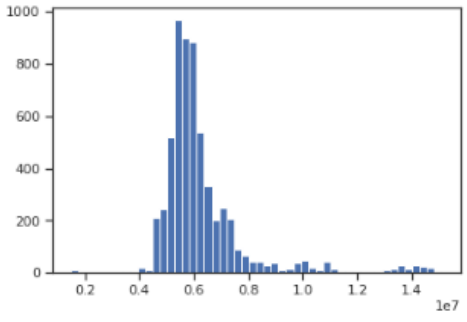
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

4.1. MinMax масштабирование

```
data = pd.read_csv('/content/results.csv')
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['milliseconds']])
```

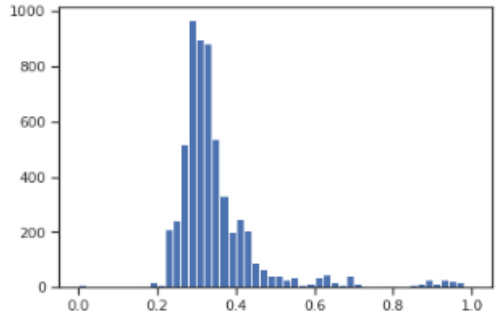
```
plt.hist(data['milliseconds'], 50)
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:839: RuntimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:840: RuntimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```



```
plt.hist(sc1_data, 50)
plt.show()
```

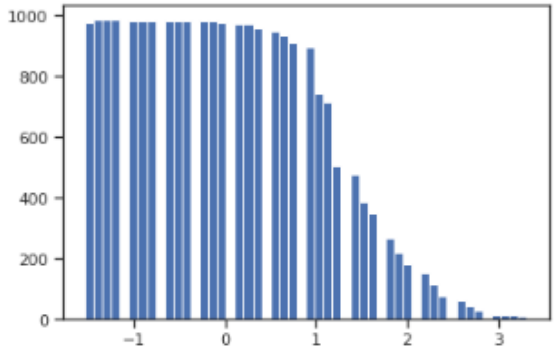
```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.
  keep = (tmp_a >= first_edge)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.
  keep &= (tmp_a <= last_edge)
```



4.2. Масштабирование данных на основе Z-оценки – StandardScaler

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['positionOrder']])

plt.hist(sc2_data, 50)
plt.show()
```



4.3. Нормализация данных

```
sc3 = Normalizer()
sc3_data = sc3.fit_transform(data[['positionOrder']])


plt.hist(sc3_data, 50)
plt.show()
```