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

Лабораторная работа №3
по дисциплине
«Методы машинного обучения»
на тему
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»

Выполнил:
студент группы ИУ5-23М
Наседкин И. А.

_____

Москва — 2020 г.

1. Лабораторная работа №3 по курсу ММО (Наседкин Игорь ИУ5-23М)

[0]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

[0]:
```
#Возьмем дата-сет сведений о бронировании отелей
data = pd.read_csv('hotel_bookings.csv', sep=",")
```

https://www.kaggle.com/jessemostipak/hotel-booking-demand

[0]:
```
# размер набора данных
data.shape
```

[0]:
```
(119390, 32)
```

[0]:
```
# типы колонок
data.dtypes
```

[0]:
```
hotel                           object
is_canceled                      int64
lead_time                        int64
arrival_date_year                int64
arrival_date_month              object
arrival_date_week_number         int64
arrival_date_day_of_month        int64
stays_in_weekend_nights          int64
stays_in_week_nights             int64
adults                           int64
children                       float64
babies                           int64
meal                            object
country                         object
market_segment                  object
distribution_channel            object
is_repeated_guest                int64
previous_cancellations           int64
previous_bookings_not_canceled   int64
reserved_room_type              object
assigned_room_type              object
booking_changes                  int64
deposit_type                    object
agent                          float64
company                        float64
days_in_waiting_list             int64
customer_type                   object
adr                            float64
```

```
required_car_parking_spaces         int64
total_of_special_requests           int64
reservation_status                 object
reservation_status_date            object
dtype: object
```

[0]:
```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

[0]:
```
hotel                               0
is_canceled                         0
lead_time                           0
arrival_date_year                   0
arrival_date_month                  0
arrival_date_week_number            0
arrival_date_day_of_month           0
stays_in_weekend_nights             0
stays_in_week_nights                0
adults                              0
children                            4
babies                              0
meal                                0
country                           488
market_segment                      0
distribution_channel                0
is_repeated_guest                   0
previous_cancellations              0
previous_bookings_not_canceled      0
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
agent                           16340
company                        112593
days_in_waiting_list                0
customer_type                       0
adr                                 0
required_car_parking_spaces         0
total_of_special_requests           0
reservation_status                  0
reservation_status_date             0
dtype: int64
```

[0]:
```
# Первые 5 строк датасета
data.head()
```

[0]:
```
          hotel  is_canceled  …  reservation_status 
↪reservation_status_date
0  Resort Hotel            0  …           Check-Out 
↪2015-07-01
```

```
1  Resort Hotel            0  …           Check-Out
  ↪2015-07-01
2  Resort Hotel            0  …           Check-Out
  ↪2015-07-02
3  Resort Hotel            0  …           Check-Out
  ↪2015-07-02
4  Resort Hotel            0  …           Check-Out
  ↪2015-07-03

[5 rows x 32 columns]
```

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

```
Всего строк: 119390
```

2. Обработка пропусков в данных

[0]:
```
# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

[0]:
```
((119390, 32), (119390, 28))
```

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

[0]:
```
((119390, 32), (217, 32))
```

[0]:
```
# Заполнение всех пропущенных значений нулями
# В данном случае это некорректно, так как нулями заполняются в том
  ↪числе категориальные колонки
data_new_3 = data.fillna(0)
data_new_3.head()
```

[0]:
```
          hotel  is_canceled  …  reservation_status
  ↪reservation_status_date
0  Resort Hotel            0  …           Check-Out
  ↪2015-07-01
1  Resort Hotel            0  …           Check-Out
  ↪2015-07-01
2  Resort Hotel            0  …           Check-Out
  ↪2015-07-02
3  Resort Hotel            0  …           Check-Out
  ↪2015-07-02
4  Resort Hotel            0  …           Check-Out
  ↪2015-07-03
```

```
[5 rows x 32 columns]
```

3. Импьютация

[0]:
```
# 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 children. Тип данных float64. Количество пустых значений 4, 0.0%.
Колонка agent. Тип данных float64. Количество пустых значений 16340, 13.
 ↪69%.
Колонка company. Тип данных float64. Количество пустых значений 112593,
 ↪94.31%.
```

[0]:
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

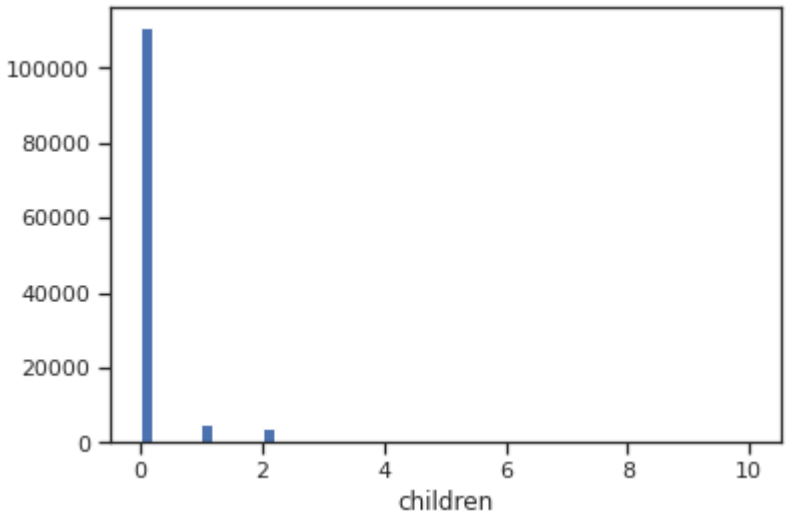
[0]:

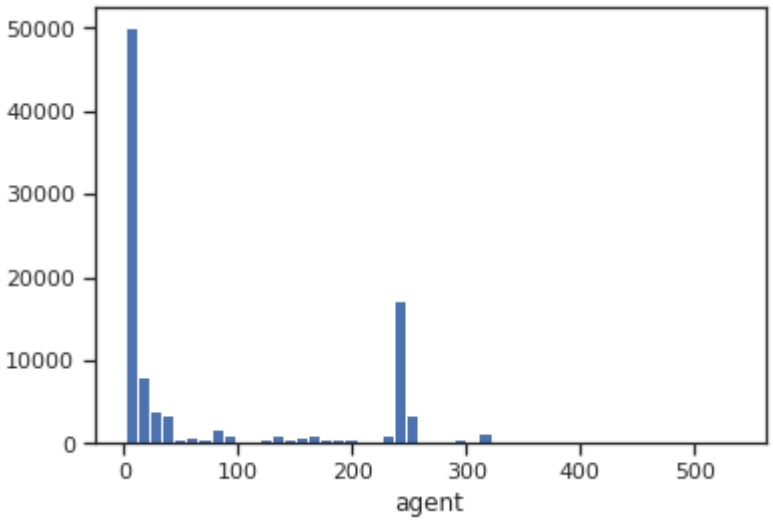
	children	agent	company
0	0.0	NaN	NaN
1	0.0	NaN	NaN
2	0.0	NaN	NaN
3	0.0	304.0	NaN
4	0.0	240.0	NaN
…	…	…	…
119385	0.0	394.0	NaN
119386	0.0	9.0	NaN
119387	0.0	9.0	NaN
119388	0.0	89.0	NaN
119389	0.0	9.0	NaN

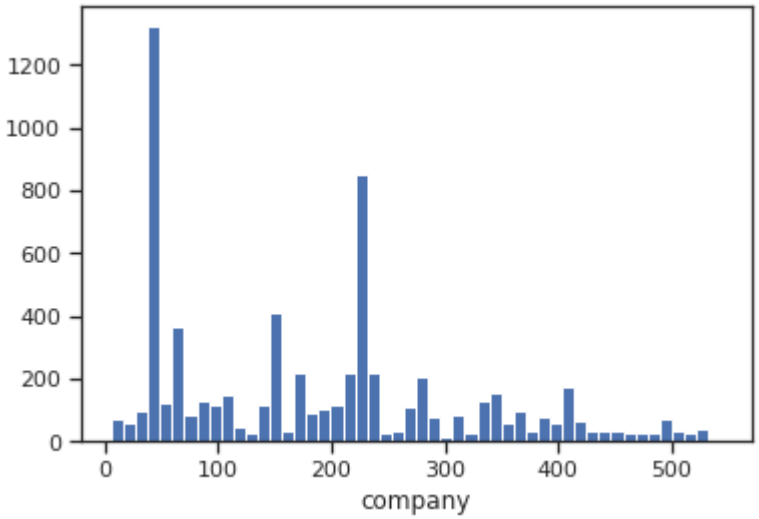
```
[119390 rows x 3 columns]
```

[0]:
```
# 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)
```







[0]:
```
# Фильтр по пустым значениям поля agent
data[data['agent'].isnull()]
```

[0]:
```
                hotel  is_canceled  …  reservation_status
reservation_status_date
0        Resort Hotel            0  …           Check-Out
2015-07-01
1        Resort Hotel            0  …           Check-Out
2015-07-01
2        Resort Hotel            0  …           Check-Out
2015-07-02
6        Resort Hotel            0  …           Check-Out
2015-07-03
18       Resort Hotel            0  …           Check-Out
2015-07-02
…                …           …  …                   …
…
119124     City Hotel            0  …           Check-Out
2017-08-30
119151     City Hotel            0  …           Check-Out
2017-08-30
119166     City Hotel            0  …           Check-Out
2017-08-31
119215     City Hotel            0  …           Check-Out
2017-09-01
119248     City Hotel            0  …           Check-Out
2017-09-01

[16340 rows x 32 columns]
```

[0]:
```
# Запоминаем индексы строк с пустыми значениями
flt_index = data[data['agent'].isnull()].index
flt_index
```

[0]:
```
Int64Index([     0,      1,      2,      6,     18,     30,     32,
            42,
                55,     56,
               …
               119117, 119118, 119119, 119122, 119123, 119124, 119151,
            119166,
               119215, 119248],
              dtype='int64', length=16340)
```

[0]:
```
# Проверяем что выводятся нужные строки
data[data.index.isin(flt_index)]
```

[0]:
```
                hotel  is_canceled  …  reservation_status
reservation_status_date
0        Resort Hotel            0  …           Check-Out
2015-07-01
1        Resort Hotel            0  …           Check-Out
2015-07-01
2        Resort Hotel            0  …           Check-Out
2015-07-02
6        Resort Hotel            0  …           Check-Out
2015-07-03
18       Resort Hotel            0  …           Check-Out
2015-07-02
…                …           …  …                   …
…
119124     City Hotel            0  …           Check-Out
2017-08-30
119151     City Hotel            0  …           Check-Out
2017-08-30
119166     City Hotel            0  …           Check-Out
2017-08-31
119215     City Hotel            0  …           Check-Out
2017-09-01
119248     City Hotel            0  …           Check-Out
2017-09-01

[16340 rows x 32 columns]
```

[0]:
```
# фильтр по колонке
data_num[data_num.index.isin(flt_index)]['agent']
```

[0]:
```
0         NaN
1         NaN
2         NaN
6         NaN
18        NaN
           ..
119124    NaN
119151    NaN
```

```
119166   NaN
119215   NaN
119248   NaN
Name: agent, Length: 16340, dtype: float64
```

[0]:
```
data_num_agent = data_num[['agent']]
data_num_agent.head()
```

[0]:
```
   agent
0    NaN
1    NaN
2    NaN
3  304.0
4  240.0
```

[0]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

[0]:
```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_agent)
mask_missing_values_only
```

[0]:
```
array([[ True],
       [ True],
       [ True],
       …,
       [False],
       [False],
       [False]])
```

[0]:
```
strategies=['mean', 'median','most_frequent']
```

[0]:
```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_agent)
    return data_num_imp[mask_missing_values_only]
```

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

[0]:
```
('mean', array([86.69338185, 86.69338185, 86.69338185, …, 86.69338185,
        86.69338185, 86.69338185]))
```

[0]:
```
strategies[1], test_num_impute(strategies[1])
```

[0]:
```
('median', array([14., 14., 14., …, 14., 14., 14.]))
```

[0]:
```
strategies[2], test_num_impute(strategies[2])
```

[0]:
```
('most_frequent', array([9., 9., 9., …, 9., 9., 9.]))
```

[0]:
```
# 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]
```

[0]:
```
data[['company']].describe()
```

[0]:
```
            company
count  6797.000000
mean    189.266735
std     131.655015
min       6.000000
25%      62.000000
50%     179.000000
75%     270.000000
max     543.000000
```

[0]:
```
test_num_impute_col(data, 'company', strategies[0])
```

[0]:
```
('company', 'mean', 112593, 189.26673532440782, 189.26673532440782)
```

[0]:
```
test_num_impute_col(data, 'company', strategies[1])
```

[0]:
```
('company', 'median', 112593, 179.0, 179.0)
```

[0]:
```
test_num_impute_col(data, 'company', strategies[2])
```

[0]:
```
('company', 'most_frequent', 112593, 40.0, 40.0)
```

4. Обработка пропусков в категориальных данных

[0]:
```
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_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=='object'):
```

```
        cat_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 country. Тип данных object. Количество пустых значений 488, 0.
 ↪41%.
```

[0]:
```
cat_temp_data = data[['country']]
cat_temp_data.head()
```

[0]:
```
  country
0     PRT
1     PRT
2     GBR
3     GBR
4     GBR
```

[0]:
```
cat_temp_data['country'].unique()
```

[0]:
```
array(['PRT', 'GBR', 'USA', 'ESP', 'IRL', 'FRA', nan, 'ROU', 'NOR',
 ↪'OMN',
       'ARG', 'POL', 'DEU', 'BEL', 'CHE', 'CN', 'GRC', 'ITA', 'NLD',
       'DNK', 'RUS', 'SWE', 'AUS', 'EST', 'CZE', 'BRA', 'FIN', 'MOZ',
       'BWA', 'LUX', 'SVN', 'ALB', 'IND', 'CHN', 'MEX', 'MAR', 'UKR',
       'SMR', 'LVA', 'PRI', 'SRB', 'CHL', 'AUT', 'BLR', 'LTU', 'TUR',
       'ZAF', 'AGO', 'ISR', 'CYM', 'ZMB', 'CPV', 'ZWE', 'DZA', 'KOR',
       'CRI', 'HUN', 'ARE', 'TUN', 'JAM', 'HRV', 'HKG', 'IRN', 'GEO',
       'AND', 'GIB', 'URY', 'JEY', 'CAF', 'CYP', 'COL', 'GGY', 'KWT',
       'NGA', 'MDV', 'VEN', 'SVK', 'FJI', 'KAZ', 'PAK', 'IDN', 'LBN',
       'PHL', 'SEN', 'SYC', 'AZE', 'BHR', 'NZL', 'THA', 'DOM', 'MKD',
       'MYS', 'ARM', 'JPN', 'LKA', 'CUB', 'CMR', 'BIH', 'MUS', 'COM',
       'SUR', 'UGA', 'BGR', 'CIV', 'JOR', 'SYR', 'SGP', 'BDI', 'SAU',
       'VNM', 'PLW', 'QAT', 'EGY', 'PER', 'MLT', 'MWI', 'ECU', 'MDG',
       'ISL', 'UZB', 'NPL', 'BHS', 'MAC', 'TGO', 'TWN', 'DJI', 'STP',
       'KNA', 'ETH', 'IRQ', 'HND', 'RWA', 'KHM', 'MCO', 'BGD', 'IMN',
       'TJK', 'NIC', 'BEN', 'VGB', 'TZA', 'GAB', 'GHA', 'TMP', 'GLP',
       'KEN', 'LIE', 'GNB', 'MNE', 'UMI', 'MYT', 'FRO', 'MMR', 'PAN',
       'BFA', 'LBY', 'MLI', 'NAM', 'BOL', 'PRY', 'BRB', 'ABW', 'AIA',
       'SLV', 'DMA', 'PYF', 'GUY', 'LCA', 'ATA', 'GTM', 'ASM', 'MRT',
       'NCL', 'KIR', 'SDN', 'ATF', 'SLE', 'LAO'], dtype=object)
```

[0]:
```
cat_temp_data[cat_temp_data['country'].isnull()].shape
```

[0]:
```
(488, 1)
```

[0]:
```
# Импьютация наиболее частыми значениями
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

```
[0]: array([['PRT'],
            ['PRT'],
            ['GBR'],
            …,
            ['DEU'],
            ['GBR'],
            ['DEU']], dtype=object)
```

```
[0]: # Пустые значения отсутствуют
     np.unique(data_imp2)
```

```
[0]: array(['ABW', 'AGO', 'AIA', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ASM',
            'ATA', 'ATF', 'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BFA',
            'BGD', 'BGR', 'BHR', 'BHS', 'BIH', 'BLR', 'BOL', 'BRA', 'BRB',
            'BWA', 'CAF', 'CHE', 'CHL', 'CHN', 'CIV', 'CMR', 'CN', 'COL',
            'COM', 'CPV', 'CRI', 'CUB', 'CYM', 'CYP', 'CZE', 'DEU', 'DJI',
            'DMA', 'DNK', 'DOM', 'DZA', 'ECU', 'EGY', 'ESP', 'EST', 'ETH',
            'FIN', 'FJI', 'FRA', 'FRO', 'GAB', 'GBR', 'GEO', 'GGY', 'GHA',
            'GIB', 'GLP', 'GNB', 'GRC', 'GTM', 'GUY', 'HKG', 'HND', 'HRV',
            'HUN', 'IDN', 'IMN', 'IND', 'IRL', 'IRN', 'IRQ', 'ISL', 'ISR',
            'ITA', 'JAM', 'JEY', 'JOR', 'JPN', 'KAZ', 'KEN', 'KHM', 'KIR',
            'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBY', 'LCA', 'LIE', 'LKA',
            'LTU', 'LUX', 'LVA', 'MAC', 'MAR', 'MCO', 'MDG', 'MDV', 'MEX',
            'MKD', 'MLI', 'MLT', 'MMR', 'MNE', 'MOZ', 'MRT', 'MUS', 'MWI',
            'MYS', 'MYT', 'NAM', 'NCL', 'NGA', 'NIC', 'NLD', 'NOR', 'NPL',
            'NZL', 'OMN', 'PAK', 'PAN', 'PER', 'PHL', 'PLW', 'POL', 'PRI',
            'PRT', 'PRY', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA', 'SAU', 'SDN',
            'SEN', 'SGP', 'SLE', 'SLV', 'SMR', 'SRB', 'STP', 'SUR', 'SVK',
            'SVN', 'SWE', 'SYC', 'SYR', 'TGO', 'THA', 'TJK', 'TMP', 'TUN',
            'TUR', 'TWN', 'TZA', 'UGA', 'UKR', 'UMI', 'URY', 'USA', 'UZB',
            'VEN', 'VGB', 'VNM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

```
[0]: # Импьютация константой
     imp3 = SimpleImputer(missing_values=np.nan, strategy='constant',
      ↪fill_value='!!!')
     data_imp3 = imp3.fit_transform(cat_temp_data)
     data_imp3
```

```
[0]: array([['PRT'],
            ['PRT'],
            ['GBR'],
            …,
            ['DEU'],
            ['GBR'],
            ['DEU']], dtype=object)
```

```
[0]: np.unique(data_imp3)
```

```
[0]: array(['!!!', 'ABW', 'AGO', 'AIA', 'ALB', 'AND', 'ARE', 'ARG', 'ARM',
            'ASM', 'ATA', 'ATF', 'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN',
            'BFA', 'BGD', 'BGR', 'BHR', 'BHS', 'BIH', 'BLR', 'BOL', 'BRA',
```

```
       'BRB', 'BWA', 'CAF', 'CHE', 'CHL', 'CHN', 'CIV', 'CMR', 'CN',
       'COL', 'COM', 'CPV', 'CRI', 'CUB', 'CYM', 'CYP', 'CZE', 'DEU',
       'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'ECU', 'EGY', 'ESP', 'EST',
       'ETH', 'FIN', 'FJI', 'FRA', 'FRO', 'GAB', 'GBR', 'GEO', 'GGY',
       'GHA', 'GIB', 'GLP', 'GNB', 'GRC', 'GTM', 'GUY', 'HKG', 'HND',
       'HRV', 'HUN', 'IDN', 'IMN', 'IND', 'IRL', 'IRN', 'IRQ', 'ISL',
       'ISR', 'ITA', 'JAM', 'JEY', 'JOR', 'JPN', 'KAZ', 'KEN', 'KHM',
       'KIR', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBY', 'LCA', 'LIE',
       'LKA', 'LTU', 'LUX', 'LVA', 'MAC', 'MAR', 'MCO', 'MDG', 'MDV',
       'MEX', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE', 'MOZ', 'MRT', 'MUS',
       'MWI', 'MYS', 'MYT', 'NAM', 'NCL', 'NGA', 'NIC', 'NLD', 'NOR',
       'NPL', 'NZL', 'OMN', 'PAK', 'PAN', 'PER', 'PHL', 'PLW', 'POL',
       'PRI', 'PRT', 'PRY', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA', 'SAU',
       'SDN', 'SEN', 'SGP', 'SLE', 'SLV', 'SMR', 'SRB', 'STP', 'SUR',
       'SVK', 'SVN', 'SWE', 'SYC', 'SYR', 'TGO', 'THA', 'TJK', 'TMP',
       'TUN', 'TUR', 'TWN', 'TZA', 'UGA', 'UKR', 'UMI', 'URY', 'USA',
       'UZB', 'VEN', 'VGB', 'VNM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

[0]:
```
data_imp3[data_imp3=='!!!'].size
```

[0]:
```
488
```

5. Кодирование категориальных признаков

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

[0]:
```
          c1
0        PRT
1        PRT
2        GBR
3        GBR
4        GBR
…        …
119385   BEL
119386   FRA
119387   DEU
119388   GBR
119389   DEU

[119390 rows x 1 columns]
```

6. Кодирование целочисленными значениями

[0]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

[0]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
```

[0]:
```
cat_enc['c1'].unique()
```

[0]:
```
array(['PRT', 'GBR', 'USA', 'ESP', 'IRL', 'FRA', 'ROU', 'NOR', 'OMN',
       'ARG', 'POL', 'DEU', 'BEL', 'CHE', 'CN', 'GRC', 'ITA', 'NLD',
       'DNK', 'RUS', 'SWE', 'AUS', 'EST', 'CZE', 'BRA', 'FIN', 'MOZ',
       'BWA', 'LUX', 'SVN', 'ALB', 'IND', 'CHN', 'MEX', 'MAR', 'UKR',
       'SMR', 'LVA', 'PRI', 'SRB', 'CHL', 'AUT', 'BLR', 'LTU', 'TUR',
       'ZAF', 'AGO', 'ISR', 'CYM', 'ZMB', 'CPV', 'ZWE', 'DZA', 'KOR',
       'CRI', 'HUN', 'ARE', 'TUN', 'JAM', 'HRV', 'HKG', 'IRN', 'GEO',
       'AND', 'GIB', 'URY', 'JEY', 'CAF', 'CYP', 'COL', 'GGY', 'KWT',
       'NGA', 'MDV', 'VEN', 'SVK', 'FJI', 'KAZ', 'PAK', 'IDN', 'LBN',
       'PHL', 'SEN', 'SYC', 'AZE', 'BHR', 'NZL', 'THA', 'DOM', 'MKD',
       'MYS', 'ARM', 'JPN', 'LKA', 'CUB', 'CMR', 'BIH', 'MUS', 'COM',
       'SUR', 'UGA', 'BGR', 'CIV', 'JOR', 'SYR', 'SGP', 'BDI', 'SAU',
       'VNM', 'PLW', 'QAT', 'EGY', 'PER', 'MLT', 'MWI', 'ECU', 'MDG',
       'ISL', 'UZB', 'NPL', 'BHS', 'MAC', 'TGO', 'TWN', 'DJI', 'STP',
       'KNA', 'ETH', 'IRQ', 'HND', 'RWA', 'KHM', 'MCO', 'BGD', 'IMN',
       'TJK', 'NIC', 'BEN', 'VGB', 'TZA', 'GAB', 'GHA', 'TMP', 'GLP',
       'KEN', 'LIE', 'GNB', 'MNE', 'UMI', 'MYT', 'FRO', 'MMR', 'PAN',
       'BFA', 'LBY', 'MLI', 'NAM', 'BOL', 'PRY', 'BRB', 'ABW', 'AIA',
       'SLV', 'DMA', 'PYF', 'GUY', 'LCA', 'ATA', 'GTM', 'ASM', 'MRT',
       'NCL', 'KIR', 'SDN', 'ATF', 'SLE', 'LAO'], dtype=object)
```

[0]:
```
np.unique(cat_enc_le)
```

[0]:
```
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,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176])
```

[0]:
```
le.inverse_transform([0, 1, 2, 3])
```

[0]:
```
array(['ABW', 'AGO', 'AIA', 'ALB'], dtype=object)
```

7. Кодирование категорий наборами бинарных значений

[0]:
```
ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(data[['meal']])
```

[0]:
```
data.shape
```

[0]:
```
(119390, 32)
```

[0]:
```
cat_enc_ohe.shape
```

[0]:
```
(119390, 5)
```

[0]:
```
cat_enc_ohe
```

[0]:
```
<119390x5 sparse matrix of type '<class 'numpy.float64'>'
        with 119390 stored elements in Compressed Sparse Row format>
```

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

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

[0]:
```
cat_enc.head(10)
```

[0]:

	c1
0	PRT
1	PRT
2	GBR
3	GBR
4	GBR
5	GBR
6	PRT
7	PRT
8	PRT
9	PRT

[0]:
```
cat_enc_ohe = ohe.fit_transform(data[['market_segment']])
```

[0]:
```
cat_enc_ohe.shape
```

[0]:
```
(119390, 8)
```

[0]:
```
cat_enc_ohe
```

```
[0]: <119390x8 sparse matrix of type '<class 'numpy.float64'>'
        with 119390 stored elements in Compressed Sparse Row format>
```

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

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

8. Pandas get dummies

```
[0]: pd.get_dummies(cat_enc).head()
```

```
[0]:    c1_ABW  c1_AGO  c1_AIA  c1_ALB  …  c1_VNM  c1_ZAF  c1_ZMB  c1_ZWE
    0       0       0       0       0  …       0       0       0       0
    1       0       0       0       0  …       0       0       0       0
    2       0       0       0       0  …       0       0       0       0
    3       0       0       0       0  …       0       0       0       0
    4       0       0       0       0  …       0       0       0       0

    [5 rows x 177 columns]
```

```
[0]: pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

```
[0]:    country_ABW  country_AGO  country_AIA  …  country_ZMB  country_ZWE
    country_nan
    0            0            0            0  …            0            0
    0
    1            0            0            0  …            0            0
    0
    2            0            0            0  …            0            0
    0
    3            0            0            0  …            0            0
    0
    4            0            0            0  …            0            0
    0

    [5 rows x 178 columns]
```

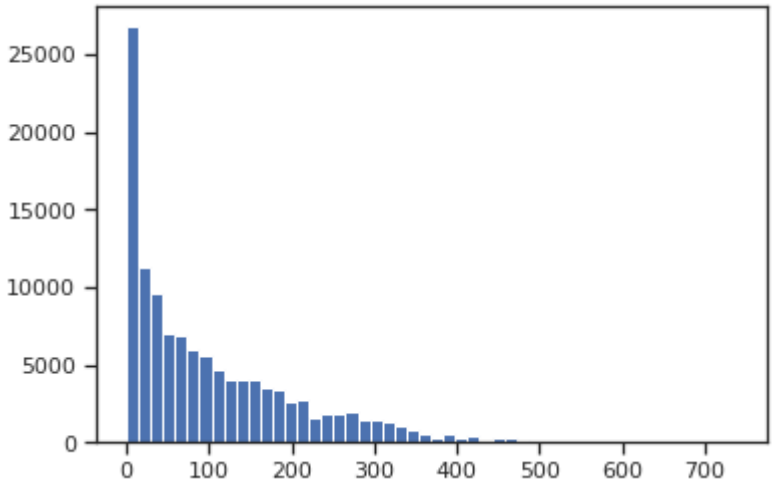
9. Масштабирование данных

[0]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
↪Normalizer
```

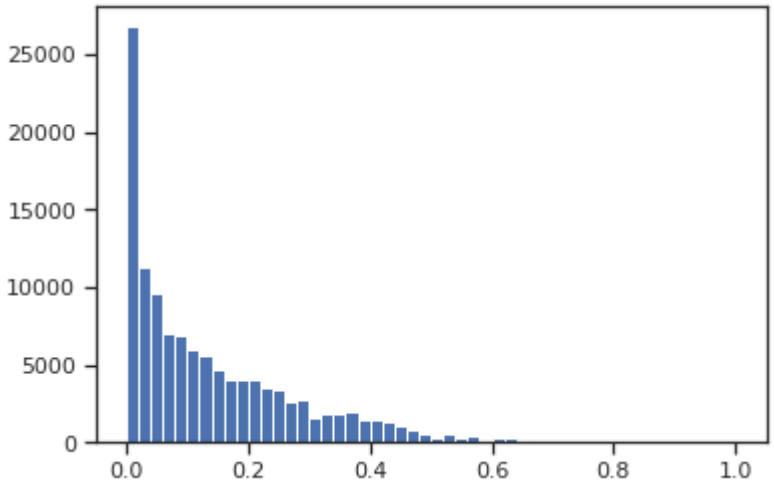
10. MinMax

[0]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['lead_time']])
```

[0]:
```
plt.hist(data['lead_time'], 50)
plt.show()
```



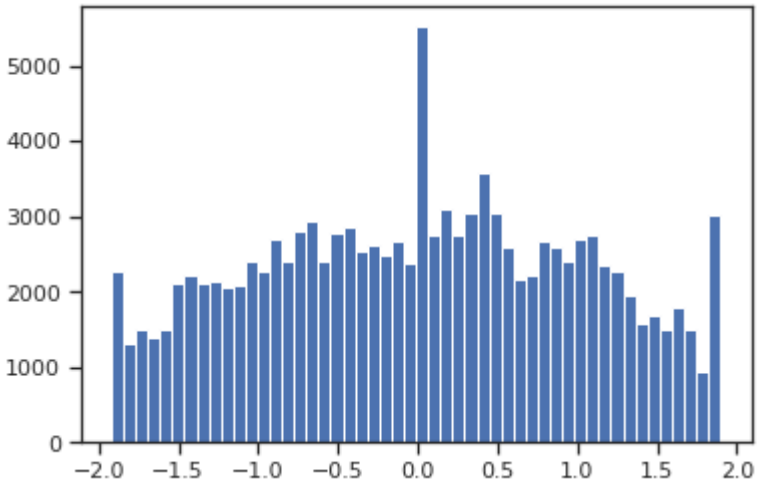
[0]:
```
plt.hist(sc1_data, 50)
plt.show()
```



11. Z-оценка

[0]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['arrival_date_week_number']])
```

[0]:
```
plt.hist(sc2_data, 50)
plt.show()
```



12. Нормализация

[0]:
```
sc3 = Normalizer()
sc3_data = sc3.fit_transform(data[['booking_changes']])
```

[0]:
```
plt.hist(sc3_data, 50)
plt.show()
```