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

Отчет по Рубежнему контролю № 1

«Методы обработки данных»

Вариант № 16

По курсу
“ Методы машинного обучения ”

Выполнил:
Али Диб А.Ж.
Студент группы ИУ5-22М

Москва, 2020

набор данных

Департамент здравоохранения разработал отчет о проверке и систему оценки. После пров
здравоохранения рассчитывает балл на основе наблюдаемых нарушений. Нарушения могу
риска: регистрируются конкретные нарушения, которые непосредственно связаны с перед
фальсификацией пищевых продуктов и загрязнением поверхностей, контактирующих с пи
умеренного риска: регистрирует конкретные нарушения, которые имеют умеренный риск в
здравоохранения и безопасности. Низкая степень риска: регистрирует нарушения, которые
имеют непосредственного риска для здоровья и безопасности населения. Счетная карточ
хранится в продовольственном учреждении и доступна для публика в этом наборе данных
построения модели машинного обучения, которая будет предсказывать, к какой категории
сначала посмотрим на наш набор данных и увидим некоторые его строки.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: Futur
  import pandas.util.testing as tm
```

```
data = pd.read_csv('restaurant-scores-lives-standard.csv', sep=",")
```

```
data.head()
```

	business_id	business_name	business_address	business_city	business_state
0	69618	Fancy Wheatfield Bakery	1362 Stockton St	San Francisco	CA
1	97975	BREADBELLY	1408 Clement St	San Francisco	CA
2	69487	Hakkasan San Francisco	1 Kearny St	San Francisco	CA
3	91044	Chopsticks Restaurant	4615 Mission St	San Francisco	CA
4	85987	Tselogs	552 Jones St	San Francisco	CA

```
data.shape
```

```
(5027, 17)
```

атрибуты набора данных с их описанием

1. **business_id**
2. **business_name**
3. **business_address**
4. **business_city**
5. **business_state**
6. **business_postal_code**
7. **business_latitude**
8. **business_longitude**
9. **business_location**
10. **business_phone_number**
11. **inspection_id**
12. **inspection_date**
13. **inspection_score**
14. **inspection_type**
15. **violation_id**
16. **violation_description**
17. **risk_category**

```
data.dtypes
```

```
business_id               int64
business_name            object
business_address         object
business_city            object
business_state           object
business_postal_code     object
business_latitude       float64
business_longitude      float64
business_location        object
business_phone_number   float64
inspection_id            object
inspection_date          object
inspection_score        float64
inspection_type          object
violation_id             object
violation_description    object
risk_category            object
dtype: object
```

1. Обработка пропусков в данных

1.1. Обработка пропусков в числовых данных

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

```
business_id                0
business_name              0
business_address           0
business_city              0
business_state             0
business_postal_code      90
business_latitude       3743
business_longitude      3743
business_location       3743
business_phone_number   3209
inspection_id              0
inspection_date            0
inspection_score        1764
inspection_type            0
violation_id            1520
violation_description   1521
risk_category           1521
dtype: int64
```

Мы видим, что девять из наших столбцов имеют нулевое значение.

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

```
Всего строк: 5027
```

```
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

```
((5027, 17), (5027, 8))
```

```
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

```
((5027, 17), (220, 17))
```

Если мы удалили столбцы, содержащие нулевое значение, мы получим 8 вместо 17.
Если мы удалили строку, содержащую нулевые значения, мы получим 5711 вместо 53973 с

Процент пустых значений в девяти столбцах:

```
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:
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col
```

```
Колонка business_postal_code. Тип данных object. Количество пустых значений 90, 1.79
Колонка business_latitude. Тип данных float64. Количество пустых значений 3743, 74.4
Колонка business_longitude. Тип данных float64. Количество пустых значений 3743, 74.
Колонка business_location. Тип данных object. Количество пустых значений 3743, 74.46
Колонка business_phone_number. Тип данных float64. Количество пустых значений 3209,
Колонка inspection_score. Тип данных float64. Количество пустых значений 1764, 35.09
Колонка violation_id. Тип данных object. Количество пустых значений 1520, 30.24%.
Колонка violation_description. Тип данных object. Количество пустых значений 1521, 3
Колонка risk_category. Тип данных object. Количество пустых значений 1521, 30.26%.
```

```
data_num = data[num_cols]
data_num
```

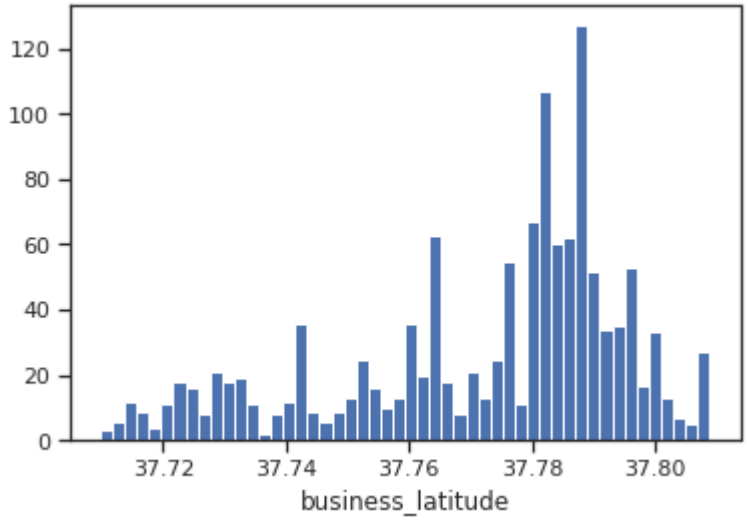
	business_postal_code	business_latitude	business_longitude	business_lo
0	94133	NaN	NaN	
1	94118	NaN	NaN	
2	94108	NaN	NaN	
3	94112	NaN	NaN	
4	94102	NaN	NaN	
...	...	...	...	
5022	94108	NaN	NaN	
5023	94109	NaN	NaN	
5024	94114	NaN	NaN	
5025	94133	37.797378	-122.403344	{'type' 'coord [-122.40
5026	94122	NaN	NaN	

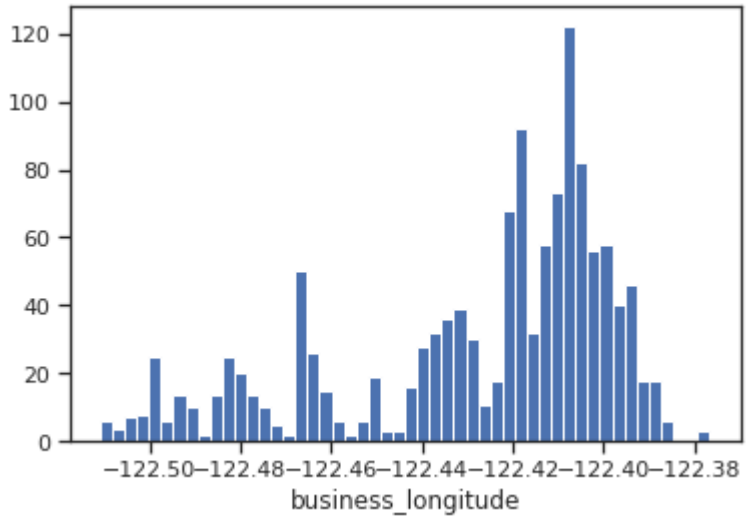
5027 rows × 9 columns

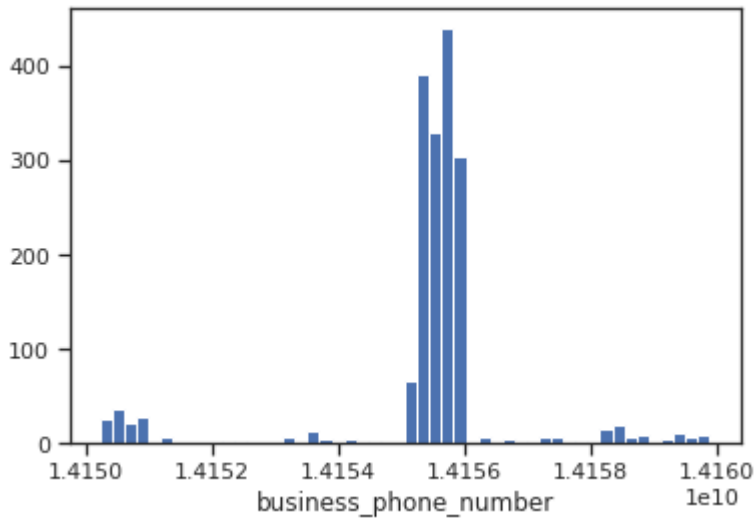
гистограмма для числовых данных

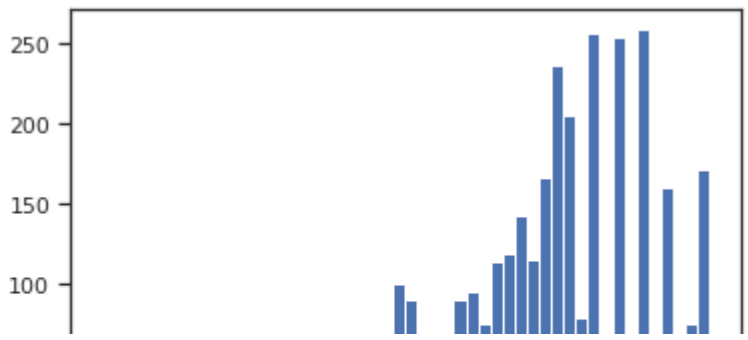
```
for col in data_num:
    if(data[col].dtype!='O'):
      plt.hist(data[col],50)
      plt.xlabel(col)
      plt.show()
```

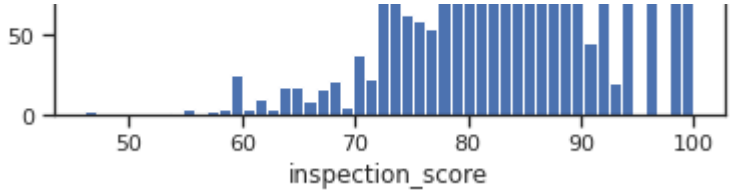
```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:839: RuntimeWar
  keep = (tmp_a >= first_edge)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:840: RuntimeWar
  keep &= (tmp_a <= last_edge)
```











Мы знаем, что столбец postal_code должен быть уникальным для каждого здания, и мы мо
поэтому мы отбросим этот столбец.

Также в столбцах (business_latitude, business_longitude, business_location) мы не можем пре
мы заполняем их нулевыми значениями. Все столбцы (postal_code, business_latitude, busine
такого значения, поскольку столбец business_address не содержит нулевого значения, поэт
анализе данных вместо этих столбцов.

```
data_num=data_num.drop(columns='business_postal_code')
temp=data_num['business_latitude'].fillna(0)
temp1=data_num['business_longitude'].fillna(0)
temp2=data_num['business_location'].fillna('---')

data_num[['business_latitude','business_longitude','business_location']]=pd.DataFra

data_num
```

```
business_latitude  business_longitude  business_location  business_phone
```

```
# Более сложная функция, которая позволяет задавать колонку и вид импьютации
def test_num_impute_col(dataset, column, strategy_param,fillvalue=None):
    temp_data = dataset[[column]]

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

    imp_num = SimpleImputer(missing_values=np.nan,strategy=strategy_param,fill_valu
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]
    data_num[[column]]=data_num_imp

    return column, strategy_param, filled_data.size, filled_data[0], filled_data[fi
```

```
5022              0.000000              0.000000                    ---
```

```
dic={}
for i in data['business_name'].index:
  if ~(data.loc[i,'business_phone_number'] != data.loc[i,'business_phone_number']):
    dic[data.loc[i,'business_name']]=data.loc[i,'business_phone_number']
```

чтобы заполнить пустые значения столбца business_phone_number, мы сначала заполним е
доступен в другой строке, и, если он недоступен, мы заполним его нулем.

```
5006              0.000000              0.000000                                  1.415
```

```
for i in data['business_name'].index:
  if (data.loc[i,'business_phone_number'] != data.loc[i,'business_phone_number']):
    if data.loc[i,'business_name'] in dic:
      data_num.loc[i,'business_phone_number']=dic[data.loc[i,'business_name']]

test_num_impute_col(data, 'business_phone_number', strategy_param='constant',fillva
```

```
('business_phone_number', 'constant', 3209, 0.0, 0.0)
```

```
test_num_impute_col(data, 'inspection_score', strategy_param='mean')
```

```
('inspection_score', 'mean', 1764, 85.49310450505669, 85.49310450505669)
```

1.2. Обработка пропусков в категориальных данных

5 cells hidden

2. Преобразование категориальных признаков в числовые

В нашем наборе данных у нас есть 11 атрибутов типа объекта, которые необходимо преобр

```
for column in data:
  if (data[column].dtype=='O'):
    print(column,len(data[column].unique()))
```

```
business_name 2276
business_address 2253
business_city 1
business_state 1
business_postal_code 43
business_location 615
inspection_id 3415
inspection_date 627
inspection_type 11
violation_id 3505
violation_description 61
risk_category 4
```

Мы будем использовать LabelEncoder со столбцами, которые имеют много значений, таких
business_location, inspection_id, violation_id.
С business_city, business_state, inspection_date, inspection_type, violation_description, risk_categ
мы будем использовать Кодирование категорий наборами бинарных значений - one-hot enc

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder


le = LabelEncoder()
name_le = le.fit_transform(data['business_name'])
np.unique(name_le)
```

```
array([   0,    1,    2, ..., 2273, 2274, 2275])
```

```
address_le = le.fit_transform(data['business_address'])
np.unique(address_le)
```

```
array([   0,    1,    2, ..., 2250, 2251, 2252])
```

```
location_le = le.fit_transform(data_num['business_location'])
np.unique(location_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,  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, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
       273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
       286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
       312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
       325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
       338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350,
       351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
       364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,
       377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389,
       390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402,
       403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415,
       416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428,
       429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441,
       442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
       455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467,
       468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480,
       481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
       494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506,
       507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,
       520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532,
       533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545,
       546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558,
       559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571,
       572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584,
       585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597,
       598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610,
       611, 612, 613, 614])
```

```
inspection_le = le.fit_transform(data['inspection_id'])
np.unique(inspection_le)
```

```
array([   0,    1,    2, ..., 3412, 3413, 3414])
```

```
violation_le = le.fit_transform(data_num['violation_id'])
np.unique(violation_le)
```

```
ohe = OneHotEncoder()
city_ohe = ohe.fit_transform(data[['business_city']])
print(city_ohe.shape)
city_ohe.toarray()[0:3]
```

```
(5027, 1)
array([[1.],
       [1.],
       [1.]])
```

```
state_ohe = ohe.fit_transform(data[['business_state']])
print(state_ohe.shape)
state_ohe.toarray()[0:3]
```

```
(5027, 1)
array([[1.],
       [1.],
       [1.]])
```

```
inspection_date_ohe = ohe.fit_transform(data[['inspection_date']])
print(inspection_date_ohe.shape)
inspection_date_ohe.toarray()[0:3]
```

```
(5027, 627)
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
inspection_type_ohe = ohe.fit_transform(data[['inspection_type']])
print(inspection_type_ohe.shape)
inspection_type_ohe.toarray()[0:3]
```

```
(5027, 11)
array([[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., 1., 0.]])
```

```
violation_description_ohe = ohe.fit_transform(data_num[['violation_description']])
print(violation_description_ohe.shape)
violation_description_ohe.toarray()[0:3]
```

```
(5027, 60)
array([[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., 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., 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.]])
```

```
risk_category_ohe = ohe.fit_transform(data_num[['risk_category']])
print(risk_category_ohe.shape)
risk_category_ohe.toarray()[0:3]
```

```
(5027, 3)
array([[0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]])
```

Теперь у нас есть все категориальные данные как числовые после того, как мы их закодир
алгоритмы машинного обучения, которые принимают только числовые данные.

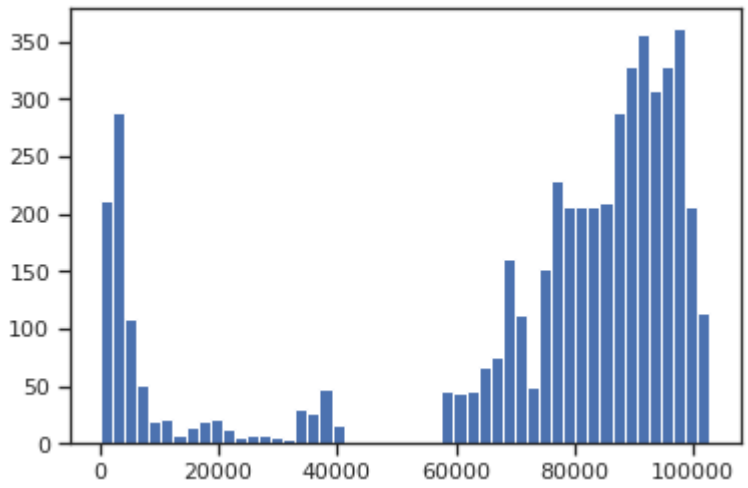
3. Масштабирование данных

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

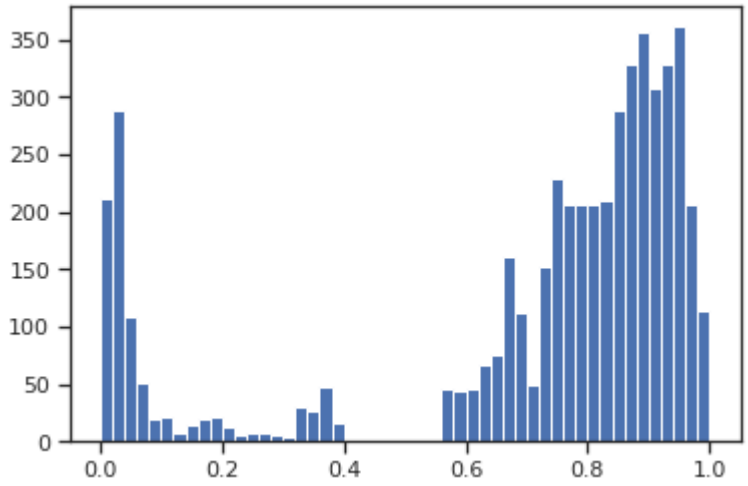
3.1. MinMax масштабирование

```
mm = MinMaxScaler()
business_id_mm = mm.fit_transform(data[['business_id']])
```

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




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

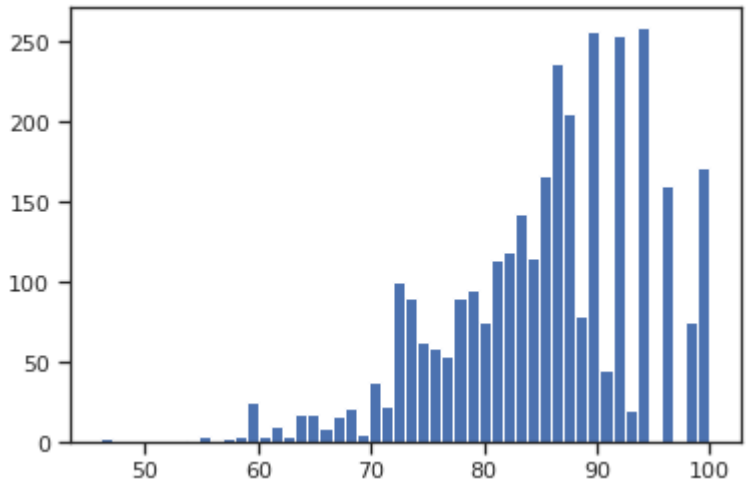


3.2. Масштабирование данных на основе Z-оценки - StandardScaler

```
sc = StandardScaler()
inspection_score_sc = sc.fit_transform(data[['inspection_score']])
```

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

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:839: RuntimeWar
  keep = (tmp_a >= first_edge)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:840: RuntimeWar
  keep &= (tmp_a <= last_edge)
```




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

```
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:839: RuntimeWar
  keep = (tmp_a >= first_edge)
/usr/local/lib/python3.6/dist-packages/numpy/lib/histograms.py:840: RuntimeWar
  keep &= (tmp_a <= last_edge)
```