**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:

Хотин П.Ю.

ИУ5-24М

Москва, 2020 год

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

data = pd.read\_csv('data/restaurant-scores-lives-standard.csv', sep=",")

data.shape

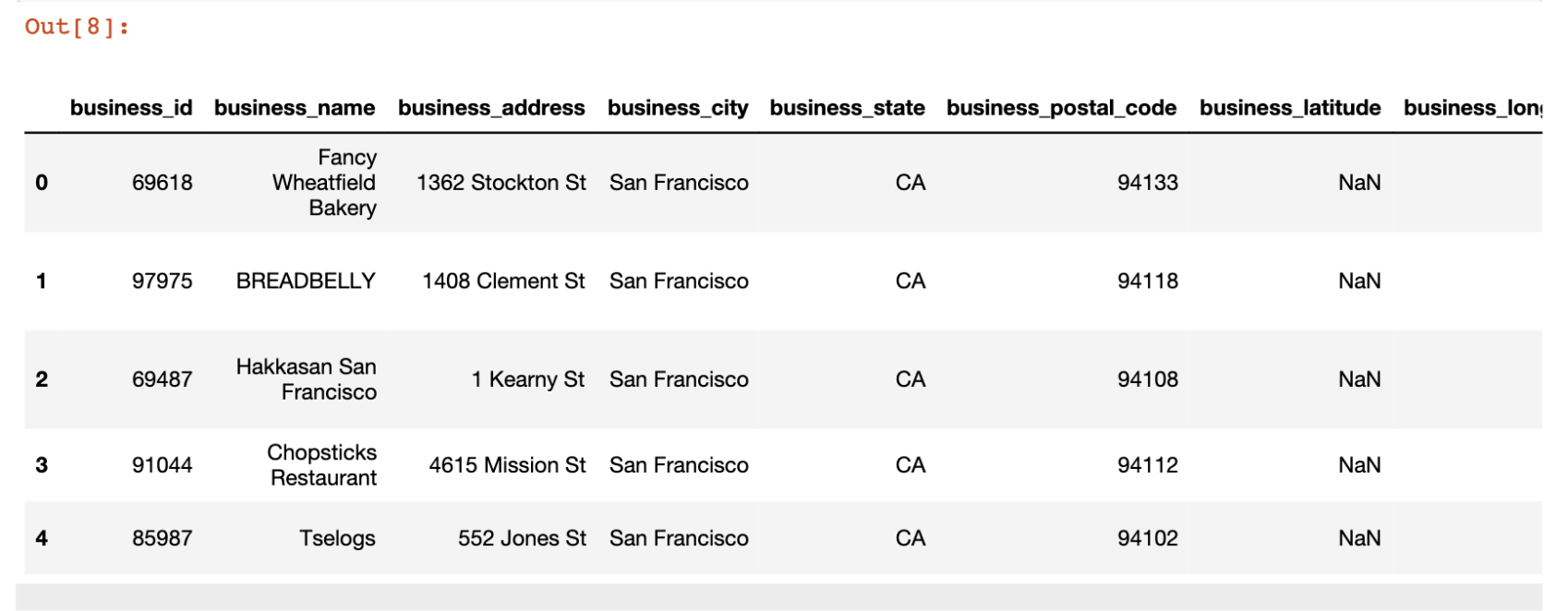
(53973, 17)

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

data.isnull().sum()

business\_id 0  
business\_name 0  
business\_address 0  
business\_city 0  
business\_state 0  
business\_postal\_code 1083  
business\_latitude 24095  
business\_longitude 24095  
business\_location 24095  
business\_phone\_number 36539  
inspection\_id 0  
inspection\_date 0  
inspection\_score 14114  
inspection\_type 0  
violation\_id 13462  
violation\_description 13462  
risk\_category 13462  
dtype: int64

data.head()

1 Обработка пропусков в данных

# Удаление колонок, содержащих пустые значения  
data\_new\_1 = data.dropna(axis=1, how='any')  
(data.shape, data\_new\_1.shape)  
print(f'Удалено колонок: {data.shape[1] - data\_new\_1.shape[1]}')

Удалено колонок: 9

# Удаление строк, содержащих пустые значения  
data\_new\_2 = data.dropna(axis=0, how='any')  
(data.shape, data\_new\_2.shape)  
print(f'Удалено строк: {data.shape[0] - data\_new\_2.shape[0]}')

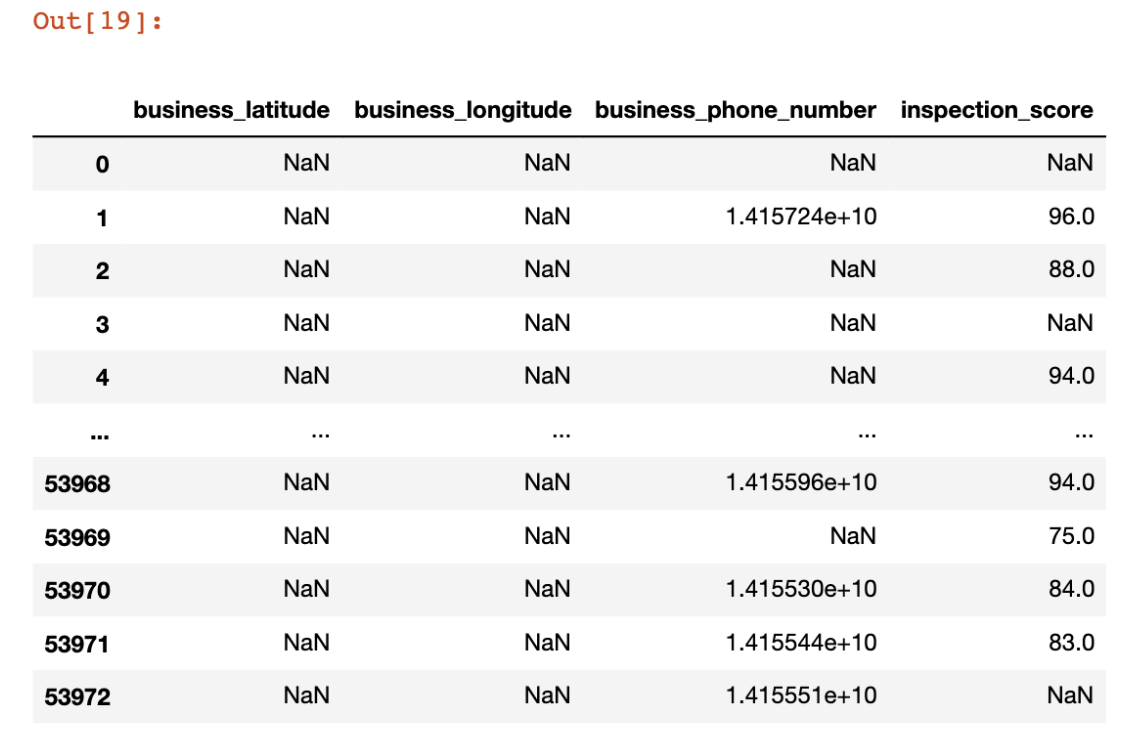
Удалено строк: 48262

1.1 Обработка пропусков в числовых данных

rows\_count = data.shape[0]  
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 / rows\_count) \* 100.0, 2)  
 print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp\_null\_count, temp\_perc))

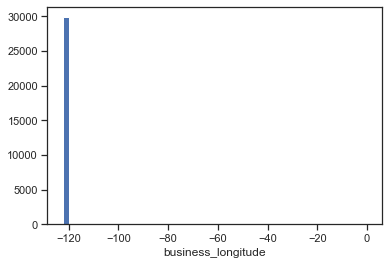
Колонка business\_latitude. Тип данных float64. Количество пустых значений 24095, 44.64%.  
Колонка business\_longitude. Тип данных float64. Количество пустых значений 24095, 44.64%.  
Колонка business\_phone\_number. Тип данных float64. Количество пустых значений 36539, 67.7%.  
Колонка inspection\_score. Тип данных float64. Количество пустых значений 14114, 26.15%.

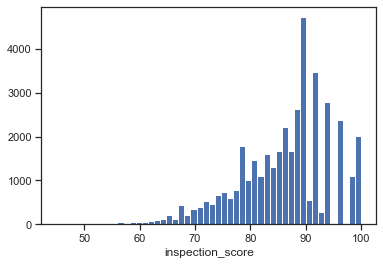
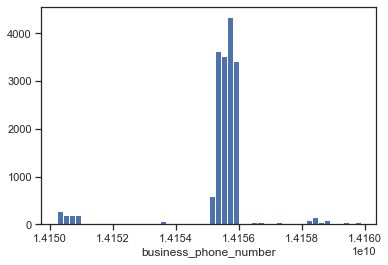
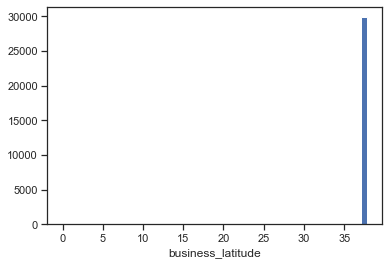
# Фильтр по колонкам с пропущенными значениями  
data\_num = data[num\_cols]  
data\_num



53973 rows × 4 columns

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



# Фильтр по пустым значениям поля inspection\_score   
data[data['inspection\_score'].isnull()]

14114 rows × 17 columns

# Запоминаем индексы строк с пустыми значениями  
flt\_index = data[data['inspection\_score'].isnull()].index  
flt\_index

Int64Index([ 0, 3, 5, 6, 7, 10, 11, 13, 14,  
 15,  
 ...  
 53932, 53936, 53941, 53945, 53950, 53955, 53957, 53958, 53961,  
 53972],  
 dtype='int64', length=14114)

# Проверяем что выводятся нужные строки  
data[data.index.isin(flt\_index)]

14114 rows × 17 columns

# фильтр по колонке  
data\_num[data\_num.index.isin(flt\_index)]['inspection\_score']

0 NaN  
3 NaN  
5 NaN  
6 NaN  
7 NaN  
 ..  
53955 NaN  
53957 NaN  
53958 NaN  
53961 NaN  
53972 NaN  
Name: inspection\_score, Length: 14114, dtype: float64

data\_num\_inspection\_score = data\_num[['inspection\_score']]  
data\_num\_inspection\_score.head()

inspection\_score

0 NaN

1 96.0

2 88.0

3 NaN

4 94.0

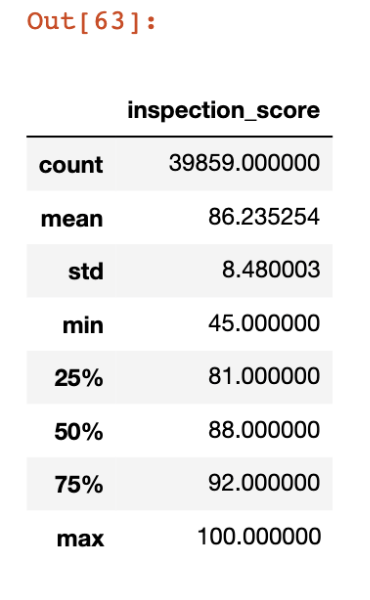
from sklearn.impute import SimpleImputer  
from sklearn.impute import MissingIndicator

# Фильтр для проверки заполнения пустых значений  
indicator = MissingIndicator()  
mask\_missing\_values\_only = indicator.fit\_transform(data\_num\_inspection\_score)  
mask\_missing\_values\_only

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

strategies=['mean', 'median','most\_frequent']

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[['inspection\_score']].describe()

for strategy in strategies:  
 print(test\_num\_impute\_col(data, 'inspection\_score', strategy))  
 print(test\_num\_impute\_col(data, 'business\_latitude', strategy))  
 print(test\_num\_impute\_col(data, 'business\_longitude', strategy), end='\n\n')

('inspection\_score', 'mean', 14114, 86.23525427130636, 86.23525427130636)  
('business\_latitude', 'mean', 24095, 37.7552651997791, 37.7552651997791)  
('business\_longitude', 'mean', 24095, -122.37375472595221, -122.37375472595221)  
  
('inspection\_score', 'median', 14114, 88.0, 88.0)  
('business\_latitude', 'median', 24095, 37.780174, 37.780174)  
('business\_longitude', 'median', 24095, -122.41913600000001, -122.41913600000001)  
  
('inspection\_score', 'most\_frequent', 14114, 90.0, 90.0)  
('business\_latitude', 'most\_frequent', 24095, 37.808240000000005, 37.808240000000005)  
('business\_longitude', 'most\_frequent', 24095, -122.41018899999999, -122.41018899999999)

1.2 Обработка пропусков в категориальных данных

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 / rows\_count) \* 100.0, 2)  
 print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp\_null\_count, temp\_perc))

Колонка business\_postal\_code. Тип данных object. Количество пустых значений 1083, 2.01%.  
Колонка business\_location. Тип данных object. Количество пустых значений 24095, 44.64%.  
Колонка violation\_id. Тип данных object. Количество пустых значений 13462, 24.94%.  
Колонка violation\_description. Тип данных object. Количество пустых значений 13462, 24.94%.  
Колонка risk\_category. Тип данных object. Количество пустых значений 13462, 24.94%.

cat\_temp\_data = data[['risk\_category']]

cat\_temp\_data['risk\_category'].unique()

array(['Moderate Risk', nan, 'Low Risk', 'High Risk'], dtype=object)

cat\_temp\_data[cat\_temp\_data['risk\_category'].isnull()].shape

(13462, 1)

# Импьютация наиболее частыми значениями  
imp2 = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')  
data\_imp2 = imp2.fit\_transform(cat\_temp\_data)  
data\_imp2

array([['Moderate Risk'],  
 ['Moderate Risk'],  
 ['Moderate Risk'],  
 ...,  
 ['Moderate Risk'],  
 ['High Risk'],  
 ['Low Risk']], dtype=object)

# Пустые значения отсутствуют  
np.unique(data\_imp2)

array(['High Risk', 'Low Risk', 'Moderate Risk'], dtype=object)

# Импьютация константой  
imp3 = SimpleImputer(missing\_values=np.nan, strategy='constant', fill\_value='Unknown')  
data\_imp3 = imp3.fit\_transform(cat\_temp\_data)  
data\_imp3

array([['Moderate Risk'],  
 ['Moderate Risk'],  
 ['Moderate Risk'],  
 ...,  
 ['Moderate Risk'],  
 ['High Risk'],  
 ['Unknown']], dtype=object)

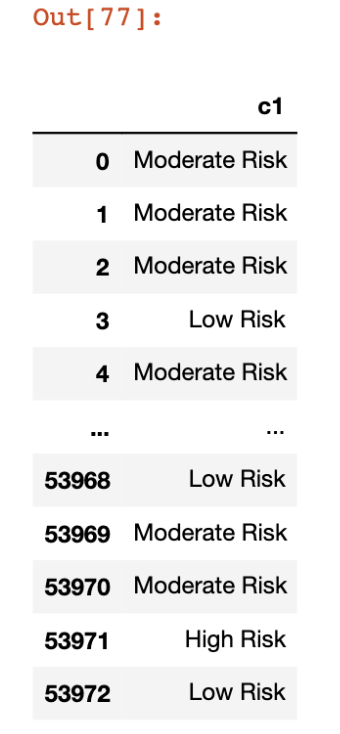
np.unique(data\_imp3)

array(['High Risk', 'Low Risk', 'Moderate Risk', 'Unknown'], dtype=object)

data\_imp3[data\_imp3=='Unknown'].size

13462

2 Преобразование категориальных признаков в числовые

cat\_enc = pd.DataFrame({'c1':data\_imp2.T[0]})  
cat\_enc

53973 rows × 1 columns

2.1 Кодирование категорий целочисленными значениями

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

le = LabelEncoder()  
cat\_enc\_le = le.fit\_transform(cat\_enc['c1'])  
cat\_enc['c1'].unique()

array(['Moderate Risk', 'Low Risk', 'High Risk'], dtype=object)

np.unique(cat\_enc\_le)

array([0, 1, 2])

le.inverse\_transform([0, 1, 2])

array(['High Risk', 'Low Risk', 'Moderate Risk'], dtype=object)

2.2 Кодирование категорий наборами бинарных значений

ohe = OneHotEncoder()  
cat\_enc\_ohe = ohe.fit\_transform(cat\_enc[['c1']])  
cat\_enc.shape

(53973, 1)

cat\_enc\_ohe.shape

(53973, 3)

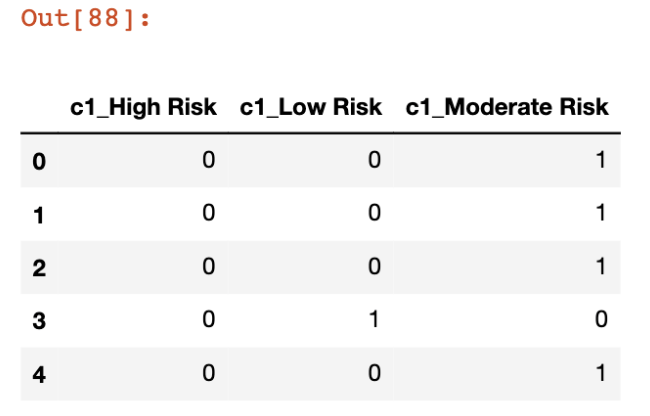
cat\_enc\_ohe

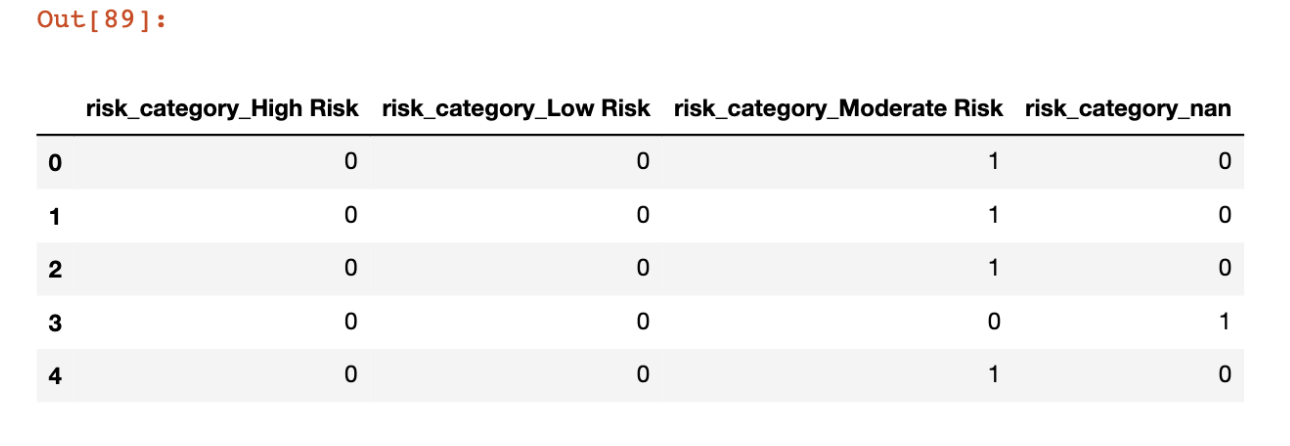
<53973x3 sparse matrix of type '<class 'numpy.float64'>'  
 with 53973 stored elements in Compressed Sparse Row format>

cat\_enc\_ohe.todense()[0:10]

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

2.3 Быстрый вариант one-hot кодирования

pd.get\_dummies(cat\_enc).head()

pd.get\_dummies(cat\_temp\_data, dummy\_na=True).head()

3 Масштабирование данных

#### Заменяю пропуски в колонке 'inspection\_score' исходного датасета, чтобы использовать ее для масштабирования/нормализации

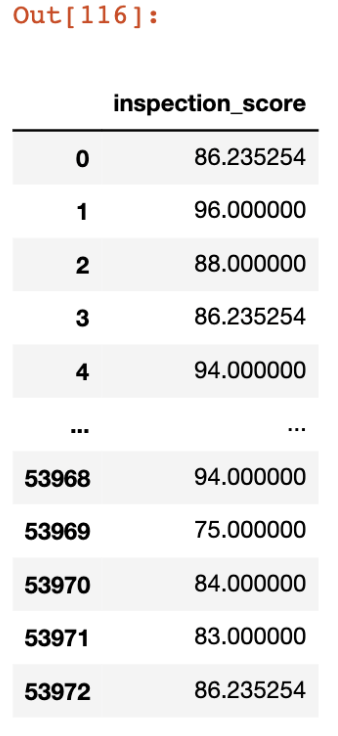
def num\_impute\_col(dataset, column, strategy\_param):  
 temp\_data = dataset[[column]]  
   
 indicator = MissingIndicator()  
 mask\_missing\_values\_only = indicator.fit\_transform(temp\_data)  
   
 imp\_num = SimpleImputer(strategy=strategy\_param)  
 data\_num\_imp = imp\_num.fit\_transform(temp\_data)  
# data\_num\_imp = imp\_num.fit\_transform(dataset[[column]])  
   
 filled\_data = data\_num\_imp[mask\_missing\_values\_only]  
   
 new\_data = imp\_num.transform(dataset[[column]])  
   
 return new\_data

num\_impute\_col(data, 'inspection\_score', 'mean')

array([[86.23525427],  
 [96. ],  
 [88. ],  
 ...,  
 [84. ],  
 [83. ],  
 [86.23525427]])

data[['inspection\_score']] = num\_impute\_col(data, 'inspection\_score', 'mean')

data[['inspection\_score']]



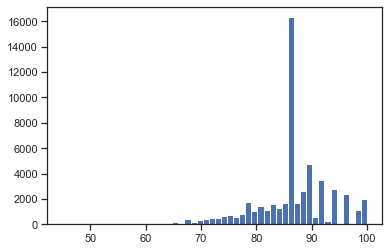
53973 rows × 1 columns

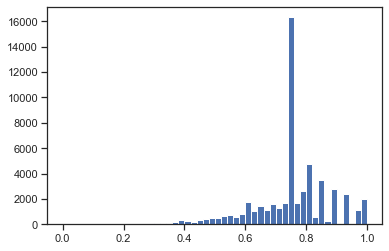
3.1 MinMax масштабирование

from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

sc1 = MinMaxScaler()  
sc1\_data = sc1.fit\_transform(data[['inspection\_score']])

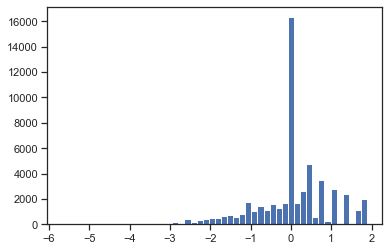
plt.hist(data['inspection\_score'], 50)  
plt.show()



plt.hist(sc1\_data, 50)  
plt.show()

3.2 Масштабирование данных на основе Z-оценки

sc2 = StandardScaler()  
sc2\_data = sc2.fit\_transform(data[['inspection\_score']])

plt.hist(sc2\_data, 50)  
plt.show()

3.3. Нормализация данных

sc3 = Normalizer()  
sc3\_data = sc3.fit\_transform(data[['inspection\_score']])

plt.hist(sc3\_data, 50)  
plt.show()