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 ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №2

Выполнила:

студент группы ИУ5-63Б

Латыпова К.Н.

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2020 г.

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

Текст программы и экранные формы:

ЛР №2

Используются данные country vaccinations

В [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Загрузка и первичный анализ данных

В [14]:
```
data = pd.read_csv('country_vaccinations.csv', sep=",")
```

В [15]:
```
# размер набора данных
data.shape
```

Out[15]:
```
(3555, 15)
```

В [16]:
```
# типы колонок
data.dtypes
```

Out[16]:
```
country                              object
iso_code                             object
date                                 object
total_vaccinations                  float64
people_vaccinated                   float64
people_fully_vaccinated             float64
daily_vaccinations_raw              float64
daily_vaccinations                  float64
total_vaccinations_per_hundred      float64
people_vaccinated_per_hundred       float64
people_fully_vaccinated_per_hundred float64
daily_vaccinations_per_million      float64
vaccines                             object
source_name                          object
source_website                       object
dtype: object
```

В [17]:
```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

Out[17]:
```
country                                0
iso_code                             272
date                                   0
total_vaccinations                  1214
people_vaccinated                   1615
people_fully_vaccinated             2277
daily_vaccinations_raw              1583
daily_vaccinations                   135
total_vaccinations_per_hundred      1214
people_vaccinated_per_hundred       1615
people_fully_vaccinated_per_hundred 2277
daily_vaccinations_per_million       135
vaccines                               0
source_name                            0
source_website                         0
dtype: int64
```

В [18]:
```
# Первые 5 строк датасета
data.head()
```

Out[18]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_h
0	Albania	ALB	2021-01-10	0.0	0.0	NaN	NaN	NaN	
1	Albania	ALB	2021-01-11	NaN	NaN	NaN	NaN	64.0	
2	Albania	ALB	2021-01-12	128.0	128.0	NaN	NaN	64.0	
3	Albania	ALB	2021-01-13	188.0	188.0	NaN	60.0	63.0	
4	Albania	ALB	2021-01-14	266.0	266.0	NaN	78.0	66.0	

В [19]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 3555
```

Обработка пропусков в данных

В [20]:
```
# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[20]: ((3555, 15), (3555, 5))

В [21]:
```
# Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

Out[21]: ((3555, 15), (995, 15))

В [22]:
```
data.head()
```

Out[22]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_h
0	Albania	ALB	2021-01-10	0.0	0.0	NaN	NaN	NaN	
1	Albania	ALB	2021-01-11	NaN	NaN	NaN	NaN	64.0	
2	Albania	ALB	2021-01-12	128.0	128.0	NaN	NaN	64.0	
3	Albania	ALB	2021-01-13	188.0	188.0	NaN	60.0	63.0	
4	Albania	ALB	2021-01-14	266.0	266.0	NaN	78.0	66.0	

В [23]:
```
# 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()
```

Out[23]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_h
0	Albania	ALB	2021-01-10	0.0	0.0	0.0	0.0	0.0	
1	Albania	ALB	2021-01-11	0.0	0.0	0.0	0.0	64.0	
2	Albania	ALB	2021-01-12	128.0	128.0	0.0	0.0	64.0	
3	Albania	ALB	2021-01-13	188.0	188.0	0.0	60.0	63.0	
4	Albania	ALB	2021-01-14	266.0	266.0	0.0	78.0	66.0	

"Внедрение значений" - импьютация (imputation)

Обработка пропусков в числовых данных

В [24]:
```
# 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 total_vaccinations. Тип данных float64. Количество пустых значений 1214, 34.15%.
Колонка people_vaccinated. Тип данных float64. Количество пустых значений 1615, 45.43%.
Колонка people_fully_vaccinated. Тип данных float64. Количество пустых значений 2277, 64.05%.
Колонка daily_vaccinations_raw. Тип данных float64. Количество пустых значений 1583, 44.53%.
Колонка daily_vaccinations. Тип данных float64. Количество пустых значений 135, 3.8%.
Колонка total_vaccinations_per_hundred. Тип данных float64. Количество пустых значений 1214, 34.15%.
Колонка people_vaccinated_per_hundred. Тип данных float64. Количество пустых значений 1615, 45.43%.
Колонка people_fully_vaccinated_per_hundred. Тип данных float64. Количество пустых значений 2277, 64.05%.
Колонка daily_vaccinations_per_million. Тип данных float64. Количество пустых значений 135, 3.8%.
```

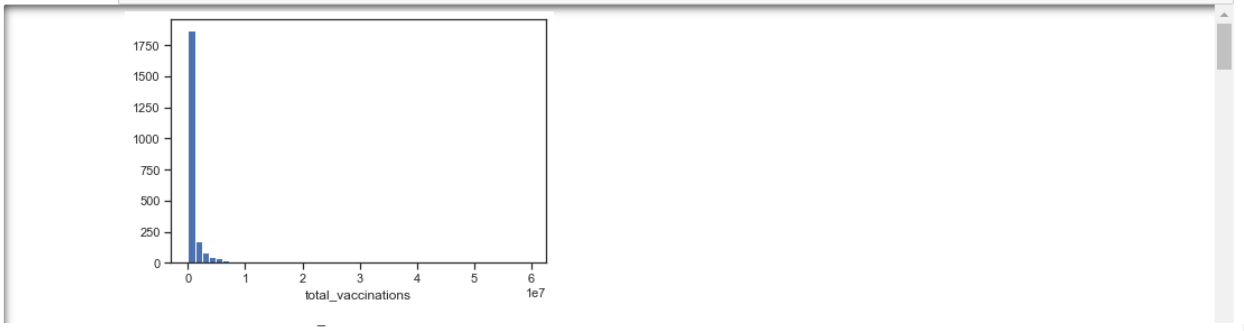
В [25]:
```
# Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

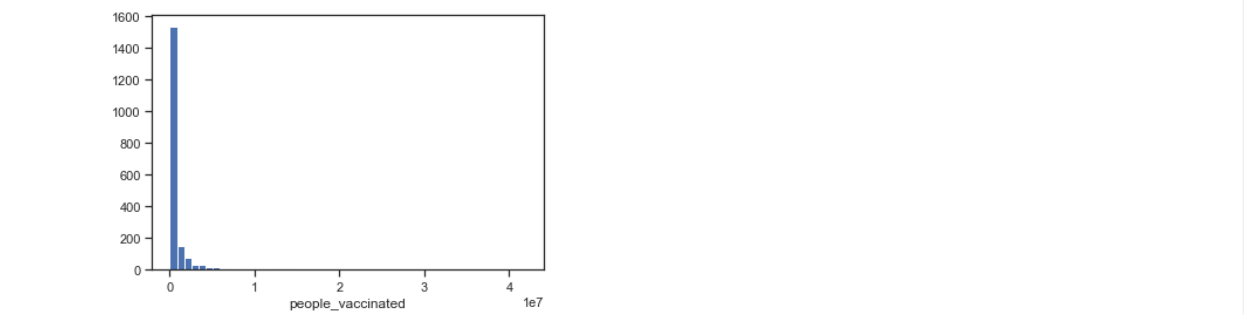
Out[25]:

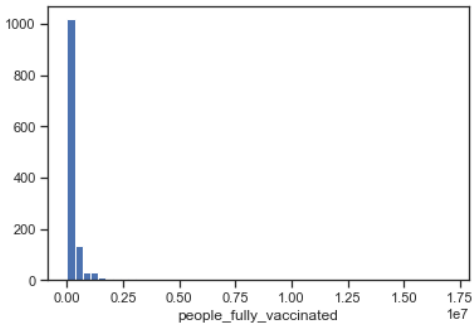
	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinat
0	0.0	0.0	NaN	NaN	NaN	0.00	
1	NaN	NaN	NaN	NaN	64.0	NaN	
2	128.0	128.0	NaN	NaN	64.0	0.00	
3	188.0	188.0	NaN	60.0	63.0	0.01	
4	266.0	266.0	NaN	78.0	66.0	0.01	
...	...	...	...	...	...	...	
3550	790211.0	784809.0	5402.0	13987.0	26206.0	25.06	
3551	803178.0	795927.0	7251.0	12967.0	24418.0	25.47	
3552	820339.0	807351.0	12988.0	17161.0	23033.0	26.02	
3553	841975.0	822633.0	19342.0	21636.0	22012.0	26.70	
3554	864498.0	839065.0	25433.0	22523.0	20649.0	27.42	

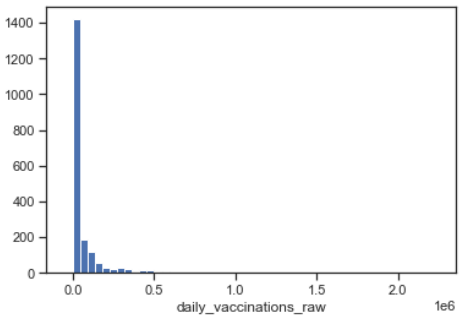
3555 rows × 9 columns

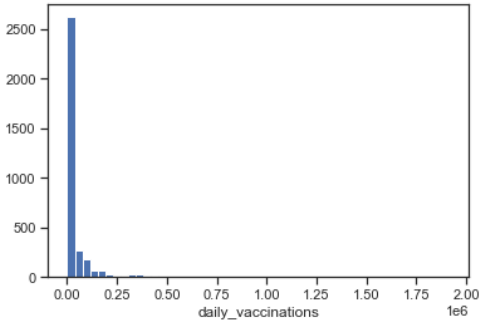
В [26]:
```
# Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

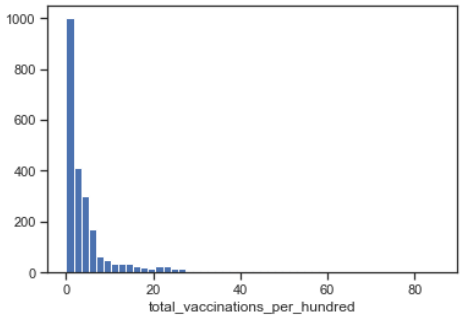


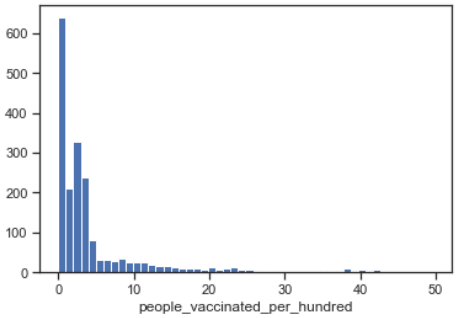


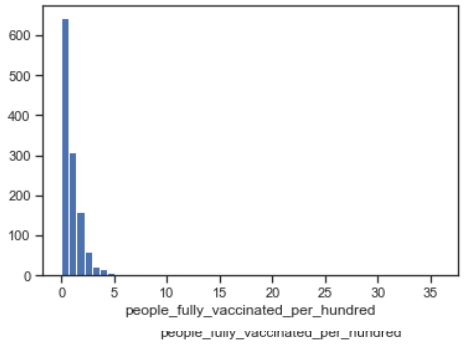

1000
800
600
400
200
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
people_fully_vaccinated
1e7

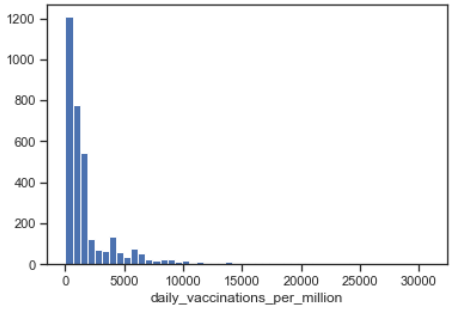

1400
1200
1000
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
daily_vaccinations_raw
1e6


2500
2000
1500
1000
500
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
daily_vaccinations
1e6


1000
800
600
400
200
0
0
20
40
60
80
total_vaccinations_per_hundred


600
500
400
300
200
100
0
0
10
20
30
40
50
people_vaccinated_per_hundred





B [56]:
```
data_num_total_vaccinations = data_num[['total_vaccinations']]
data_num_total_vaccinations.head()
```

Out[56]:

	total_vaccinations
0	0.0
1	NaN
2	128.0
3	188.0
4	266.0

B [29]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

B [57]:
```
# Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_total_vaccinations)
mask_missing_values_only
```

Out[57]:
```
array([[False],
       [ True],
       [False],
       ...,
       [False],
       [False],
       [False]])
```

B [31]:
```
strategies=['mean', 'median', 'most_frequent']
```

B [58]:
```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_total_vaccinations)
    return data_num_imp[mask_missing_values_only]
```

B [33]:
```
strategies[0], test_num_impute(strategies[0])
```

Out[33]:
```
('mean',
 array([1486318.75651431, 1486318.75651431, 1486318.75651431, ...,
        1486318.75651431, 1486318.75651431, 1486318.75651431]))
```

B [34]:
```
strategies[1], test_num_impute(strategies[1])
```

Out[34]:
```
('median', array([189525., 189525., 189525., ..., 189525., 189525., 189525.]))
```

B [35]:
```
strategies[2], test_num_impute(strategies[2])
```

Out[35]:
```
('most_frequent', array([0., 0., 0., ..., 0., 0., 0.]))
```

In [36]:
```
# 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]
```

In [37]:
```
data[['people_vaccinated']].describe()
```

Out[37]:

	people_vaccinated
count	1.940000e+03
mean	1.238670e+06
std	4.052163e+06
min	0.000000e+00
25%	2.636475e+04
50%	1.661540e+05
75%	6.296642e+05
max	4.197740e+07

In [38]:
```
test_num_impute_col(data, 'people_vaccinated', strategies[0])
```

Out[38]:
```
('people_vaccinated', 'mean', 1615, 1238669.6015463918, 1238669.6015463918)
```

In [39]:
```
test_num_impute_col(data, 'people_vaccinated', strategies[1])
```

Out[39]:
```
('people_vaccinated', 'median', 1615, 166154.0, 166154.0)
```

In [40]:
```
test_num_impute_col(data, 'people_vaccinated', strategies[2])
```

Out[40]:
```
('people_vaccinated', 'most_frequent', 1615, 0.0, 0.0)
```

Обработка пропусков в категориальных данных

In [45]:
```
# Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
countr = []
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'):
        countr.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 iso_code. Тип данных object. Количество пустых значений 272, 7.65%.
```

In [46]:
```
countr_data = data[['iso_code']]
countr_data.head()
```

Out[46]:

	iso_code
0	ALB
1	ALB
2	ALB
3	ALB
4	ALB

In [47]:
```
countr_data['iso_code'].unique()
```

Out[47]:
```
array(['ALB', 'DZA', 'AND', 'AIA', 'ARG', 'AUT', 'AZE', 'BHR', 'BGD',
       'BRB', 'BEL', 'BMU', 'BOL', 'BRA', 'BGR', 'KHM', 'CAN', 'CYM',
       'CHL', 'CHN', 'COL', 'CRI', 'HRV', 'CYP', 'CZE', 'DNK', 'DOM',
       'ECU', 'EGY', nan, 'EST', 'FRO', 'FIN', 'FRA', 'DEU', 'GIB', 'GRC',
       'GRL', 'GGY', 'GUY', 'HUN', 'ISL', 'IND', 'IDN', 'IRN', 'IRL',
       'IMN', 'ISR', 'ITA', 'JPN', 'JEY', 'KWT', 'LVA', 'LIE', 'LTU',
       'LUX', 'MAC', 'MDV', 'MLT', 'MUS', 'MEX', 'MCO', 'MAR', 'MMR',
       'NPL', 'NLD', 'OWID_NCY', 'NOR', 'OMN', 'PAK', 'PAN', 'PER', 'POL',
       'PRT', 'QAT', 'ROU', 'RUS', 'SHN', 'SAU', 'SRB', 'SYC', 'SGP',
       'SVK', 'SVN', 'ZAF', 'ESP', 'LKA', 'SWE', 'CHE', 'TUR', 'TCA',
       'ARE', 'GBR', 'USA'], dtype=object)
```

In [48]:
```
countr_data[countr_data['iso_code'].isnull()].shape
```

Out[48]:
```
(272, 1)
```

In [60]:
```
# Импьютация наиболее частыми значениями
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(countr_data)
data_imp2
```

Out[60]:
```
array([['ALB'],
       ['ALB'],
       ['ALB'],
       ...,
       ['GBR'],
       ['GBR'],
       ['GBR']], dtype=object)
```

In [53]:
```
np.unique(data_imp2)
```

Out[53]:
```
array(['AIA', 'ALB', 'AND', 'ARE', 'ARG', 'AUT', 'AZE', 'BEL', 'BGD',
       'BGR', 'BHR', 'BMU', 'BOL', 'BRA', 'BRB', 'CAN', 'CHE', 'CHL',
       'CHN', 'COL', 'CRI', 'CYM', 'CYP', 'CZE', 'DEU', 'DNK', 'DOM',
       'DZA', 'ECU', 'EGY', 'ESP', 'EST', 'FIN', 'FRA', 'FRO', 'GBR',
       'GGY', 'GIB', 'GRC', 'GRL', 'GUY', 'HRV', 'HUN', 'IDN', 'IMN',
       'IND', 'IRL', 'IRN', 'ISL', 'ISR', 'ITA', 'JEY', 'JPN', 'KHM',
       'KWT', 'LIE', 'LKA', 'LTU', 'LUX', 'LVA', 'MAC', 'MAR', 'MCO',
       'MDV', 'MEX', 'MLT', 'MMR', 'MUS', 'NLD', 'NOR', 'NPL', 'OMN',
       'OWID_NCY', 'PAK', 'PAN', 'PER', 'POL', 'PRT', 'QAT', 'ROU', 'RUS',
       'SAU', 'SGP', 'SHN', 'SRB', 'SVK', 'SVN', 'SWE', 'SYC', 'TCA',
       'TUR', 'USA', 'ZAF'], dtype=object)
```

In [76]:
```
# Импьютация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='NA')
data_imp3 = imp3.fit_transform(countr_data)
data_imp3
```

Out[76]:
```
array([['ALB'],
       ['ALB'],
       ['ALB'],
       ...,
       ['NA'],
       ['NA'],
       ['NA']], dtype=object)
```

In [54]:
```
np.unique(data_imp3)
```

Out[54]:
```
array(['AIA', 'ALB', 'AND', 'ARE', 'ARG', 'AUT', 'AZE', 'BEL', 'BGD',
       'BGR', 'BHR', 'BMU', 'BOL', 'BRA', 'BRB', 'CAN', 'CHE', 'CHL',
       'CHN', 'COL', 'CRI', 'CYM', 'CYP', 'CZE', 'DEU', 'DNK', 'DOM',
       'DZA', 'ECU', 'EGY', 'ESP', 'EST', 'FIN', 'FRA', 'FRO', 'GBR',
       'GGY', 'GIB', 'GRC', 'GRL', 'GUY', 'HRV', 'HUN', 'IDN', 'IMN',
       'IND', 'IRL', 'IRN', 'ISL', 'ISR', 'ITA', 'JEY', 'JPN', 'KHM',
       'KWT', 'LIE', 'LKA', 'LTU', 'LUX', 'LVA', 'MAC', 'MAR', 'MCO',
       'MDV', 'MEX', 'MLT', 'MMR', 'MUS', 'NA', 'NLD', 'NOR', 'NPL',
       'OMN', 'OWID_NCY', 'PAK', 'PAN', 'PER', 'POL', 'PRT', 'QAT', 'ROU',
       'RUS', 'SAU', 'SGP', 'SHN', 'SRB', 'SVK', 'SVN', 'SWE', 'SYC',
       'TCA', 'TUR', 'USA', 'ZAF'], dtype=object)
```

In [55]:
```
data_imp3[data_imp3=='NA'].size
```

Out[55]:
```
272
```

Преобразование категориальных признаков в числовые

In [61]:
```
countr_vac = pd.DataFrame({'c1':data_imp2.T[0]})
countr_vac
```

Out[61]:

	c1
0	ALB
1	ALB
2	ALB
3	ALB
4	ALB
...	...
3550	GBR
3551	GBR
3552	GBR
3553	GBR
3554	GBR

3555 rows x 1 columns

Кодирование категорий целочисленными значениями - label encoding

В [62]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

В [64]:
```
le = LabelEncoder()
countr_vac_le = le.fit_transform(countr_vac['c1'])
```

В [65]:
```
countr_vac['c1'].unique()
```

Out[65]:
```
array(['ALB', 'DZA', 'AND', 'AIA', 'ARG', 'AUT', 'AZE', 'BHR', 'BGD',
       'BRB', 'BEL', 'BMU', 'BOL', 'BRA', 'BGR', 'KHM', 'CAN', 'CYM',
       'CHL', 'CHN', 'COL', 'CRI', 'HRV', 'CYP', 'CZE', 'DNK', 'DOM',
       'ECU', 'EGY', 'GBR', 'EST', 'FRO', 'FIN', 'FRA', 'DEU', 'GIB',
       'GRC', 'GRL', 'GGY', 'GUY', 'HUN', 'ISL', 'IND', 'IDN', 'IRN',
       'IRL', 'IMN', 'ISR', 'ITA', 'JPN', 'JEY', 'KWT', 'LVA', 'LIE',
       'LTU', 'LUX', 'MAC', 'MDV', 'MLT', 'MUS', 'MEX', 'MCO', 'MAR',
       'MMR', 'NPL', 'NLD', 'OWID_NCY', 'NOR', 'OMN', 'PAK', 'PAN', 'PER',
       'POL', 'PRT', 'QAT', 'ROU', 'RUS', 'SHN', 'SAU', 'SRB', 'SYC',
       'SGP', 'SVK', 'SVN', 'ZAF', 'ESP', 'LKA', 'SWE', 'CHE', 'TUR',
       'TCA', 'ARE', 'USA'], dtype=object)
```

В [67]:
```
np.unique(countr_vac_le)
```

Out[67]:
```
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])
```

В [68]:
```
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, 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])
```

Out[68]:
```
array(['AIA', 'ALB', 'AND', 'ARE', 'ARG', 'AUT', 'AZE', 'BEL', 'BGD',
       'BGR', 'BHR', 'BMU', 'BOL', 'BRA', 'BRB', 'CAN', 'CHE', 'CHL',
       'CHN', 'COL', 'CRI', 'CYM', 'CYP', 'CZE', 'DEU', 'DNK', 'DOM',
       'DZA', 'ECU', 'EGY', 'ESP', 'EST', 'FIN', 'FRA', 'FRO', 'GBR',
       'GGY', 'GIB', 'GRC', 'GRL', 'GUY', 'HRV', 'HUN', 'IDN', 'IMN',
       'IND', 'IRL', 'IRN', 'ISL', 'ISR', 'ITA', 'JEY', 'JPN', 'KHM',
       'KWT', 'LIE', 'LKA', 'LTU', 'LUX', 'LVA', 'MAC', 'MAR', 'MCO',
       'MDV', 'MEX', 'MLT', 'MMR', 'MUS', 'NLD', 'NOR', 'NPL', 'OMN',
       'OWID_NCY', 'PAK', 'PAN', 'PER', 'POL', 'PRT', 'QAT', 'ROU', 'RUS',
       'SAU', 'SGP', 'SHN', 'SRB', 'SVK', 'SVN', 'SWE', 'SYC', 'TCA',
       'TUR', 'USA', 'ZAF'], dtype=object)
```

Кодирование категорий наборами бинарных значений - one-hot encoding

В [69]:
```
ohe = OneHotEncoder()
countr_vac_ohe = ohe.fit_transform(countr_vac[['c1']])
```

В [70]:
```
countr_vac.shape
```

Out[70]:
```
(3555, 1)
```

В [71]:
```
countr_vac_ohe.shape
```

Out[71]:
```
(3555, 93)
```

В [72]:
```
countr_vac_ohe
```

Out[72]:
```
<3555x93 sparse matrix of type '<class 'numpy.float64'>'
        with 3555 stored elements in Compressed Sparse Row format>
```

```
В [73]: countr_vac_ohe.todense()[0:10]
```

```
Out[73]: matrix([[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., 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., 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., 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., 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., 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., 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., 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., 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., 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., 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., 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., 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.]])
```

```
В [74]: countr_vac.head(10)
```

Out[74]:

	c1
0	ALB
1	ALB
2	ALB
3	ALB
4	ALB
5	ALB
6	ALB
7	ALB
8	ALB
9	ALB

Pandas get_dummies - быстрый вариант one-hot кодирования

B [75]:
```
pd.get_dummies(countr_vac).head()
```

Out[75]:

	c1_AIA	c1_ALB	c1_AND	c1_ARE	c1_ARG	c1_AUT	c1_AZE	c1_BEL	c1_BGD	c1_BGR	...	c1_SHN	c1_SRB	c1_SVK	c1_SVN	c1_SWE	c1_SYC	c1_
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

5 rows × 93 columns

B [77]:
```
pd.get_dummies(countr_data, dummy_na=True).head()
```

Out[77]:

	iso_code_AIA	iso_code_ALB	iso_code_AND	iso_code_ARE	iso_code_ARG	iso_code_AUT	iso_code_AZE	iso_code_BEL	iso_code_BGD	iso_code_BGR	.
0	0	1	0	0	0	0	0	0	0	0	.
1	0	1	0	0	0	0	0	0	0	0	.
2	0	1	0	0	0	0	0	0	0	0	.
3	0	1	0	0	0	0	0	0	0	0	.
4	0	1	0	0	0	0	0	0	0	0	.

5 rows × 94 columns

ЛР №2

Используются данные Heart attack analysis & prediction dataset

B [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

B [2]:
```
data = pd.read_csv('heart.csv', sep=",")
```

B [3]:
```
# размер набора данных
data.shape
```

Out[3]:
```
(303, 14)
```

B [4]:
```
# типы колонок
data.dtypes
```

Out[4]:
```
age           int64
sex           int64
cp            int64
trtbps        int64
chol          int64
fbs           int64
restecg       int64
thalachh      int64
exng          int64
oldpeak     float64
slp           int64
caa           int64
thall         int64
output        int64
dtype: object
```

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

Out[5]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

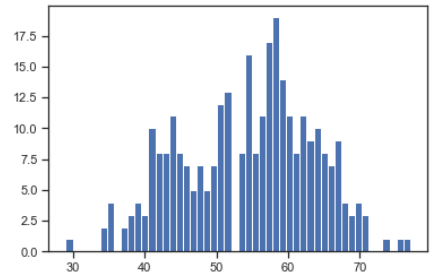
Масштабирование значений

B [7]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

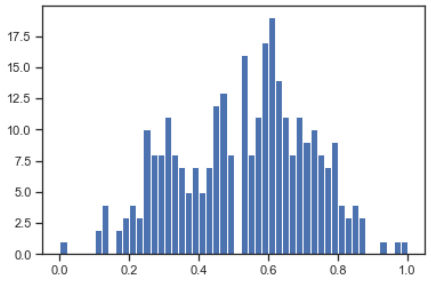
MinMax масштабирование

B [8]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['age']])
```

В [9]:
```
plt.hist(data['age'], 50)
plt.show()
```



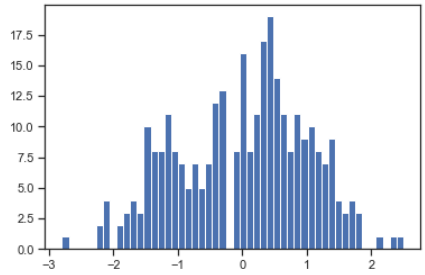
В [10]:
```
plt.hist(sc1_data, 50)
plt.show()
```



Масштабирование данных на основе Z-оценки - StandardScaler

В [11]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['age']])
```

В [12]:
```
plt.hist(sc2_data, 50)
plt.show()
```



В [ ]: