



Министерство образования и науки Российской Федерации
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

Кафедра «Информатика и системы управления» (ИУ5)

ДИСЦИПЛИНА: «Технологии машинного обучения»

Отчет по лабораторной работе №3
«Обработка пропусков в данных, кодирование категориальных
признаков, масштабирование данных»

Выполнила:

Студентка группы ИУ5-61Б

Мартынова Д.П.

Преподаватели:

Гапанюк Ю.Е.

Москва, 2020 г.

Цель лабораторной работы: изучение способов предварительной обработки данных для дальнейшего формирования моделей.

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.

Выполнение ЛР:

Загрузка и первичный анализ данных

```
In [32]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```
In [33]: # используем выборку, связанную с коронавирусом
data = pd.read_csv('COVID19_line_list_data.csv', sep=",")
```

```
In [34]: data.shape
```

```
Out[34]: (1085, 27)
```

```
In [35]: data.dtypes
```

```
Out[35]: id                                int64
case_in_country                           float64
reporting date                            object
Unnamed: 3                                float64
summary                                   object
location                                  object
country                                   object
gender                                    object
age                                       float64
symptom_onset                             object
If_onset_approximated                     float64
hosp_visit_date                           object
exposure_start                            object
exposure_end                             object
visiting Wuhan                            int64
from Wuhan                                float64
death                                     object
recovered                                 object
symptom                                   object
source                                    object
link                                      object
Unnamed: 21                               float64
Unnamed: 22                               float64
Unnamed: 23                               float64
Unnamed: 24                               float64
Unnamed: 25                               float64
Unnamed: 26                               float64
dtype: object
```

```
In [36]: # проверим есть ли пропущенные значения
data.isnull().sum()
```

```
Out[36]: id                                0
case_in_country                           197
reporting date                             1
Unnamed: 3                                1085
summary                                    5
location                                  0
country                                   0
gender                                    183
age                                        242
symptom_onset                             522
If_onset_approximated                     525
hosp_visit_date                           578
exposure_start                             957
exposure_end                               744
visiting Wuhan                             0
from Wuhan                                 4
death                                      0
recovered                                  0
symptom                                    815
source                                     0
link                                       0
Unnamed: 21                               1085
Unnamed: 22                               1085
Unnamed: 23                               1085
Unnamed: 24                               1085
Unnamed: 25                               1085
Unnamed: 26                               1085
dtype: int64
```

```
In [37]: #удалим полностью пустые колонки
dat = data.dropna(axis=1, how='all')
dat.shape
```

```
Out[37]: (1085, 20)
```

1. Обработка пропусков в данных

1.1. Простые стратегии - удаление или заполнение нулями

```
In [39]: # Удаление колонок, содержащих пустые значения
data_new_1 = dat.dropna(axis=1, how='any')
(dat.shape, data_new_1.shape)
```

```
Out[39]: ((1085, 20), (1085, 8))
```

```
In [40]: # Удаление строк, содержащих пустые значения
data_new_2 = dat.dropna(axis=0, how='any')
(dat.shape, data_new_2.shape)
```

```
Out[40]: ((1085, 20), (20, 20))
```

Заполнить все пропуски нулями не получится, потому что присутствуют категориальные столбцы

1.2. "Внедрение значений" - импьютация (imputation)

1.2.1. Обработка пропусков в числовых данных

```
In [118]: # Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
total_count = dat.shape[0]
for col in dat.columns:
    # Количество пустых значений
    temp_null_count = dat[dat[col].isnull()].shape[0]
    dt = str(dat[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 case_in_country. Тип данных float64. Количество пустых значений 197, 18.16%.
Колонка age. Тип данных float64. Количество пустых значений 242, 22.3%.
Колонка If_onset_approximated. Тип данных float64. Количество пустых значений 525, 48.39%.
Колонка from Wuhan. Тип данных float64. Количество пустых значений 4, 0.37%.

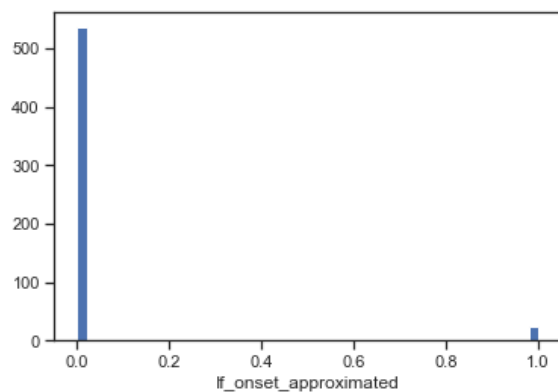
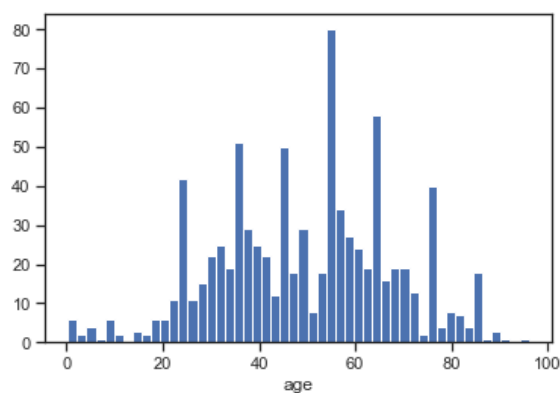
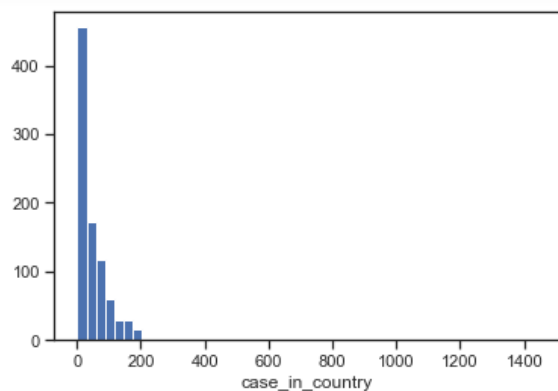
```
In [119]: # Фильтр по колонкам с пропущенными значениями
data_num = dat[num_cols]
data_num
```

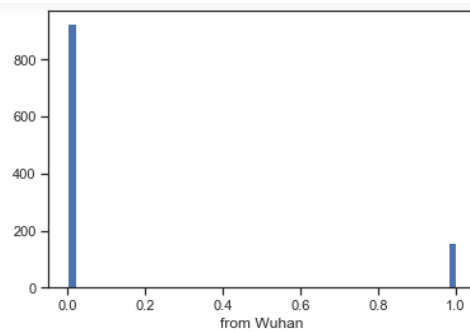
Out[119]:

	case_in_country	age	If_onset_approximated	from Wuhan
0	NaN	66.0	0.0	0.0
1	NaN	56.0	0.0	1.0
2	NaN	46.0	0.0	1.0
3	NaN	60.0	NaN	0.0
4	NaN	58.0	NaN	0.0
...
1080	2.0	24.0	NaN	0.0
1081	1.0	35.0	NaN	0.0
1082	1.0	NaN	NaN	0.0
1083	1.0	NaN	NaN	0.0
1084	1.0	70.0	NaN	0.0

1085 rows × 4 columns

```
In [120]: # Гистограмма по признакам
for col in data_num:
    plt.hist(dat[col], 50)
    plt.xlabel(col)
    plt.show()
```





```
In [121]: # Фильтр по пустым значениям поля MasVnrArea
dat[dat['age'].isnull()]
```

```
Out[121]:
```

	id	case_in_country	reporting date	summary	location	country	gender	age	symptom_onset	lf_onset_approximated	hosp_visit_date	exposure_start
22	23	NaN	1/22/2020	First confirmed imported COVID-19 pneumonia pa...	Shanxi	China	male	NaN	1/19/2020	0.0	1/20/2020	01/12/20
91	92	NaN	1/24/2020	new recovered imported COVID-19 pneumonia pati...	Beijing	China	female	NaN	NaN	NaN	NaN	NaN
116	117	NaN	1/25/2020	new recovered imported COVID-19 pneumonia pati...	Beijing	China	male	NaN	NaN	NaN	NaN	NaN

```
In [122]: # Запоминаем индексы строк с пустыми значениями
flt_index = dat[dat['age'].isnull()].index
flt_index
```

```
Out[122]: Int64Index([ 22,  91, 116, 145, 146, 147, 177, 201, 202, 204,
...,
1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1082, 1083],
dtype='int64', length=242)
```

```
In [123]: data_age = data_num[['age']]
data_age.head()
```

```
Out[123]:
```

	age
0	66.0
1	56.0
2	46.0
3	60.0
4	58.0

```
In [124]: from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

```
In [125]: # Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_age)
mask_missing_values_only
```

```
Out[125]: array([[False],
[False],
[False],
...,
[ True],
[ True],
[False]])
```

```
In [126]: strategies=['mean', 'median', 'most_frequent']
```

```
In [127]: def test_num_impute(strategy_param):
imp_num = SimpleImputer(strategy=strategy_param)
data_num_imp = imp_num.fit_transform(data_age)
return data_num_imp[mask_missing_values_only]
```

```
In [128]: strategies[0], test_num_impute(strategies[0])
```

```
In [129]: strategies[1], test_num_impute(strategies[1])
```

[illegible]

```
In [130]: strategies[2], test_num_impute(strategies[2])
```

[illegible]

Заменяем все пустые данные столбца 'age' на данные test_num_impute(strategies[0])

```
In [131]: new_age = pd.DataFrame({'id': flt_index,
                                'age': test_num_impute(strategies[0])})
new_age
```

Out[131]:

	id	age
0	22	49.483689
1	91	49.483689
2	116	49.483689
3	145	49.483689
4	146	49.483689
...
237	1076	49.483689
238	1077	49.483689
239	1078	49.483689
240	1082	49.483689
241	1083	49.483689

```
In [133]: for index, row in new_age.iterrows():
          dat.loc[row['id'], 'age'] = row['age']
          dat
```

Таким образом заменили все пустые значения в столбце age, на среднее значение по этому столбцу

1.2.2. Обработка пропусков в категориальных данных

```
In [135]: # Выберем категориальные колонки с пропущенными значениями
          # Цикл по колонкам датасета
          cat_cols = []
          for col in dat.columns:
              # Количество пустых значений
              temp_null_count = dat[dat[col].isnull()].shape[0]
              dt = str(dat[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 reporting date. Тип данных object. Количество пустых значений 1, 0.09%.
Колонка summary. Тип данных object. Количество пустых значений 5, 0.46%.
Колонка gender. Тип данных object. Количество пустых значений 183, 16.87%.
Колонка symptom_onset. Тип данных object. Количество пустых значений 522, 48.11%.
Колонка hosp_visit_date. Тип данных object. Количество пустых значений 578, 53.27%.
Колонка exposure_start. Тип данных object. Количество пустых значений 957, 88.2%.
Колонка exposure_end. Тип данных object. Количество пустых значений 744, 68.57%.
Колонка symptom. Тип данных object. Количество пустых значений 815, 75.12%.

```
In [139]: cat_temp_data['gender'].unique()
```

```
Out[139]: array(['male', 'female', nan], dtype=object)
```

```
In [140]: cat_temp_data[cat_temp_data['gender'].isnull()].shape
```

```
Out[140]: (183, 1)
```

```
In [141]: # Импутация наиболее частыми значениями
          imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
          data_imp2 = imp2.fit_transform(cat_temp_data)
          data_imp2
```

```
Out[141]: array([[ 'male'],
                  [ 'female'],
                  [ 'male'],
                  ...,
                  [ 'male'],
                  [ 'male'],
                  [ 'male']], dtype=object)
```

```
In [142]: # Пустые значения отсутствуют
          np.unique(data_imp2)
```

```
Out[142]: array(['female', 'male'], dtype=object)
```

```
In [160]: for index, row in cat_enc.iterrows():
          dat.loc[index, 'gender'] = row['c1']
          dat
```

2. Преобразование категориальных признаков в числовые

```
In [147]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
In [149]: le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
cat_enc_le
```

```
Out[149]: array([1, 0, 1, ..., 1, 1, 1])
```

```
In [166]: gen_enc = pd.DataFrame({'gen':cat_enc_le})
for index, row in gen_enc.iterrows():
    dat.loc[index, 'gender'] = row['gen']
dat
```

C:\Users\Dasik\anaconda3\lib\site-packages\pandas\core\indexing.py:965: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
self.obj[item] = s
```

```
Out[166]:
```

	id	case_in_country	reporting date	summary	location	country	gender	age	symptom_onset	lf_onset_approximated	hosp_visit_date
0	1	NaN	1/20/2020	First confirmed imported COVID-19 pneumonia pa...	Shenzhen, Guangdong	China	1	66.000000	01/03/20	0.0	01/11/20

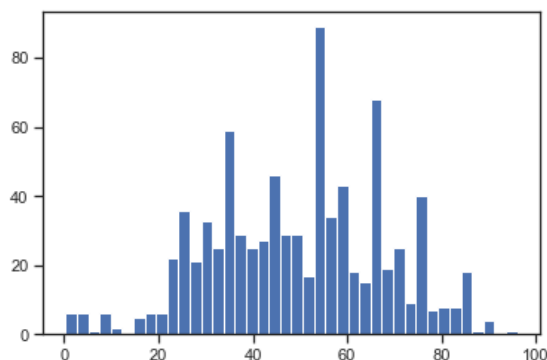
3. Масштабирование данных

3.1. MinMax масштабирование

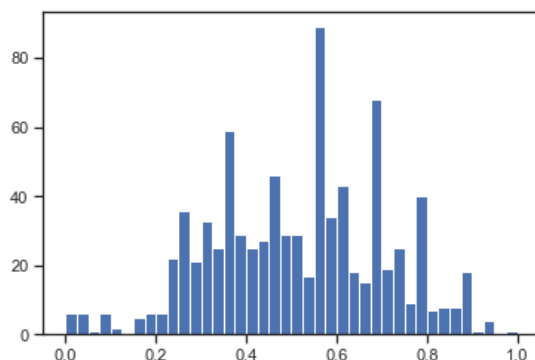
```
In [167]: from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

```
In [168]: sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['age']])
```

```
In [174]: plt.hist(data['age'], 40)
plt.show()
```



```
In [175]: plt.hist(sc1_data, 40)
plt.show()
```



3.2. Нормализация данных

```
In [177]: sc2 = Normalizer()  
sc2_data = sc2.fit_transform(dat[['age']])
```

```
In [178]: plt.hist(sc2_data, 40)  
plt.show()
```

