

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



Лабораторная работа №2
по дисциплине
«Методы машинного обучения»
на тему
«Обработка признаков»

Выполнил:
студент группы ИУ5-23М
Дин Но

Москва — 2021 г.

Цель лабораторной работы:

Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

1. Выбрать набор данных (датасет), содержащий категориальные и числовые 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 лекций решить следующие задачи:
 - устранение пропусков в данных;
 - кодирование категориальных признаков;
 - нормализацию числовых признаков.

Ход выполнения работы

1. Обработка пропусков в данных

Данные, которые я выбираю, - это то, как каждая страна в мире ежедневно вакцинируется против нового коронавируса.

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import sklearn.impute
import sklearn.preprocessing

pd.set_option("display.width",70)
data=pd.read_csv("covid19.csv")
data.head
```

```
Out[1]: <bound method NDFrame.head of
country iso_code date total_vaccinations \
0 Albania ALB 2021-01-10 0.0
1 Albania ALB 2021-01-11 NaN
2 Albania ALB 2021-01-12 128.0
3 Albania ALB 2021-01-13 188.0
4 Albania ALB 2021-01-14 266.0
... ..
3307 Wales NaN 2021-02-11 719954.0
3308 Wales NaN 2021-02-12 753669.0
3309 Wales NaN 2021-02-13 776224.0
3310 Wales NaN 2021-02-14 790211.0
3311 Wales NaN 2021-02-15 803178.0

people_vaccinated people_fully_vaccinated \
0 0.0 NaN
1 NaN NaN
2 128.0 NaN
3 188.0 NaN
4 266.0 NaN
```

data.dtypes

```
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
```

Найдем все пропуски в данных:

```
data.isnull().sum() #Обработка пропусков в данных
```

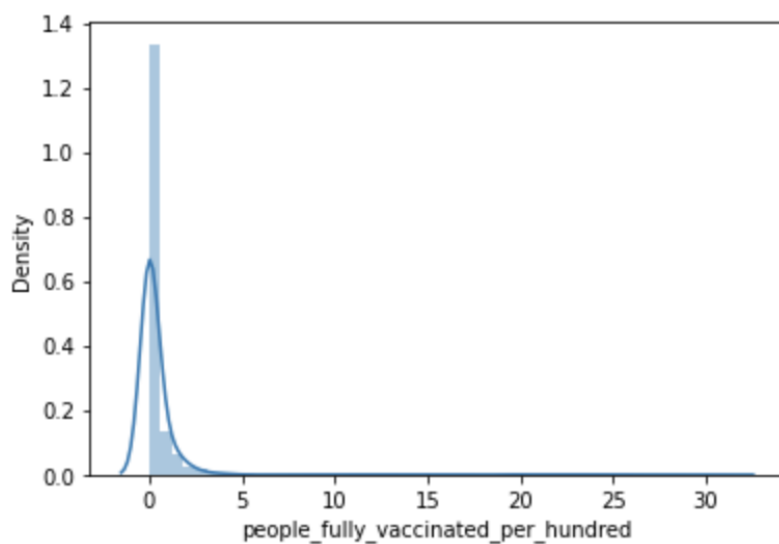
```
country          0
iso_code         260
date             0
total_vaccinations 1167
people_vaccinated 1538
people_fully_vaccinated 2175
daily_vaccinations_raw 1528
daily_vaccinations 126
total_vaccinations_per_hundred 1167
people_vaccinated_per_hundred 1538
people_fully_vaccinated_per_hundred 2175
daily_vaccinations_per_million 126
vaccines         0
source_name      0
source_website   0
dtype: int64
```

Очевидно, что мы будем работать с колонкой ‘people_fully_vaccinated’ или ‘people_fully_vaccinated_per_hundred’. Я выбрал ‘people_fully_vaccinated_per_hundred’

Самый простой вариант — заполнить пропуски нулями:

```
sns.distplot(data['people_fully_vaccinated_per_hundred'].fillna(0));
```

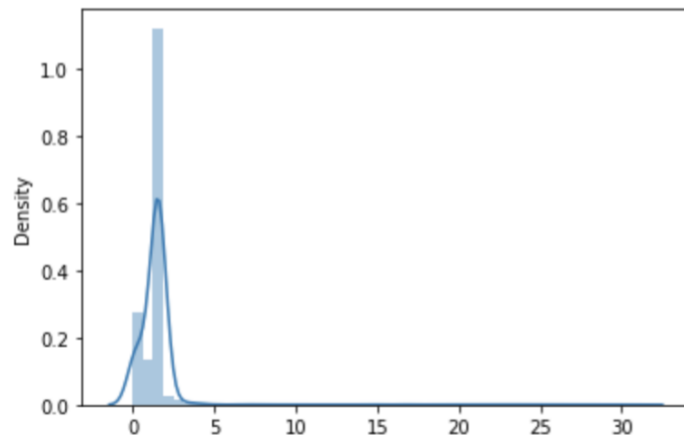
```
/Users/ding/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:250: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use `displot` (a faceted version) or `histplot` (an axes-level function with similar flexibility) or `histplot` (an axes-level function with similar flexibility) or `histplot` (an axes-level function with similar flexibility).
warnings.warn(msg, FutureWarning)
```



Видно, что в данной ситуации это приводит к выбросам. Логичнее было бы приложениям без данных присваивать средний:

```
mean_imp = sklearn.impute.SimpleImputer(strategy="mean")
mean_rat = mean_imp.fit_transform(data[["people_fully_vaccinated_per_hundred"]])
sns.distplot(mean_rat);
```

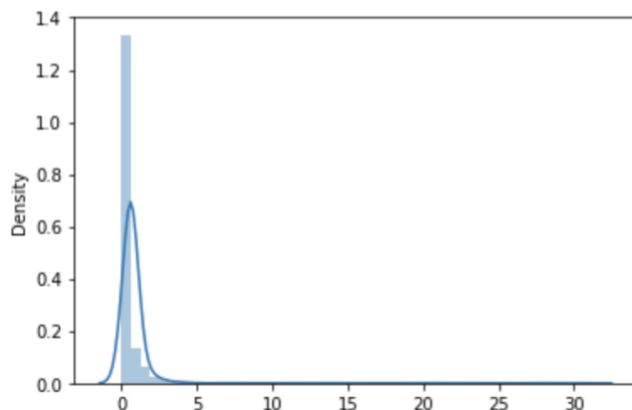
```
/Users/ding/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:25:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use the `hist` level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```



Попробуем также медианный и самые частые данные:

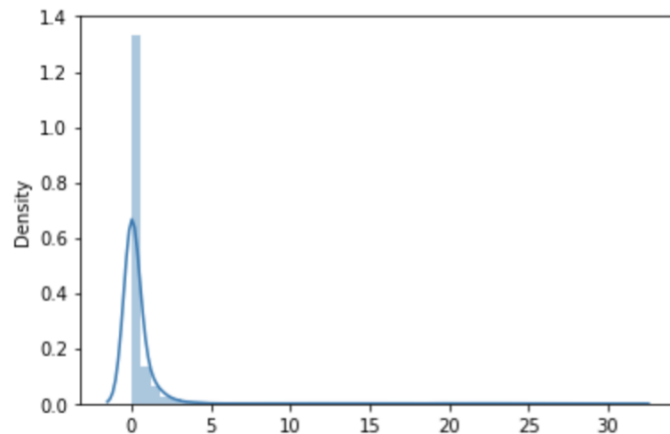
```
med_imp = sklearn.impute.SimpleImputer(strategy="median")
med_rat = med_imp.fit_transform(data[["people_fully_vaccinated_per_hundred"]])
sns.distplot(med_rat);
```

```
/Users/ding/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:25:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use the `hist` level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```



```
freq_imp = sklearn.impute.SimpleImputer(strategy="most_frequent")
freq_rat = freq_imp.fit_transform(data[["people_fully_vaccinated_per_hundred"]])
sns.distplot(freq_rat);
```

/Users/ding/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:255: deprecated function and will be removed in a future version. Please adapt your code level function with similar flexibility) or `histplot` (an axes-level function for warnings.warn(msg, FutureWarning)



Видно, что самый близкий к нормальному распределению график дал обычное среднее значение. Остановимся на нём:

```
data["people_fully_vaccinated_per_hundred"] = mean_rat
```

2. Кодирование категориальных признаков

Рассмотрим колонку 'vaccines':

```
data["people_fully_vaccinated_per_hundred"] = mean_rat
```

```
vaccines1 = data["vaccines"].dropna().astype(str) #Кодирование категориальных признаков  
vaccines1.value_counts()
```

```
Moderna, Oxford/AstraZeneca, Pfizer/BioNTech      970  
Pfizer/BioNTech                                   771  
Oxford/AstraZeneca, Pfizer/BioNTech                521  
Moderna, Pfizer/BioNTech                           327  
Sputnik V                                           127  
Oxford/AstraZeneca                                  99  
Sinovac                                              71  
Pfizer/BioNTech, Sinovac                           64  
Sinopharm/Beijing, Sinopharm/Wuhan, Sinovac        57  
Pfizer/BioNTech, Sinopharm/Beijing                 56  
Oxford/AstraZeneca, Sinopharm/Beijing               56  
Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinopharm/Wuhan, Sputnik V  43  
Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V       38  
Covaxin, Oxford/AstraZeneca                        33  
Oxford/AstraZeneca, Sinovac                         32  
Oxford/AstraZeneca, Sputnik V                      21  
Sinopharm/Beijing                                  17  
Oxford/AstraZeneca, Sinopharm/Beijing, Sputnik V    9  
Name: vaccines, dtype: int64
```

Выполним кодирование категорий целочисленными значениями:

```
le = sklearn.preprocessing.LabelEncoder()  
vaccines1_le = le.fit_transform(vaccines1)  
print(np.unique(vaccines1_le))  
le.inverse_transform(np.unique(vaccines1_le))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17]
```

```
array(['Covaxin, Oxford/AstraZeneca',  
      'Moderna, Oxford/AstraZeneca, Pfizer/BioNTech',  
      'Moderna, Pfizer/BioNTech', 'Oxford/AstraZeneca',  
      'Oxford/AstraZeneca, Pfizer/BioNTech',  
      'Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing, Sinopharm/Wuhan, Sputnik V',  
      'Oxford/AstraZeneca, Sinopharm/Beijing',  
      'Oxford/AstraZeneca, Sinopharm/Beijing, Sputnik V',  
      'Oxford/AstraZeneca, Sinovac', 'Oxford/AstraZeneca, Sputnik V',  
      'Pfizer/BioNTech', 'Pfizer/BioNTech, Sinopharm/Beijing',  
      'Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V',  
      'Pfizer/BioNTech, Sinovac', 'Sinopharm/Beijing',  
      'Sinopharm/Beijing, Sinopharm/Wuhan, Sinovac', 'Sinovac',  
      'Sputnik V'], dtype=object)
```

Выполним кодирование категорий наборами бинарных значений:

```
vaccines_oh = pd.get_dummies(vaccines1)
vaccines_oh.head()
```

I/AstraZeneca, izer/BioNTech, pharm/Beijing, pharm/Wuhan, Sputnik V	Oxford/AstraZeneca, Sinopharm/Beijing	Oxford/AstraZeneca, Sinopharm/Beijing, Sputnik V	Oxford/AstraZeneca, Sinovac	Oxford/AstraZeneca, Sputnik V	Pfizer/BioNTech	Pfizer/BioNTech, Sinopharm/Beijing	Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V	p
0	0	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	0	1	0	0	
0	0	0	0	0	1	0	0	

```
vaccines_oh[vaccines_oh["Sputnik V"] == 1].head()
```

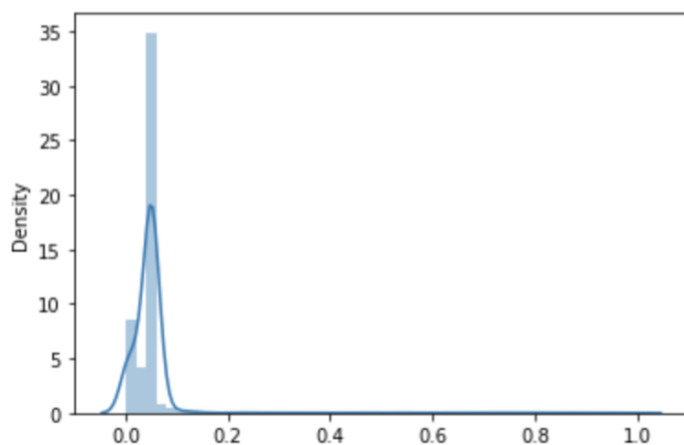
AstraZeneca, Sinovac	Oxford/AstraZeneca, Sputnik V	Pfizer/BioNTech	Pfizer/BioNTech, Sinopharm/Beijing	Pfizer/BioNTech, Sinopharm/Beijing, Sputnik V	Pfizer/BioNTech, Sinovac	Sinopharm/Beijing	Sinopharm/Beijing, Sinopharm/Wuhan, Sinovac	Sinovac	Sputnik V
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1

3. Масштабирование данных

Для начала попробуем обычное MinMax-масштабирование:

```
mm = sklearn.preprocessing.MinMaxScaler() # Масштабирование данных
sns.distplot(mm.fit_transform(data[["people_fully_vaccinated_per_hundred"]]));
```

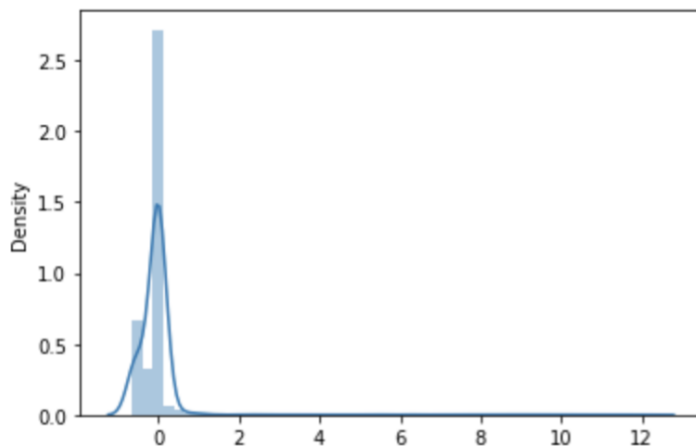
/Users/ding/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2: DeprecationWarning: distplot is a deprecated function and will be removed in a future version. Please adapt your code to use the 'histplot' level function with similar flexibility) or 'histplot' (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)



Результат вполне ожидаемый и вполне приемлемый. Но попробуем и другие варианты, например, масштабирование на основе Z-оценки:

```
ss = sklearn.preprocessing.StandardScaler()  
sns.distplot(ss.fit_transform(data[["people_fully_vaccinated_per_hundred"]]));
```

/Users/ding/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:104: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function).
warnings.warn(msg, FutureWarning)



Также результат ожидаемый, но его применимость зависит от дальнейшего использования.

Список литературы

- [1] Гапанюк Ю. Е. Лабораторная работа «Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugaryanyuk/ml_course/wiki/LAB_MISSING (дата обращения: 05.04.2019).
- [2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource]//Read the Docs. — 2019. — Access mode: <https://ipython.readthedocs.io/en/stable/> (online; accessed: 20.02.2019).
- [3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: <https://seaborn.pydata.org/> (online; accessed: 20.02.2019).

- [4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: <http://pandas.pydata.org/pandas-docs/stable/> (online; accessed: 20.02.2019).
- [5] Gupta L. Google Play Store Apps [Electronic resource] // Kaggle. — 2019. — Access mode: <https://www.kaggle.com/lava18/google-play-store-apps> (online; accessed: 05.04.2019).