

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 пропусков в данных»

Выполнил:

студент группы: ИУ5-23М

Ся Тунтун

Москва — 2022 г.

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

2.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.

3. Ход выполнения работы

Подключим все необходимые библиотеки

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import sklearn.impute
import sklearn.preprocessing
# Enable inline plots
%matplotlib inline
# Set plot style
sns.set(style="ticks")
# Set plots formats to save high resolution PNG
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")
```

Зададим ширину текстового представления данных, чтобы в дальнейшем текст в отчёте влезал на A4 :

```
In [2]: pd.set_option("display.width", 70)
```

Для выполнения данной лабораторной работы возьмём набор данных по приложениям в waetherAUS (Погода в Австралии)

```
In [15]: data = pd.read_csv("weatherAUS.csv")
```

Посмотрим на эти наборы данных:

```
data.head(5)
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidi
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	

5 rows × 23 columns

```
In [17]: data.dtypes
```

```
Out[17]: Date                object
Location                object
MinTemp                float64
MaxTemp                float64
Rainfall                float64
Evaporation            float64
Sunshine                float64
WindGustDir            object
WindGustSpeed          float64
WindDir9am             object
WindDir3pm             object
WindSpeed9am           float64
WindSpeed3pm           float64
Humidity9am            float64
Humidity3pm            float64
Pressure9am            float64
Pressure3pm            float64
Cloud9am               float64
Cloud3pm               float64
Temp9am                float64
Temp3pm                float64
RainToday              object
RainTomorrow           object
dtype: object
```

```
In [18]: data.shape
```

```
Out[18]: (145460, 23)
```

3.1. Обработка пропусков в данных

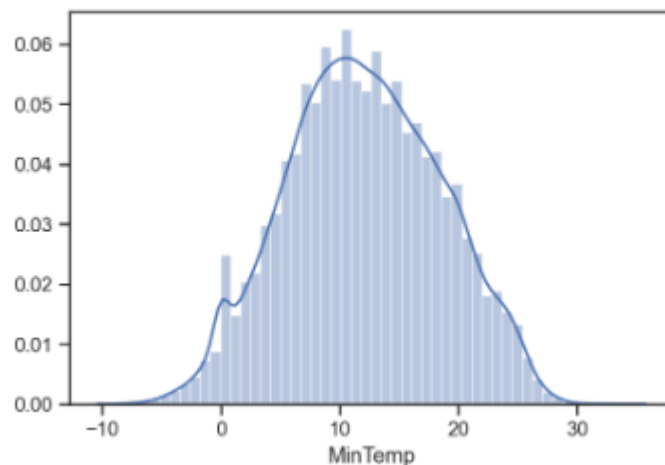
Найдем все пропуски в данных:

```
In [19]: data.isnull().sum()
```

```
Out[19]: Date          0
Location          0
MinTemp          1485
MaxTemp          1261
Rainfall          3261
Evaporation       62790
Sunshine          69835
WindGustDir       10326
WindGustSpeed     10263
WindDir9am        10566
WindDir3pm         4228
WindSpeed9am       1767
WindSpeed3pm       3062
Humidity9am        2654
Humidity3pm        4507
Pressure9am        15065
Pressure3pm        15028
Cloud9am           55888
Cloud3pm           59358
Temp9am            1767
Temp3pm            3609
RainToday          3261
RainTomorrow       3267
dtype: int64
```

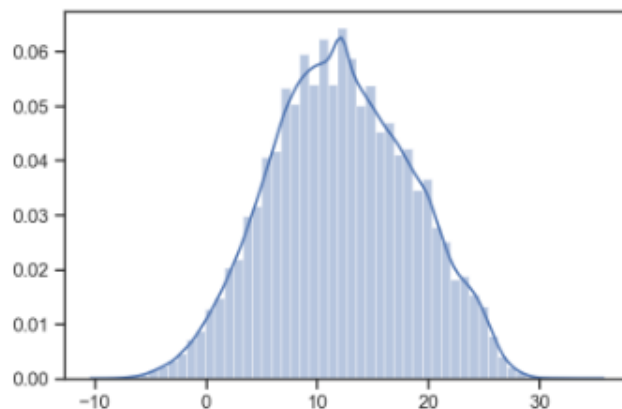
Очевидно, что мы будем работать с колонкой `MinTemp`. Самый простой вариант — заполнить пропуски нулями:

```
In [20]: sns.distplot(data["MinTemp"].fillna(0));
```



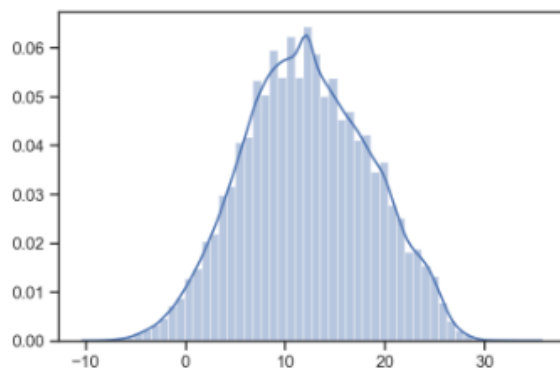
Видно, что в данной ситуации это приводит к выбросам. Логичнее было бы приложениям без рейтинга присваивать средний рейтинг:

```
: mean_imp = sklearn.impute.SimpleImputer(strategy="mean")
mean_mintemp = mean_imp.fit_transform(data[["MinTemp"]])
sns.distplot(mean_mintemp);
```

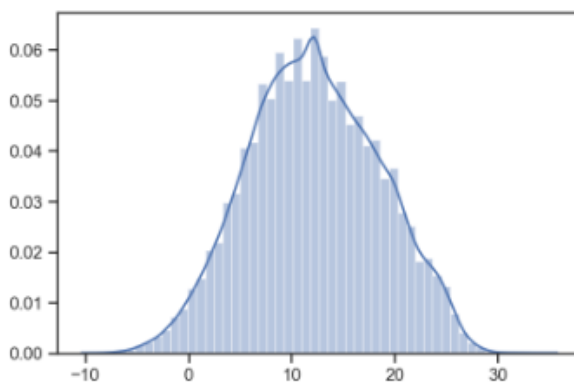


Попробуем также медианный рейтинг и самый частый рейтинг:

```
: med_imp = sklearn.impute.SimpleImputer(strategy="median")
med_mintemp = med_imp.fit_transform(data[["MinTemp"]])
sns.distplot(med_mintemp);
```



```
: freq_imp = sklearn.impute.SimpleImputer(strategy="most_frequent")
freq_mintemp = freq_imp.fit_transform(data[["MinTemp"]])
sns.distplot(freq_mintemp);
```



Как видно, эти три значения очень близки к нормальному распределению , поэтому выбираем любое из них на линии, здесь выбираем среднее

```
: data["MinTemp"] = mean_mintemp
```

3.2. Кодирование категориальных признаков

Рассмотрим колонку WindGustDir:

```
wgd = data["WindGustDir"].dropna().astype(str)
wgd.value_counts()
```

```
W      9915
SE     9418
N      9313
SSE    9216
E      9181
S      9168
WSW    9069
SW     8967
SSW    8736
WNW    8252
NW     8122
ENE    8104
ESE    7372
NE     7133
NNW    6620
NNE    6548
Name: WindGustDir, dtype: int64
```

Выполним кодирование категорий целочисленными значениями:

```
In [38]: le = sklearn.preprocessing.LabelEncoder()
wgd_le = le.fit_transform(wgd)
print(np.unique(wgd_le))
le.inverse_transform(np.unique(wgd_le))

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

Out[38]: array(['E', 'ENE', 'ESE', 'N', 'NE', 'NNE', 'NNW', 'NW', 'S', 'SE', 'SSE',
                'SSW', 'SW', 'W', 'WNW', 'WSW'], dtype=object)
```

Выполним кодирование категорий наборами бинарных значений:

```
wgd_oh = pd.get_dummies(wgd)
wgd_oh.head()
```

	E	ENE	ESE	N	NE	NNE	NNW	NW	S	SE	SSE	SSW	SW	W	WNW	WSW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

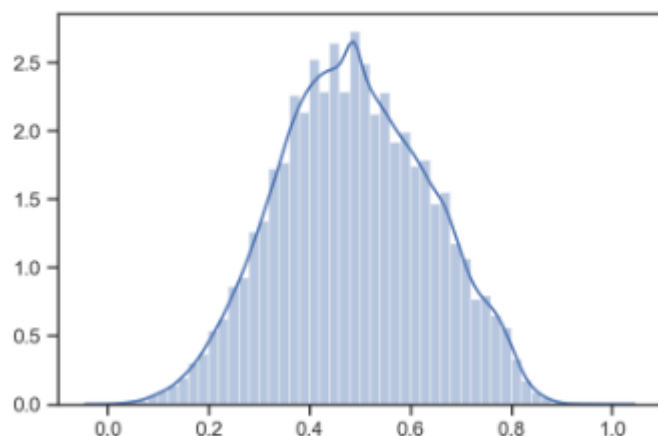
```
: wgd_oh[wgd_oh["W"] == 1].head()
```

	E	ENE	ESE	N	NE	NNE	NNW	NW	S	SE	SSE	SSW	SW	W	WNW	WSW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

3.3. Масштабирование данных

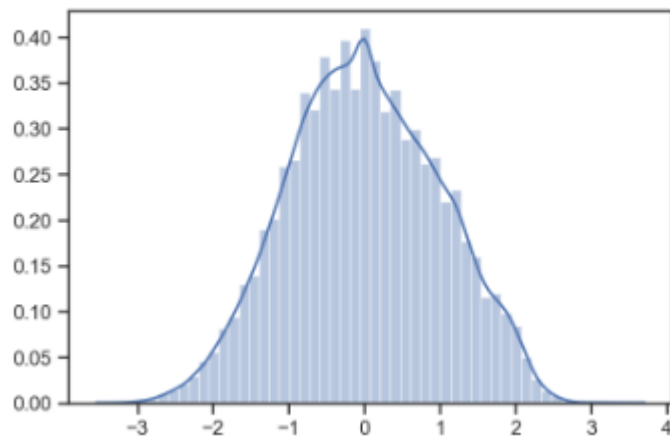
Для начала попробуем обычное MinMax-масштабирование:

```
: mm = sklearn.preprocessing.MinMaxScaler()
sns.distplot(mm.fit_transform(data[["MinTemp"]]));
```



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[["MinTemp"]]));
```



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

Список литературы

- [1] Гапанюк Ю. Е. Лабораторная работа «Обработка признаков» [Электронный ресурс]
https://github.com/ugapanyuk/ml_course_2021/wiki/LAB_MMO__FEATURES