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:
студент группы ИУ5-23М
Умряев Д. Т.

Москва — 2020 г.

1. Цель лабораторной работы

Изучить способы предварительной обработки данных для дальнейшего формирования моделей [1].

2. Задание

Требуется [1]:

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 (не менее 3 признаков);
   - кодирование категориальных признаков (не менее 3 признаков);
   - масштабирование данных (не менее 3 признаков).

3. Ход выполнения работы

3.1. Загрузка и первичный анализ данных

Подключим все необходимые библиотеки и настроим отображение графиков [2, 3]:

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.preprocessing import Normalizer

%matplotlib inline

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 А4 [4]:

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

Для выполнения данной лабораторной работы возьмём набор данных Melbourne Housing Market [5]:

[3]:
```
data = pd.read_csv("MELBOURNE_HOUSE_PRICES_LESS.csv")
```

[4]:
```
data.head()
```

[4]:
```
         Suburb           Address  Rooms Type      Price Method  \
0    Abbotsford     49 Lithgow St      3    h  1490000.0      S
1    Abbotsford     59A Turner St      3    h  1220000.0      S
2    Abbotsford     119B Yarra St      3    h  1420000.0      S
3    Aberfeldie        68 Vida St      3    h  1515000.0      S
4  Airport West  92 Clydesdale Rd      2    h   670000.0      S

    SellerG       Date  Postcode             Regionname  \
0    Jellis  1/04/2017      3067  Northern Metropolitan
1  Marshall  1/04/2017      3067  Northern Metropolitan
2    Nelson  1/04/2017      3067  Northern Metropolitan
3     Barry  1/04/2017      3040   Western Metropolitan
4    Nelson  1/04/2017      3042   Western Metropolitan

   Propertycount  Distance                CouncilArea
0           4019       3.0          Yarra City Council
1           4019       3.0          Yarra City Council
2           4019       3.0          Yarra City Council
3           1543       7.5  Moonee Valley City Council
4           3464      10.4  Moonee Valley City Council
```

[5]:
```
data.dtypes
```

[5]:
```
Suburb            object
Address           object
Rooms              int64
Type              object
Price            float64
Method            object
SellerG           object
Date              object
Postcode           int64
Regionname        object
Propertycount      int64
Distance         float64
CouncilArea       object
dtype: object
```

[6]:
```
data.shape
```

[6]:
```
(63023, 13)
```

3.2. Обработка пропусков в данных

Найдем все пропуски в данных:

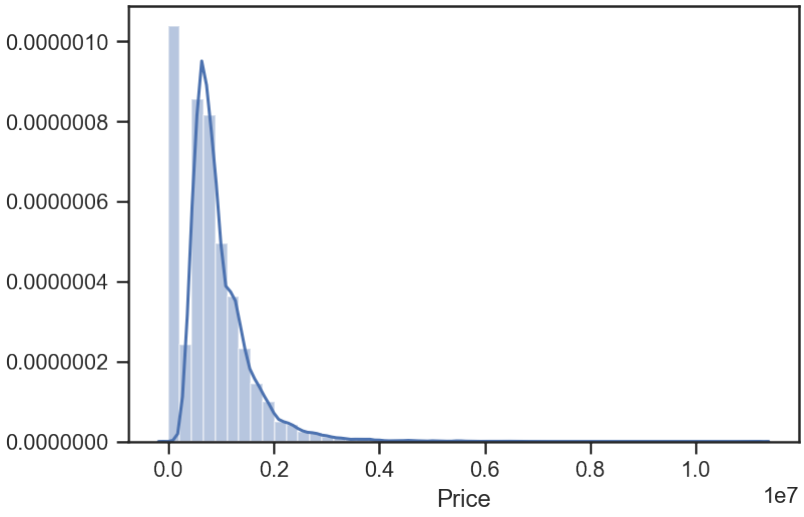
[7]:
```
data.isnull().sum()
```

[7]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price           14590
Method              0
SellerG             0
Date                0
Postcode            0
Regionname          0
Propertycount       0
Distance            0
CouncilArea         0
dtype: int64
```

Будем работать с колонкой `Price`.

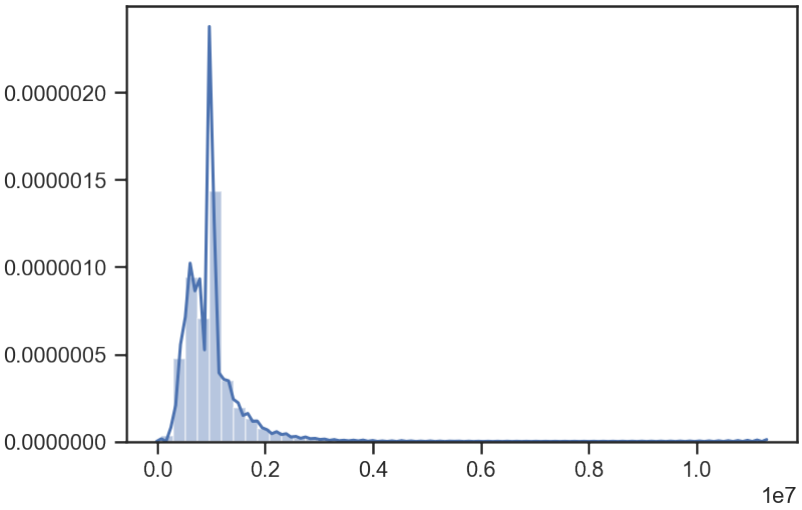
3.2.1. Заполнение пропусков нулями

[8]:
```
sns.distplot(data["Price"].fillna(0));
```



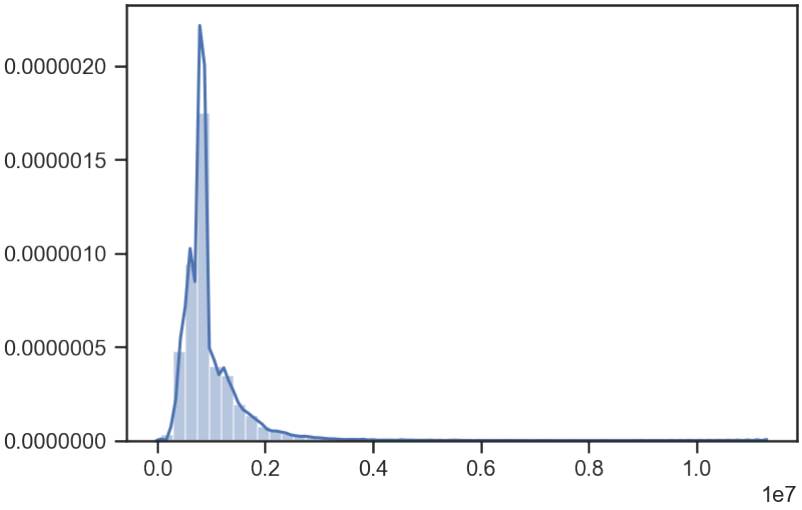
3.2.2. Заполнение пропусков средним значением

[9]:
```
mean_imp = SimpleImputer(strategy="mean")
mean_price = mean_imp.fit_transform(data[["Price"]])
sns.distplot(mean_price);
```



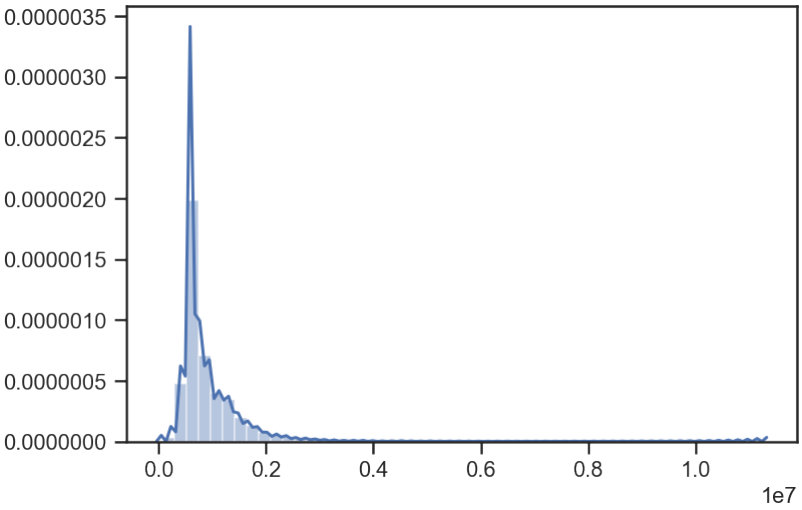
3.2.3. Заполнение пропусков медианным значением

[10]:
```
median_imp = SimpleImputer(strategy="median")
median_price = median_imp.fit_transform(data[["Price"]])
sns.distplot(median_price);
```



И последний вариант — наиболее частое значение:

[11]:
```
freq_imp = SimpleImputer(strategy="most_frequent")
freq_price = freq_imp.fit_transform(data[["Price"]])
sns.distplot(freq_price);
```



[12]:
```
data[["Price"]] = median_price
```

3.3. Кодирование категориальных признаков

Рассмотрим колонку Type:

[13]:
```
types = data[["Type"]]
types["Type"].unique()
```

[13]:
```
array(['h', 't', 'u'], dtype=object)
```

3.3.1. Кодирование категорий целочисленными значениями

[14]:
```
le = LabelEncoder()
type_le = le.fit_transform(types["Type"])
np.unique(type_le)
```

[14]:
```
array([0, 1, 2])
```

[15]:
```
le.inverse_transform(np.unique(type_le))
```

[15]:
```
array(['h', 't', 'u'], dtype=object)
```

3.3.2. Кодирование категорий наборами бинарных значений

[16]:
```
ohe = OneHotEncoder()
types_ohe = ohe.fit_transform(types)
```

[17]:
```
types.shape
```

[17]:
```
(63023, 1)
```

[18]:
```
types_ohe.shape
```

```
[18]: (63023, 3)
```

```
[19]: types_ohe.todense()[0:10]
```

```
[19]: matrix([[1., 0., 0.],
              [1., 0., 0.],
              [1., 0., 0.],
              [1., 0., 0.],
              [1., 0., 0.],
              [0., 1., 0.],
              [0., 0., 1.],
              [1., 0., 0.],
              [1., 0., 0.],
              [1., 0., 0.]])
```

```
[20]: types.head(10)
```

```
[20]:   Type
      0    h
      1    h
      2    h
      3    h
      4    h
      5    t
      6    u
      7    h
      8    h
      9    h
```

3.3.3. Pandas get_dummies - быстрый вариант one-hot кодирования

```
[21]: type_oh = pd.get_dummies(types)
      type_oh.head(10)
```

```
[21]:    Type_h  Type_t  Type_u
      0       1       0       0
      1       1       0       0
      2       1       0       0
      3       1       0       0
      4       1       0       0
      5       0       1       0
      6       0       0       1
      7       1       0       0
      8       1       0       0
      9       1       0       0
```

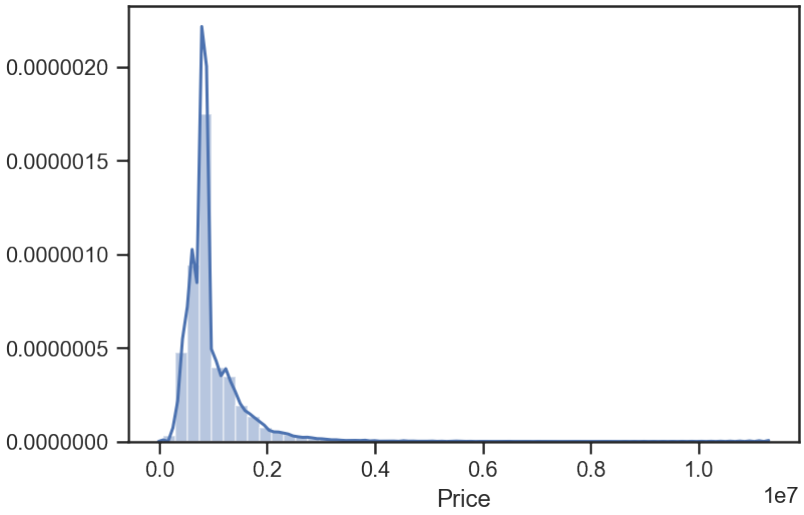
3.4. Масштабирование данных

Термины “масштабирование” и “нормализация” часто используются как синонимы. Масштабирование предполагает изменение диапазона измерения величины, а нормализация - из-

менение распределения этой величины.

[22]:
```
sns.distplot(data['Price'])
```

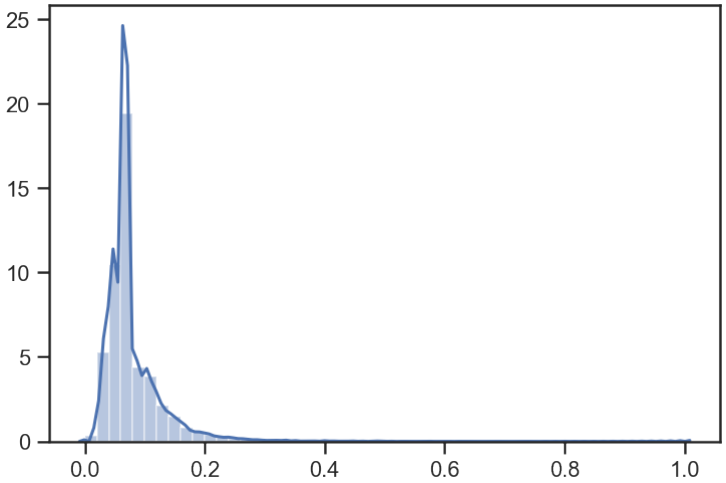
[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e1f06227c8>
```



3.4.1. MinMax-масштабирование

[23]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Price']])
sns.distplot(sc1_data)
```

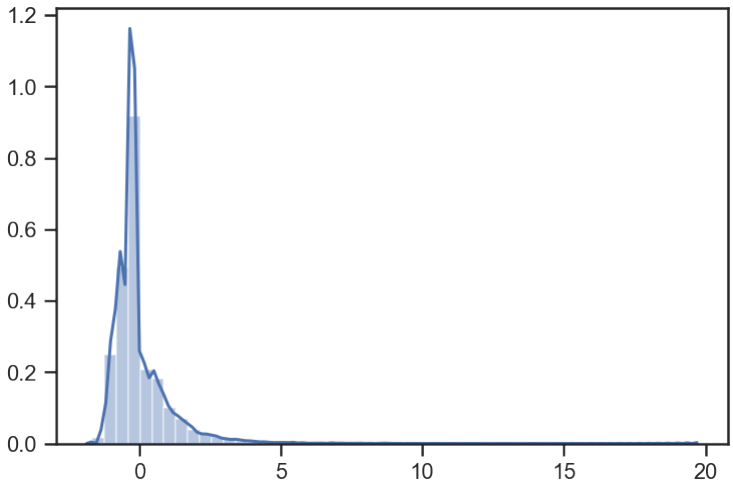
[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e1ee3119c8>
```



3.4.2. Масштабирование данных на основе Z-оценки

[24]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Price']])
sns.distplot(sc2_data)
```

[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e1f0d20748>
```



Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных» [Электронный ресурс] // GitHub. — 2020. — Режим доступа: `https://github.com/ugapanyuk/ml_course/wiki/LAB_MISSING` (дата обращения: 14.03.2020).

[2] Team The IPython Development. IPython 7.13.0 Documentation [Electronic resource] // Read the Docs. — 2020. — Access mode: `https://ipython.readthedocs.io/en/stable/` (online; accessed: 14.03.2020).

[3] Waskom M. seaborn 0.10.0 documentation [Electronic resource] // PyData. — 2020. — Access mode: `https://seaborn.pydata.org/` (online; accessed: 14.03.2020).

[4] pandas 1.0.1 documentation [Electronic resource] // PyData. — 2020. — Access mode: `http://pandas.pydata.org/pandas-docs/stable/` (online; accessed: 14.03.2020).

[5] Pino T. Melbourne Housing Market [Electronic resource] // Kaggle. — 2019. — Access mode: `https://www.kaggle.com/anthonypino/melbourne-housing-market` (online; accessed: 14.03.2020).