Московский государственный технический

университет им. Н. Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №2
«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»

Выполнил:
студент группы РТ5-61Б

Алиев Тимур

Подпись и дата:

Проверил:
доцент каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2022 г.

Описание задания

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.

Jupyter notebook

Загрузка и первичный анализ

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

Ввод [2]:
```
# размер набора данных
data = pd.read_csv('laptop.csv', sep=",")
```

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

Out[3]:
```
(130, 11)
```

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

Out[4]:
```
Unnamed: 0              int64
Brand                  object
Model                  object
Series                 object
Processor              object
Processor_Gen          object
RAM                   float64
Hard_Disk_Capacity     object
OS                     object
Rating                float64
Price                   int64
dtype: object
```

Ввод [5]:
```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

Out[5]:
```
Unnamed: 0             0
Brand                  0
Model                 14
Series                50
Processor              7
Processor_Gen          7
RAM                    8
Hard_Disk_Capacity     8
OS                     8
Rating                 0
Price                  0
dtype: int64
```

Ввод [6]:
```
# Первые 5 строк датасета
data.head()
```

Out[6]:

	Unnamed: 0	Brand	Model	Series	Processor	Processor_Gen	RAM	Hard_Disk_Capacity	OS	Rating	Price
0	0	DELL	Inspiron	NaN	i3	11th	8.0	1 TB HDD	Windows 11 Home	3.7	39040
1	1	DELL	Vostro	NaN	i5	11th	8.0	1 TB HDD	Windows 10 Home	3.6	50840
2	2	ASUS	VivoBook	15	i3	10th	8.0	512 GB SSD	Windows 11 Home	4.3	37940
3	3	DELL	Inspiron	NaN	i3	11th	8.0	1 TB HDD	256 GB SSD	4.4	44440
4	4	ASUS	TUF	Gaming	i5	10th	8.0	512 GB SSD	Windows 10 Home	4.5	57940

Ввод [7]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 130
```

Обработка пропусков в данных

Ввод [8]:
```
# Удаление строк, содержащих пустые значения
data_new = data.dropna(axis=0, how='any')
(data.shape, data_new.shape)
```

Out[8]:
```
((130, 11), (72, 11))
```

Ввод [9]:
```
data_new.head()
```

Out[9]:

	Unnamed: 0	Brand	Model	Series	Processor	Processor_Gen	RAM	Hard_Disk_Capacity	OS	Rating	Price
2	2	ASUS	VivoBook	15	i3	10th	8.0	512 GB SSD	Windows 11 Home	4.3	37940
4	4	ASUS	TUF	Gaming	i5	10th	8.0	512 GB SSD	Windows 10 Home	4.5	57940
5	5	ASUS	Ryzen	3	3250U	3rd	8.0	256 GB SSD	Windows 10 Home	4.3	35940
6	6	DELL	Inspiron	Athlon	3050U	-	4.0	256 GB SSD	Windows 11 Home	4.2	33940
8	8	Lenovo	IdeaPad	3	i3	10th	8.0	1 TB HDD	Windows 10 Home	4.1	37440

Преобразование категориальных признаков в числовые

Ввод [10]:
```
# one-hot кодирование (то есть кодирование бинарными значениями)
pd.get_dummies(data_new).head()
```

Out[10]:

	Unnamed: 0	RAM	Rating	Price	Brand_ASUS	Brand_DELL	Brand_HP	Brand_Lenovo	Brand_MICROSOFT	Brand_MSI	...	Hard_Disk_Capacity_32 GB EMMC Storage	Hard_Dis
2	2	8.0	4.3	37940	1	0	0	0	0	0	...	0	
4	4	8.0	4.5	57940	1	0	0	0	0	0	...	0	
5	5	8.0	4.3	35940	1	0	0	0	0	0	...	0	
6	6	4.0	4.2	33940	0	1	0	0	0	0	...	0	
8	8	8.0	4.1	37440	0	0	0	1	0	0	...	0	

5 rows × 102 columns

Ввод [11]:
```
from sklearn.preprocessing import LabelEncoder
```

Ввод [12]:
```
brand_enc = data_new.T
brand_enc = pd.DataFrame({'c1': brand_enc.T["Brand"]})
brand_enc
```

Out[12]:

	c1
2	ASUS
4	ASUS
5	ASUS
6	DELL
8	Lenovo
...	...
115	Lenovo
116	Lenovo
118	ASUS
128	ASUS
129	Lenovo

72 rows × 1 columns

Ввод [13]:
```
brand_enc['c1'].unique()
```

Out[13]:
```
array(['ASUS', 'DELL', 'Lenovo', 'HP', 'acer', 'MSI', 'realme',
       'MICROSOFT'], dtype=object)
```

Ввод [14]:
```
le = LabelEncoder()
brand_enc_le = le.fit_transform(brand_enc['c1'])
```

Ввод [14]:
```
le = LabelEncoder()
brand_enc_le = le.fit_transform(brand_enc['c1'])
```

Ввод [15]:
```
# Наименования категорий в соответствии с порядковыми номерами

# Свойство называется classes, потому что предполагается что мы решаем
# задачу классификации и каждое значение категории соответствует
# какому-либо классу целевого признака

le.classes_
```

Out[15]:
```
array(['ASUS', 'DELL', 'HP', 'Lenovo', 'MICROSOFT', 'MSI', 'acer',
       'realme'], dtype=object)
```

Ввод [16]:
```
brand_enc_le
```

Out[16]:
```
array([0, 0, 0, 1, 3, 3, 3, 1, 2, 3, 6, 0, 3, 0, 0, 3, 0, 2, 0, 3, 0, 3,
       3, 0, 2, 1, 1, 5, 0, 0, 3, 0, 6, 3, 0, 2, 2, 3, 0, 0, 7, 5, 0, 3,
       3, 0, 3, 3, 2, 1, 0, 2, 0, 3, 0, 2, 3, 4, 0, 0, 6, 2, 3, 0, 0, 3,
       3, 3, 3, 0, 0, 3])
```

Ввод [17]:
```
np.unique(brand_enc_le)
```

Out[17]:
```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

Ввод [18]:
```
# В этом примере видно, что перед кодированием
# уникальные значения признака сортируются в лексикографическом порядке
le.inverse_transform([0, 1, 2, 3])
```

Out[18]:
```
array(['ASUS', 'DELL', 'HP', 'Lenovo'], dtype=object)
```

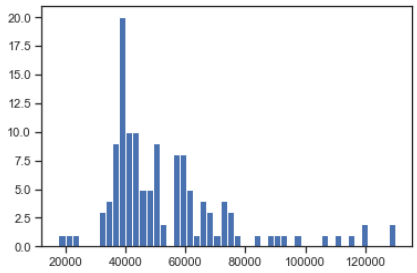
Масштабирование данных

MinMax масштабирование

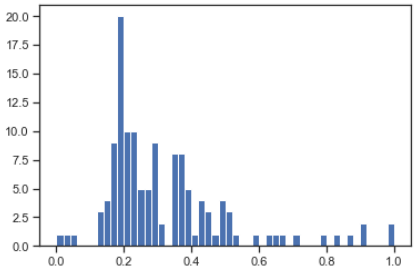
Ввод [19]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

Ввод [20]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Price']])
```

Ввод [21]:
```
plt.hist(data['Price'], 50)
plt.show()
```



Ввод [22]:
```
plt.hist(sc1_data, 50)
plt.show()
```



Масштабирование данных на основе Z-оценки - StandardScaler

Ввод [23]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Price']])
```

Ввод [24]:
```
plt.hist(sc2_data, 50)
plt.show()
```