Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет

имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №2

«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных**»**

Выполнил:

студент группы ИУ5-63Б

Иванченко Максим

Проверил:

к.т.н., доц., Ю. Е. Гапанюк

2024 г.

Обработка пропусков, кодирование категориальных признаков и масштабирование данных

```
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
```

Обзор данных

```
data = pd.read_csv('aug_train.csv')
data.shape
```

```
(19158, 14)
```

```
pd.concat({'Object type':data.dtypes, 'Null
count':data.isnull().sum()}, axis=1)
```

```
                        Object type  Null count
enrollee_id                   int64           0
city                         object           0
city_development_index      float64           0
gender                       object        4508
relevent_experience          object           0
enrolled_university          object         386
education_level              object         460
major_discipline             object        2813
experience                   object          65
company_size                 object        5938
company_type                 object        6140
last_new_job                 object         423
training_hours                int64           0
target                      float64           0
```

```
data.head()
```

```
   enrollee_id      city  city_development_index gender  \
0         8949  city_103                   0.920   Male
1        29725   city_40                   0.776   Male
2        11561   city_21                   0.624    NaN
3        33241  city_115                   0.789    NaN
4          666  city_162                   0.767   Male

       relevent_experience enrolled_university education_level  \
0  Has relevent experience       no_enrollment        Graduate
1   No relevent experience       no_enrollment        Graduate
2   No relevent experience    Full time course        Graduate
3   No relevent experience                 NaN        Graduate
4  Has relevent experience       no_enrollment         Masters
```

```
  major_discipline experience company_size    company_type
last_new_job  \
0             STEM        >20          NaN             NaN
1
1             STEM         15        50-99         Pvt Ltd
>4
2             STEM          5          NaN             NaN
never
3  Business Degree         <1          NaN         Pvt Ltd
never
4             STEM        >20        50-99  Funded Startup
4

   training_hours  target
0              36     1.0
1              47     0.0
2              83     0.0
3              52     1.0
4               8     0.0
```

Берем колонку 'enrolled_university' для заполнения пустых ячеек

```
data['enrolled_university'].unique()
```

```
array(['no_enrollment', 'Full time course', nan, 'Part time course'],
      dtype=object)
```

```
# количество для каждого значения столба без учета null
nonNullValues =
data['enrolled_university'].value_counts(ascending=False)
# количество null
nullValues = pd.Series({'nan':
data['enrolled_university'].isnull().sum()})

pd.concat([nonNullValues, nullValues], axis=0)
```

```
no_enrollment       13817
Full time course     3757
Part time course     1198
nan                   386
dtype: int64
```

Заполняем пропуски наиболее популярным значением

```
# создаем импьютор
imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
filledColumn = imp.fit_transform(data[['enrolled_university']])
filledColumn
```

```
array([['no_enrollment'],
       ['no_enrollment'],
       ['Full time course'],
       ...,
       ['no_enrollment'],
       ['no_enrollment'],
       ['no_enrollment']], dtype=object)
```

```
np.unique(filledColumn)
```

```
array(['Full time course', 'Part time course', 'no_enrollment'],
      dtype=object)
```

```
# ndarray to Series
filledColumnSeries = pd.Series(map(lambda x: x[0], filledColumn))
filledColumnSeries
```

```
0            no_enrollment
1            no_enrollment
2         Full time course
3            no_enrollment
4            no_enrollment
                ...
19153        no_enrollment
19154        no_enrollment
19155        no_enrollment
19156        no_enrollment
19157        no_enrollment
Length: 19158, dtype: object
```

```
# check if 'no_enrollment' number increases
filledColumnSeries.value_counts()
```

```
no_enrollment       14203
Full time course     3757
Part time course     1198
Name: count, dtype: int64
```

```
data['enrolled_university'] = filledColumnSeries
data['enrolled_university'].isnull().sum()
```

```
0
```

Кодирование категориального признака

```
edu_level = pd.DataFrame(data['education_level'])
edu_level.head()
```

```
  education_level
0        Graduate
1        Graduate
2        Graduate
```

```
3        Graduate
4         Masters
```

```
edu_level['education_level'].unique()
```

```
array(['Graduate', 'Masters', 'High School', nan, 'Phd', 'Primary
School'],
      dtype=object)
```

```
imp = SimpleImputer(missing_values=np.nan, strategy='constant',
fill_value='NA')
filledColumn = imp.fit_transform(edu_level)
np.unique(filledColumn)
```

```
array(['Graduate', 'High School', 'Masters', 'NA', 'Phd',
       'Primary School'], dtype=object)
```

```
edu_level['education_level'] = pd.Series(pd.Series(map(lambda x: x[0],
filledColumn)))
print(edu_level.head(), edu_level['education_level'].unique(), sep='\
n')
```

```
  education_level
0        Graduate
1        Graduate
2        Graduate
3        Graduate
4         Masters
['Graduate' 'Masters' 'High School' 'NA' 'Phd' 'Primary School']
```

```
d = {'NA': 0, 'Primary School': 1, 'High School': 2, 'Graduate': 3,
'Masters': 4, 'Phd': 5}
edu_level['coded'] = edu_level['education_level'].map(d)
edu_level.head(10)
```

```
  education_level  coded
0        Graduate      3
1        Graduate      3
2        Graduate      3
3        Graduate      3
4         Masters      4
5        Graduate      3
6     High School      2
7        Graduate      3
8        Graduate      3
9        Graduate      3
```

Масштабирование данных (другой датасет)

```
cars = pd.read_csv('used_cars_data.csv')
cars.head()
```

```
   S.No.                                  Name    Location  Year  \
0      0            Maruti Wagon R LXI CNG      Mumbai  2010
1      1  Hyundai Creta 1.6 CRDi SX Option        Pune  2015
2      2                      Honda Jazz V     Chennai  2011
3      3                 Maruti Ertiga VDI     Chennai  2012
4      4   Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013

   Kilometers_Driven Fuel_Type Transmission Owner_Type     Mileage
Engine  \
0              72000       CNG       Manual      First  26.6 km/kg
998 CC
1              41000    Diesel       Manual      First  19.67 kmpl
1582 CC
2              46000    Petrol       Manual      First   18.2 kmpl
1199 CC
3              87000    Diesel       Manual      First  20.77 kmpl
1248 CC
4              40670    Diesel    Automatic     Second   15.2 kmpl
1968 CC

       Power  Seats  New_Price  Price
0  58.16 bhp    5.0        NaN   1.75
1  126.2 bhp    5.0        NaN  12.50
2   88.7 bhp    5.0  8.61 Lakh   4.50
3  88.76 bhp    7.0        NaN   6.00
4  140.8 bhp    5.0        NaN  17.74
```

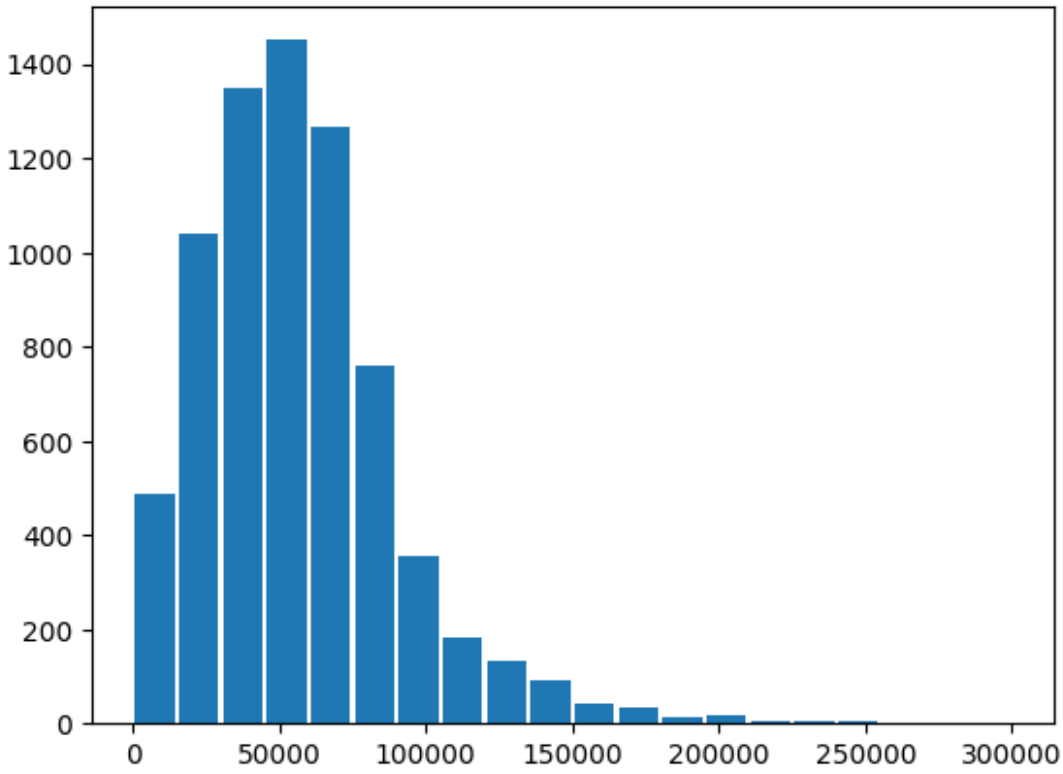
```
cars.shape
```

```
(7253, 14)
```

```
# Удаляем записи, где пробег сильно больше большинства
cars = cars[cars.Kilometers_Driven <= 300000]
cars.shape
```

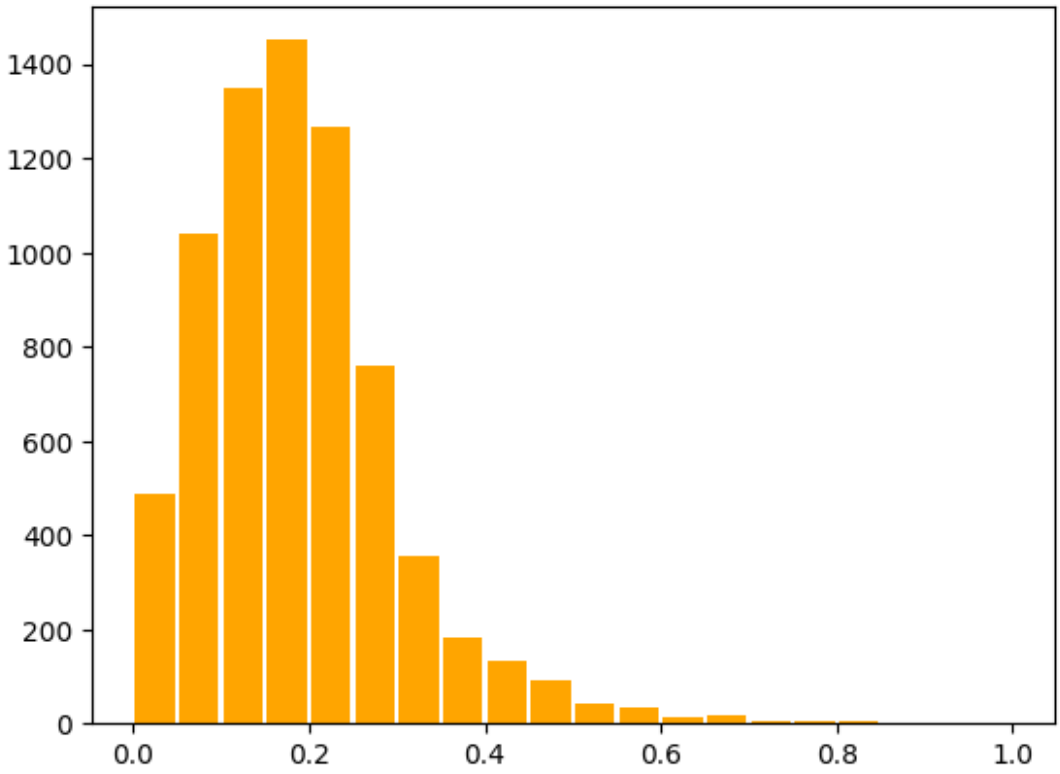
```
(7245, 14)
```

```
plt.hist(cars['Kilometers_Driven'], rwidth=0.9, bins=20)
plt.show()
```




```
scaler = MinMaxScaler()
scaledKilometersDriven =
scaler.fit_transform(cars[['Kilometers_Driven']])

plt.hist(scaledKilometersDriven, rwidth=0.9, bins=20, color='orange')
plt.show()
```


1400
1200
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0