Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

“Методы машинного обучения”

ЛАБОРАТОРНАЯ РАБОТА № 3. **«Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных»**

Студент группы ИУ5-24М

Петропавлов Д.М.

_____ Дата

_____ Подпись

Москва 2020

Цель лабораторной работы:

изучение способов предварительной обработки данных для дальнейшего формирования моделей.

Задание:

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:

a) обработку пропусков в данных (не менее 3 признаков);

b) кодирование категориальных признаков (не менее 3 признаков);

c) масштабирование данных (не менее 3 признаков).

In [13]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [14]:
```
data = pd.read_csv('C:/Dataset/dataset-limpo.csv', sep=",")
```

Размер набора данных:

In [15]:
```
total_count = data.shape
print('Всего строк: {}'.format(total_count[0]))
print('Всего колонок: {}'.format(total_count[1]))
```

```
Всего строк: 12899
Всего колонок: 36
```

1. Обработка пропусков в данных

Выберем колонки с пропущенными значениями

In [19]:
```
num_cols = []
total_count = data.shape[0]
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

```
Колонка bairro. Тип данных object. Количество пустых значений 1703, 13.2%.
Колонка descricao. Тип данных object. Количество пустых значений 1, 0.01%.
Колонка ip_address_origin. Тип данных object. Количество пустых значений 1194, 9.26%.
Колонка registrou_bo. Тип данных object. Количество пустых значений 524, 4.06%.
Колонка Bicicleta. Тип данных object. Количество пустых значений 12643, 98.02%.
Колонка Bolsa ou Mochila. Тип данных object. Количество пустых значений 9268, 71.85%.
Колонка Carteira. Тип данных object. Количество пустых значений 8656, 67.11%.
Колонка Cartão de Crédito. Тип данных object. Количество пустых значений 10123, 78.48%.
Колонка Celular. Тип данных object. Количество пустых значений 4522, 35.06%.
Колонка Computador. Тип данных object. Количество пустых значений 12787, 99.13%.
Колонка DVD. Тип данных object. Количество пустых значений 12805, 99.27%.
Колонка Dinheiro. Тип данных object. Количество пустых значений 11169, 86.59%.
Колонка Documentos. Тип данных object. Количество пустых значений 9239, 71.63%.
Колонка Equipamento de Som. Тип данных object. Количество пустых значений 12660, 98.15%.
Колонка Estepe. Тип данных object. Количество пустых значений 12575, 97.49%.
Колонка MP4 ou Ipod. Тип данных object. Количество пустых значений 12347, 95.72%.
Колонка Móveis. Тип данных object. Количество пустых значений 12854, 99.65%.
Колонка Notebook. Тип данных object. Количество пустых значений 12135, 94.08%.
Колонка Outros. Тип данных object. Количество пустых значений 7656, 59.35%.
Колонка Relógio. Тип данных object. Количество пустых значений 11549, 89.53%.
Колонка Som. Тип данных object. Количество пустых значений 12793, 99.18%.
Колонка Tablet. Тип данных object. Количество пустых значений 12456, 96.57%.
Колонка Tv. Тип данных object. Количество пустых значений 12721, 98.62%.
```

In [23]:
```
cat_temp_data_registr = data[['registrou_bo']]
cat_temp_data_registr['registrou_bo'].unique()
```

Out[23]:
```
array([True, False, nan], dtype=object)
```

Импьютация наиболее частыми значениями

In [24]:
```
imp1 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp1 = imp1.fit_transform(cat_temp_data_registr)
data_imp1
```

Out[24]:
```
array([[True],
       [True],
       [True],
       ...,
       [False],
       [True],
       [False]], dtype=object)
```

Пустые значения отсутствуют

In [25]:
```
np.unique(data_imp1)
```

Out[25]:
```
array([False, True], dtype=object)
```

Импьютация константой

In [31]:
```
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='192.0.0.1')
data_imp3 = imp3.fit_transform(cat_temp_data_ip)
data_imp3
```

Out[31]:
```
array([['192.0.0.1'],
       ['187.11.1.51'],
       ['189.19.160.82'],
       ...,
       ['177.103.229.244'],
       ['200.161.48.250'],
       ['177.32.220.159']], dtype=object)
```

Пустые значения отсутствуют

In [32]:
```
np.unique(data_imp3)
```

Out[32]:
```
array(['104.129.196.99', '104.129.198.64', '104.132.119.91', ...,
       '95.22.55.157', '95.233.181.88', '98.210.7.128'], dtype=object)
```

2. Кодирование категориальных признаков

2.1. Кодирование категорий целочисленными значениями

In [33]:
```
cat_enc = pd.DataFrame({'c1':data_imp1.T[0]})
cat_enc
```

Out[33]:

	c1
0	True
1	True
2	True
3	True
4	True
...	...
12894	False
12895	True
12896	False
12897	True
12898	False

12899 rows × 1 columns

In [35]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
cat_enc['c1'].unique()
```

Out[35]:
```
array([True, False], dtype=object)
```

In [36]:
```
np.unique(cat_enc_le)
```

Out[36]:
```
array([0, 1])
```

In [38]:
```
le.inverse_transform([ 0,  1])
```

Out[38]:
```
array([False, True], dtype=object)
```

2.2. Кодирование категорий наборами бинарных значений

In [39]:
```
cat_enc2 = pd.DataFrame({'c2':data_imp2.T[0]})
cat_enc2
```

Out[39]:

	c2
0	Butantã
1	Itaquera
2	Itaquera
3	Morumbi
4	Alto de Pinheiros
...	...
12894	Jardim das Camelias
12895	Jardim Paulista
12896	Vila Joao Ramalho
12897	Campo Belo
12898	Pinheiros

12899 rows × 1 columns

In [40]:
```
ohe = OneHotEncoder()
cat_enc2_ohe = ohe.fit_transform(cat_enc2[['c2']])
cat_enc2_ohe
```

Out[40]:
```
<12899x1577 sparse matrix of type '<class 'numpy.float64'>'
        with 12899 stored elements in Compressed Sparse Row format>
```

In [41]:
```
cat_enc2_ohe.todense()[0:10]
```

Out[41]:
```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

In [42]:
```
cat_enc2.head(10)
```

Out[42]:

	c2
0	Butantã
1	Itaquera
2	Itaquera
3	Morumbi
4	Alto de Pinheiros
5	Butantã
6	Sumaré
7	Vila Mariana
8	Barra Funda
9	Santana

2.3. Быстрый вариант one-hot кодирования

In [43]:
```
cat_enc3 = pd.DataFrame({'c3':data_imp3.T[0]})
cat_enc3
```

Out[43]:

	c3
0	192.0.0.1
1	187.11.1.51
2	189.19.160.82
3	199.67.140.46
4	187.92.158.154
...	...
12894	189.100.243.242
12895	189.38.212.209
12896	177.103.229.244
12897	200.161.48.250
12898	177.32.220.159

12899 rows × 1 columns

In [44]:
```
pd.get_dummies(cat_enc).head()
```

Out[44]:

	c1_False	c1_True
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1

In [46]:
```
pd.get_dummies(cat_temp_data_ip, dummy_na=True).head()
```

Out[46]:

	ip_address_origin_104.129.196.99	ip_address_origin_104.129.198.64	ip_address_origin_104.132.119.91	ip_address_origin_107.167.108.172	ip_address_origin_107.
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

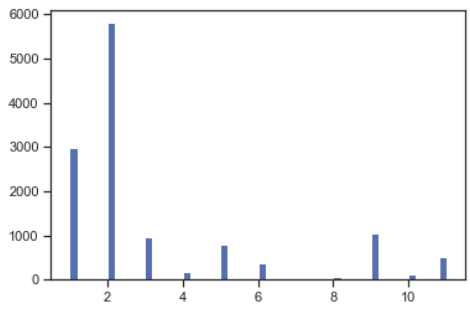
5 rows × 10433 columns

3. Масштабирование данных

In [47]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

3.1. MinMax масштабирование

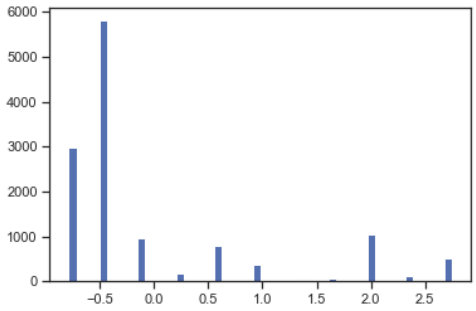
In [52]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['tipo_assalto_id']])
plt.hist(data['tipo_assalto_id'], 50)
plt.show()
```



3.2. Масштабирование данных на основе Z-оценки

In [51]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['tipo_assalto_id']])

plt.hist(sc2_data, 50)
plt.show()
```



3.3. Нормализация данных

In [53]:
```
sc3 = Normalizer()
sc3_data = sc3.fit_transform(data[['tipo_assalto_id']])

plt.hist(sc3_data, 50)
plt.show()
```