



Отчет
Лабораторная работа № 3
По курсу «Технологии машинного обучения»
«Обработка пропусков данных, кодирование категориальных признаков, масштабирование данных»

ИСПОЛНИТЕЛЬ:

Сергеев И.В.
Группа ИУ5-64Б

"__" _____ 2020 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

"__" _____ 2020 г.

Описание задания

Цель лабораторной работы - изучение способов предварительной обработки данных для дальнейшего формирования моделей.

Задание - 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.). Для выбранного датасета (датасетов) на основе материалов лекции решить следующие задачи:

- обработку пропусков в данных;
- кодирование категориальных признаков;
- масштабирование данных.

Текст программы

Программа разрабатывалась в IDE PyCharm. Ниже приведён полный листинг программы:

```
###

import numpy
import pandas
import seaborn
import matplotlib.pyplot as plt
%matplotlib inline
seaborn.set(style="ticks")
data = pandas.read_csv('dataset.csv', sep=",")
data.head()

###

print(data.shape)
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

###

data.isnull().sum()

###

# Here select empty numeric columns
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        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))

    ###

data_num = data[num_cols]
data_num

###

for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()

###

from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
cat_temp_data = data[['Product_Category_2']]
cat_temp_data.head()

###

cat_temp_data[cat_temp_data['Product_Category_2'].isnull()].shape

###

###

imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=0)
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3

###

numpy.unique(data_imp3)

###

data_imp3[data_imp3==0].size

###

sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Purchase']])

###

plt.hist(data['Purchase'], 50)
plt.show()

###

plt.hist(sc1_data, 50)
plt.show()

###

sc2 = StandardScaler()

```

```

sc2_data = sc2.fit_transform(data[['Purchase']])

#%%

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

#%%

temp_data = data[['Gender']]
temp_data.head()

#%%

temp_data['Gender'].unique()

#%%

temp_data[temp_data['Gender'].isnull()].shape

#%%

from sklearn.impute import SimpleImputer
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(temp_data)
data_imp2

#%%

numpy.unique(data_imp2)

#%%

cat_enc = pandas.DataFrame({'c1':data_imp2.T[0]})
cat_enc

#%%

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
cat_enc['c1'].unique()

#%%

numpy.unique(cat_enc_le)

#%%

le.inverse_transform([0, 1])

#%%

ohe = OneHotEncoder()
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])

#%%

cat_enc.shape

#%%

```

```
cat_enc_ohe.shape

#%%

cat_enc_ohe.todense()[0:10]

#%%

cat_enc.head(10)

#%%

pandas.get_dummies(cat_enc).head()
```

Примеры выполнения программы

Выполнение программы, а также наглядная демонстрация входных и выходных данных (таблиц, графиков и тд) осуществлялась на базе Jupyter Notebook, сервер которого запускался из-под PyCharm. Ниже приведены скриншоты, отражающие работу программы:

The screenshot displays a Jupyter Notebook interface with the following content:

Cell In [1]:

```
import numpy
import pandas
import seaborn
import matplotlib.pyplot as plt
%matplotlib inline
seaborn.set(style="ticks")
data = pandas.read_csv('dataset.csv', sep=",")
data.head()
```

Out[1]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	NaN
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN

Cell In [2]:

```
print(data.shape)
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

Out[2]:

```
(537577, 12)
```

Cell In [3]:

```
data.isnull().sum()
```

Out[3]:

```
User_ID      0
Product_ID    0
Gender        0
Age           0
Occupation    0
City_Category 0
Stay_In_Current_City_Years 0
Marital_Status 0
Product_Category_1 0
Product_Category_2 0
Product_Category_3 0
```

jupyter lab3 (autosaved) Python 3 Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Code

```
In [3]: data.isnull().sum()

Out[3]:
User_ID                0
Product_ID             0
Gender                 0
Age                   0
Occupation             0
City_Category          0
Stay_In_Current_City_Years  0
Marital_Status         0
Product_Category_1      0
Product_Category_2    166986
Product_Category_3    373299
Purchase               0
dtype: int64
```

```
In [4]: # Here select empty numeric columns
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count > 0 and (dt == 'float64' or dt == 'int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}'.format(col), 'Тип данных {}'.format(dt), 'Количество пустых значений {}'.format(temp_null_count), '{}%'.format(temp_perc))

Колонка Product_Category_2. Тип данных float64. Количество пустых значений 166986, 31.06%.
Колонка Product_Category_3. Тип данных float64. Количество пустых значений 373299, 69.44%.
```

```
In [5]: data_num = data[num_cols]
data_num

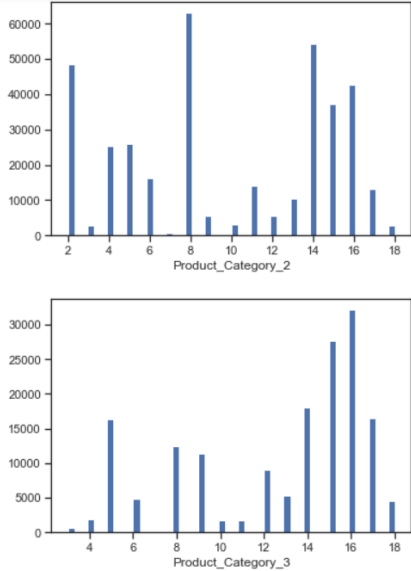
Out[5]:
Product_Category_2 Product_Category_3
```

jupyter lab3 (autosaved) Python 3 Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Code



```
In [7]: from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

```
In [7]: from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
cat_temp_data = data[['Product_Category_2']]
cat_temp_data.head()
```

Out[7]:

	Product_Category_2
0	NaN
1	6.0
2	NaN
3	14.0
4	NaN

```
In [8]: cat_temp_data[cat_temp_data['Product_Category_2'].isnull()].shape
```

Out[8]: (166986, 1)

In []:

```
In [9]: imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=0)
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

Out[9]: array([[0.],
[6.],
[0.],
...,
[15.],
[0.],
[8.]])

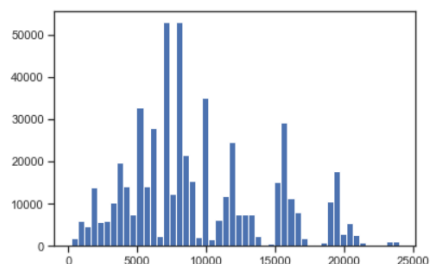
Out[10]: array([0., 2., 3., 4., 5., 6., 7., 8., 9., 10., 11., 12., 13.,
14., 15., 16., 17., 18.])

In [11]: data_imp3[data_imp3==0].size

Out[11]: 166986

```
In [12]: sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['Purchase']])
```

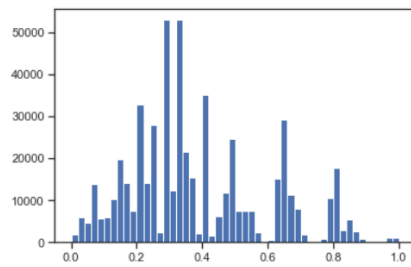
```
In [13]: plt.hist(data['Purchase'], 50)
plt.show()
```



```
In [14]: plt.hist(sc1_data, 50)
plt.show()
```

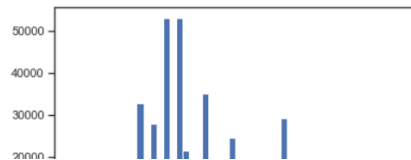


```
In [14]: plt.hist(sc1_data, 50)
plt.show()
```

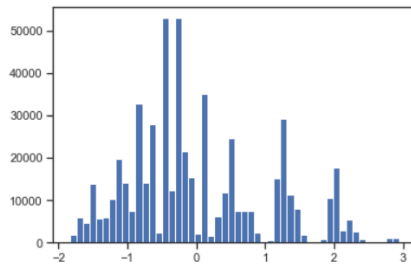


```
In [15]: sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Purchase']])
```

```
In [16]: plt.hist(sc2_data, 50)
plt.show()
```



```
In [16]: plt.hist(sc2_data, 50)
plt.show()
```



```
In [17]: temp_data = data[['Gender']]
temp_data.head()
```

Out[17]:

	Gender
0	F
1	F
2	F
3	F
4	M

```
In [18]: temp_data['Gender'].unique()
```

Out[18]: array(['F', 'M'], dtype=object)

jupyter lab3 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 ⌵

In [18]: `temp_data['Gender'].unique()`
 Out[18]: `array(['F', 'M'], dtype=object)`

In [19]: `temp_data[temp_data['Gender'].isnull()].shape`
 Out[19]: `(0, 1)`

In [20]: `from sklearn.impute import SimpleImputer`
`imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')`
`data_imp2 = imp2.fit_transform(temp_data)`
`data_imp2`
 Out[20]: `array([['F'],`
 `['F'],`
 `['F'],`
 `...,`
 `['M'],`
 `['M'],`
 `['M']], dtype=object)`

In [21]: `numpy.unique(data_imp2)`
 Out[21]: `array(['F', 'M'], dtype=object)`

In [22]: `cat_enc = pandas.DataFrame({'c1':data_imp2.T[0]})`
`cat_enc`
 Out[22]:

	c1
0	F
1	F
2	F

jupyter lab3 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 ⌵

In [23]: `from sklearn.preprocessing import LabelEncoder, OneHotEncoder`
`le = LabelEncoder()`
`cat_enc_le = le.fit_transform(cat_enc['c1'])`
`cat_enc['c1'].unique()`
 Out[23]: `array(['F', 'M'], dtype=object)`

In [24]: `numpy.unique(cat_enc_le)`
 Out[24]: `array([0, 1])`

In [25]: `le.inverse_transform([0, 1])`
 Out[25]: `array(['F', 'M'], dtype=object)`

In [26]: `ohe = OneHotEncoder()`
`cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])`

In [27]: `cat_enc.shape`
 Out[27]: `(537577, 1)`

In [28]: `cat_enc_ohe.shape`
 Out[28]: `(537577, 2)`

In [29]: `cat_enc_ohe.todense()[0:10]`
 Out[29]: `matrix([[1., 0.],`
 `[1., 0.],`
 `[1., 0.],`
 `[1., 0.],`
 `[0., 1.],`
 `...,`
 `[0., 1.],`
 `[0., 1.],`
 `[0., 1.]])`

```
In [30]: cat_enc.head(10)
```

```
Out[30]:
```

	c1
0	F
1	F
2	F
3	F
4	M
5	M
6	M
7	M
8	M
9	M

```
In [31]: pandas.get_dummies(cat_enc).head()
```

```
Out[31]:
```

	c1_F	c1_M
0	1	0
1	1	0
2	1	0
3	1	0
4	0	1

```
In [ ]:
```