Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчет по лабораторной работе №3

по курсу
Технологии Машинного Обучения

Выполнил:
Муравьев О.М.
ИУ5-62

_____

Проверил:
Гапанюк Ю.Е.

_____

Москва, 2019

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:

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

Код и результаты выполнения

Выбираем набор данных. В качестве набора данных возьму набор данных в котором описаны все пассажиры Титаника.

1. Подключим библиотеки:

```
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, Normalizer
%matplotlib inline
sns.set(style="ticks")
```

2. Подключаем набор данных

```
data = pd.read_csv('train.csv', sep=",")
```

3. Описание датасета:

Типы данных колонок

```
data.dtypes
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Проверим были ли пропущены значения в каких-нибудь колонках

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

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Размер датасета

```
data.shape
```

```
(891, 12)
```

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

```
Всего строк: 891
```

Первые пять строк датасета

```
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

4. Обработка пропущенных данных:

1.1.1. Удаление колонок содержащих пустые значения

```
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

```
((891, 12), (891, 9))
```

1.1.2. Удаление строк содержащих пустые значения

```
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

```
((891, 12), (183, 12))
```

1.1.2. Заполнение всех пропущенных значений нулями, что некорректно для категориальных знаичений ¶

```
data_new_3 = data.fillna(0)
data_new_3.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	0	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	0	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	0	S

1.2.1. Обработка пропусков в числовых данных

Выберем числовые колонки с пропущенными значениями

```
num_cols = [col for col in data.columns if (data[data[col].isnull()].shape[0] > 0 \
                                            and (data[col].dtype=='float64' or data[col].dtype=='int64'))]
for col in num_cols:
    print(f"Колонка {col}, количество пропусков {data[col].isnull().sum()} - \
    {round((data[col].isnull().sum()/total_count)*100,2)}%")
```

```
Колонка Age, количество пропусков 177 - 19.87%
```

```
strategies=['mean', 'median','most_frequent']
```

т.е. нашими стратегиями будут:

- Среднее значение
- Медиана
- Наиболее часто встречающаяся величина

Определим функцию для импьютации в которую будет отправляться название стратегии как аргумент

```
def test_num_impute(strat):
    # Определяем сратегию
    imp_num = SimpleImputer(strategy=strat)
    data_num_imp = imp_num.fit_transform(data_num_age)
    return data_num_imp[mask_missing_values_only]
```

1.2.2. Обработка пропусков в категориальных данных

Выберем категорильные колонки с пропущенными значениями

```
: cat_cols = [col for col in data.columns if (data[data[col].isnull()].shape[0] > 0 \
                                              and data[col].dtype=='object')]
  for col in cat_cols:
      print(f"Колонка {col}, количество пропусков {data[col].isnull().sum()} - \
      {round((data[col].isnull().sum()/total_count)*100,2)}%")
```

```
Колонка Cabin, количество пропусков 687 - 77.1%
Колонка Embarked, количество пропусков 2 - 0.22%
```

Получим уникальные значения для колонки

```
cat_temp_data['Cabin'].unique()
```

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

Импьютация наиболее частыми выражениями

```
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

Импьютация константой

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

Проверим, что пустые значения отсутствуют

```
np.unique(data_imp2)
```

```
array(['A10', 'A14', 'A16', 'A19', 'A20', 'A23', 'A24', 'A26', 'A31',
       'A32', 'A34', 'A36', 'A5', 'A6', 'A7', 'B101', 'B102', 'B18',
       'B19', 'B20', 'B22', 'B28', 'B3', 'B30', 'B35', 'B37', 'B38',
       'B39', 'B4', 'B41', 'B42', 'B49', 'B5', 'B50', 'B51 B53 B55',
       'B57 B59 B63 B66', 'B58 B60', 'B69', 'B71', 'B73', 'B77', 'B78',
       'B79', 'B80', 'B82 B84', 'B86', 'B94', 'B96 B98', 'C101', 'C103',
       'C104', 'C106', 'C110', 'C111', 'C118', 'C123', 'C124', 'C125',
       'C126', 'C128', 'C148', 'C2', 'C22 C26', 'C23 C25 C27', 'C30',
       'C32', 'C45', 'C46', 'C47', 'C49', 'C50', 'C52', 'C54', 'C62 C64',
       'C65', 'C68', 'C7', 'C70', 'C78', 'C82', 'C83', 'C85', 'C86',
       'C87', 'C90', 'C91', 'C92', 'C93', 'C95', 'C99', 'D', 'D10 D12',
       'D11', 'D15', 'D17', 'D19', 'D20', 'D21', 'D26', 'D28', 'D30',
       'D33', 'D35', 'D36', 'D37', 'D45', 'D46', 'D47', 'D48', 'D49',
       'D50', 'D56', 'D6', 'D7', 'D9', 'E10', 'E101', 'E12', 'E121',
       'E17', 'E24', 'E25', 'E31', 'E33', 'E34', 'E36', 'E38', 'E40',
       'E44', 'E46', 'E49', 'E50', 'E58', 'E63', 'E67', 'E68', 'E77',
       'E8', 'F E69', 'F G63', 'F G73', 'F2', 'F33', 'F38', 'F4', 'G6',
       'T'], dtype=object)
```

```
np.unique(data_imp3)
```

```
array(['A10', 'A14', 'A16', 'A19', 'A20', 'A23', 'A24', 'A26', 'A31',
       'A32', 'A34', 'A36', 'A5', 'A6', 'A7', 'B101', 'B102', 'B18',
       'B19', 'B20', 'B22', 'B28', 'B3', 'B30', 'B35', 'B37', 'B38',
       'B39', 'B4', 'B41', 'B42', 'B49', 'B5', 'B50', 'B51 B53 B55',
       'B57 B59 B63 B66', 'B58 B60', 'B69', 'B71', 'B73', 'B77', 'B78',
       'B79', 'B80', 'B82 B84', 'B86', 'B94', 'B96 B98', 'C101', 'C103',
       'C104', 'C106', 'C110', 'C111', 'C118', 'C123', 'C124', 'C125',
       'C126', 'C128', 'C148', 'C2', 'C22 C26', 'C23 C25 C27', 'C30',
       'C32', 'C45', 'C46', 'C47', 'C49', 'C50', 'C52', 'C54', 'C62 C64',
       'C65', 'C68', 'C7', 'C70', 'C78', 'C82', 'C83', 'C85', 'C86',
       'C87', 'C90', 'C91', 'C92', 'C93', 'C95', 'C99', 'D', 'D10 D12',
       'D11', 'D15', 'D17', 'D19', 'D20', 'D21', 'D26', 'D28', 'D30',
       'D33', 'D35', 'D36', 'D37', 'D45', 'D46', 'D47', 'D48', 'D49',
       'D50', 'D56', 'D6', 'D7', 'D9', 'E10', 'E101', 'E12', 'E121',
       'E17', 'E24', 'E25', 'E31', 'E33', 'E34', 'E36', 'E38', 'E40',
       'E44', 'E46', 'E49', 'E50', 'E58', 'E63', 'E67', 'E68', 'E77',
       'E8', 'F E69', 'F G63', 'F G73', 'F2', 'F33', 'F38', 'F4', 'G6',
       'MyConst', 'T'], dtype=object)
```

5. Замена категориальных данных на числовые:

2.1 Label encoding - кодирование целыми значениями

```
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['Cabin'])
```

```
cat_enc['Cabin'].unique()
```

```
array(['B96 B98', 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'E10', 'E44', 'A34',
       'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14', 'B37',
       'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38', 'B39',
       'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68', 'B41',
       'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48', 'E58',
       'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

```
np.unique(cat_enc_le)
```

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146])
```

2.2. One-hot encoding - Кодирование наборами бинарных значений

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

```
cat_enc.shape
```

```
(891, 1)
```

```
cat_enc_ohe.shape
```

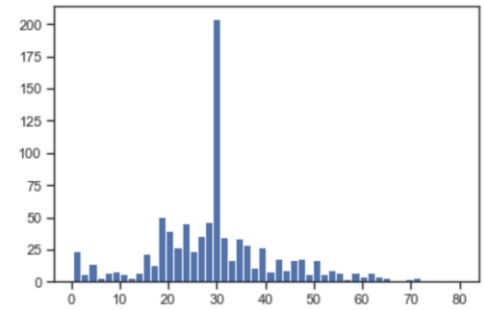
```
(891, 147)
```

6. Масштабирование данных

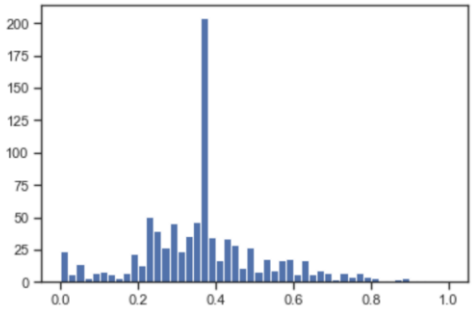
4.1. MinMax - масштабирование

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

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



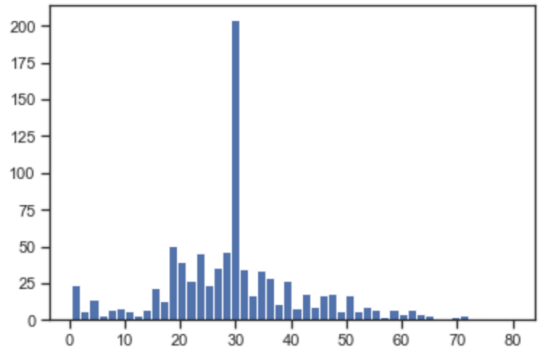

```
plt.hist(sc1_data, 50)
plt.show()
```



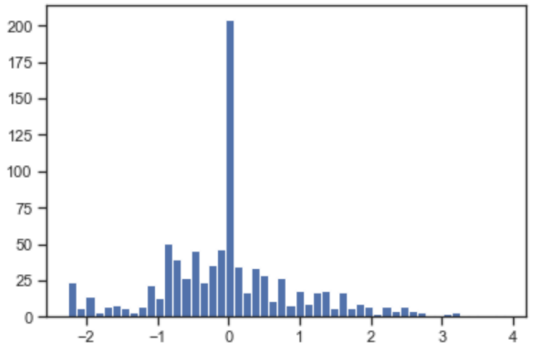
4.2. Z-оценка - StandartScaling

```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['Age']])
```

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



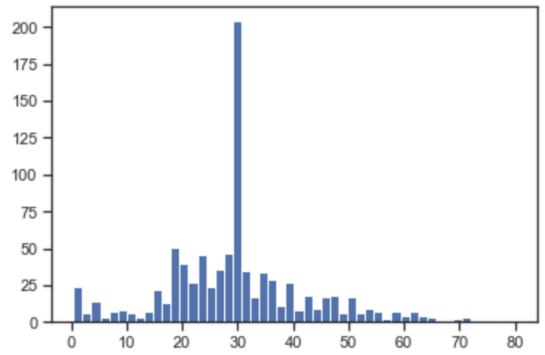

```
plt.hist(sc2_data,50)
plt.show()
```



7. Нормализация

```
sc3 = Normalizer()
sc3_data = sc3.fit_transform(data[['Age']])
```

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




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