

Московский государственный технический университет им. Н.Э. Баумана  
Факультет «Информатика и системы управления»  
Кафедра «Системы обработки информации и управления»



**Отчет**

**Лабораторные работа № 3**

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

**По курсу «Технологии машинного обучения»**

**ИСПОЛНИТЕЛЬ:**

Матиенко Андрей

Группа ИУ5-61Б

\_\_\_\_\_ 2020 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

\_\_\_\_\_ 2020 г.

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

**Решение:**

```
housing = load_housing_data()
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

## Кодирование категориальных признаков

### Категориальные признаки

```
In [91]: housing['ocean_proximity'].value_counts()
```

```
Out[91]: <1H OCEAN      9136
         INLAND      6551
         NEAR OCEAN   2658
         NEAR BAY     2290
         ISLAND         5
         Name: ocean_proximity, dtype: int64
```

## Кодирование с помощью LabelEncoder

```
In [92]: from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()  
cat_enc_le = le.fit_transform(housing['ocean_proximity'])  
cat_enc_le
```

```
Out[92]: array([3, 3, 3, ..., 1, 1, 1])
```

```
In [93]: np.unique(cat_enc_le)
```

```
Out[93]: array([0, 1, 2, 3, 4])
```

```
In [94]: le.inverse_transform([0, 1, 2, 3, 4])
```

```
Out[94]: array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],  
              dtype=object)
```

---

## Кодирование с помощью OneHotEncoder

```
In [96]: from sklearn.preprocessing import OneHotEncoder
```

```
ohe = OneHotEncoder()  
cat_ohe = ohe.fit_transform(housing[['ocean_proximity']])
```

```
In [97]: ohe.categories_
```

```
Out[97]: [array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],  
              dtype=object)]
```

```
In [98]: cat_ohe.toarray()  
#cat_ohe.A
```

```
Out[98]: array([[0., 0., 0., 1., 0.],  
               [0., 0., 0., 1., 0.],  
               [0., 0., 0., 1., 0.],  
               ...,  
               [0., 1., 0., 0., 0.],  
               [0., 1., 0., 0., 0.],  
               [0., 1., 0., 0., 0.]])
```

---

# Обработка пропусков в данных

## Первый способ определить признаки с нулевыми значениями

**total\_bedrooms** имеет 20433 ненулевых объекта из 20640

```
In [100]: housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 11 columns):
longitude           20640 non-null float64
latitude            20640 non-null float64
housing_median_age  20640 non-null float64
total_rooms         20640 non-null float64
total_bedrooms      20433 non-null float64
population          20640 non-null float64
households          20640 non-null float64
median_income       20640 non-null float64
median_house_value  20640 non-null float64
ocean_proximity     20640 non-null object
ocean_proximity_le  20640 non-null int32
dtypes: float64(9), int32(1), object(1)
memory usage: 1.7+ MB
```

## Второй способ

```
In [101]: housing.isnull().sum()
```

```
Out[101]: longitude           0
latitude            0
housing_median_age   0
total_rooms          0
total_bedrooms       207
population           0
households           0
median_income        0
median_house_value   0
ocean_proximity      0
ocean_proximity_le   0
dtype: int64
```

```
In [102]: sample_incomplete_rows = housing[housing.isnull().any(axis=1)].head()
sample_incomplete_rows
```

```
Out[102]:
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	oc
290	-122.16	37.77	47.0	1256.0	NaN	570.0	218.0	4.3750	161900.0	NEAR BAY	
341	-122.17	37.75	38.0	992.0	NaN	732.0	259.0	1.6196	85100.0	NEAR BAY	
538	-122.28	37.78	29.0	5154.0	NaN	3741.0	1273.0	2.5762	173400.0	NEAR BAY	
663	-122.24	37.75	45.0	891.0	NaN	384.0	146.0	4.9489	247100.0	NEAR BAY	
696	-122.10	37.69	41.0	746.0	NaN	387.0	161.0	3.9063	178400.0	NEAR BAY	

## Первый способ решить эту проблему

Удалить строки с нулевыми значениями

```
In [103]: sample_incomplete_rows.dropna(subset=['total_bedrooms'])
```

```
Out[103]:
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	ocean_proximity_le
290	-122.16	37.77	47.0	1256.0	570.0	218.0	4.3750	161900.0	NEAR BAY	3	
341	-122.17	37.75	38.0	992.0	732.0	259.0	1.6196	85100.0	NEAR BAY	3	
538	-122.28	37.78	29.0	5154.0	3741.0	1273.0	2.5762	173400.0	NEAR BAY	3	
563	-122.24	37.75	45.0	891.0	384.0	146.0	4.9489	247100.0	NEAR BAY	3	
696	-122.10	37.69	41.0	746.0	387.0	161.0	3.9063	178400.0	NEAR BAY	3	

## Второй способ

Удалить столбцы, у которых есть нулевые значения(пропуски)

```
In [104]: sample_incomplete_rows.drop("total_bedrooms", axis=1)
```

```
Out[104]:
```

	longitude	latitude	housing_median_age	total_rooms	population	households	median_income	median_house_value	ocean_proximity	ocean_proximity_le
290	-122.16	37.77	47.0	1256.0	570.0	218.0	4.3750	161900.0	NEAR BAY	3
341	-122.17	37.75	38.0	992.0	732.0	259.0	1.6196	85100.0	NEAR BAY	3
538	-122.28	37.78	29.0	5154.0	3741.0	1273.0	2.5762	173400.0	NEAR BAY	3
563	-122.24	37.75	45.0	891.0	384.0	146.0	4.9489	247100.0	NEAR BAY	3
696	-122.10	37.69	41.0	746.0	387.0	161.0	3.9063	178400.0	NEAR BAY	3

## Третий способ

Заменить нулевые (пустые) значения средним/медианой/самой частой величиной

```
In [106]: mean_ = housing['total_bedrooms'].mean()
sample_incomplete_rows['total_bedrooms'].fillna(mean_, inplace=True)
sample_incomplete_rows
```

```
Out[106]:
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	ocean_proximity_le
290	-122.16	37.77	47.0	1256.0	537.870553	570.0	218.0	4.3750	161900.0	NEAR BAY	3
341	-122.17	37.75	38.0	992.0	537.870553	732.0	259.0	1.6196	85100.0	NEAR BAY	3
538	-122.28	37.78	29.0	5154.0	537.870553	3741.0	1273.0	2.5762	173400.0	NEAR BAY	3
563	-122.24	37.75	45.0	891.0	537.870553	384.0	146.0	4.9489	247100.0	NEAR BAY	3
696	-122.10	37.69	41.0	746.0	537.870553	387.0	161.0	3.9063	178400.0	NEAR BAY	3

## Масштабирование данных

```
In [115]: from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
housing_1 = housing.copy()
housing_1.drop(['ocean_proximity'], axis=1, inplace=True)
housing_1 = scaler.fit_transform(housing_1)
df = pd.DataFrame(housing_1)
df
```

```
Out[115]:
```

	0	1	2	3	4	5	6	7	8	9
0	-1.327835	1.052548	0.982143	-0.804819	-0.970325	-0.974429	-0.977033	2.344766	2.129631	1.291089
1	-1.322844	1.043185	-0.607019	2.045890	1.348276	0.861439	1.669961	2.332238	1.314156	1.291089
2	-1.332827	1.038503	1.856182	-0.535746	-0.825561	-0.820777	-0.843637	1.782699	1.258693	1.291089
3	-1.337818	1.038503	1.856182	-0.624215	-0.718768	-0.766028	-0.733781	0.932968	1.165100	1.291089
4	-1.337818	1.038503	1.856182	-0.462404	-0.611974	-0.759847	-0.629157	-0.012881	1.172900	1.291089
...	...	...	...	...	...	...	...	...	...	...
20635	-0.758826	1.801647	-0.289187	-0.444985	-0.388895	-0.512592	-0.443449	-1.216128	-1.115804	-0.116739
20636	-0.818722	1.806329	-0.845393	-0.888704	-0.920488	-0.944405	-1.008420	-0.691593	-1.124470	-0.116739
20637	-0.823713	1.778237	-0.924851	-0.174995	-0.125472	-0.369537	-0.174042	-1.142593	-0.992746	-0.116739
20638	-0.873626	1.778237	-0.845393	-0.355600	-0.305834	-0.604429	-0.393753	-1.054583	-1.058608	-0.116739
20639	-0.833696	1.750146	-1.004309	0.068408	0.185416	-0.033977	0.079672	-0.780129	-1.017878	-0.116739

20640 rows × 10 columns