Лабораторная работа №2

Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных.

Выполнил: Пакало А. С., РТ5-61Б

Импортирование необходимых библиотек, подготовка окружения

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

Текстовое описание набора данных

В качестве набора данных был выбран датасет San Francisco Building Permits

Описание аттрибутов:

In [ ]:
```
attribute_description = pd.read_excel('data/BuildingPermitsAttributeDescription.xlsx')
attribute_description.drop('Sl No', inplace=True, axis=1)
attribute_description
```

Out[ ]:

	Column name	Description
0	Permit Number	Number assigned while filing
1	Permit Type	Type of the permit represented numerically.
2	Permit Type Definition	Description of the Permit type, for example\n ...
3	Permit Creation Date	Date on which permit created, later than \nor ...
4	Block	Related to address
5	Lot	Related to address
6	Street Number	Related to address
7	Street Number Suffix	Related to address
8	Street Name	Related to address
9	Street Name Suffix	Related to address
10	Unit	Unit of a building
11	Unit suffix	Suffix if any, for the unit
12	Description	Details about purpose of the permit.\n Example...
13	Current Status	Current status of the permit application.
14	Current Status Date	Date at which current status was entered
15	Filed Date	Filed date for the permit
16	Issued Date	Issued date for the permit
17	Completed Date	The date on which project was completed, \napp...
18	First Construction Document Date	Date on which construction was documented
19	Structural Notification	Notification to meet some legal need, given or...
20	Number of Existing Stories	Number of existing stories in the building. \n...
21	Number of Proposed Stories	Number of proposed stories for the constructio...
22	Voluntary Soft-Story \nRetrofit	Soft story to meet earth quake regulations
23	Fire Only Permit	Fire hazard prevention related permit
24	Permit Expiration Date	Expiration date related to issued permit.
25	Estimated Cost	Initial estimation of the cost of the project
26	Revised Cost	Revised estimation of the cost of the project
27	Existing Use	Existing use of the building
28	Existing Units	Existing number of units
29	Proposed Use	Proposed use of the building
30	Proposed Units	Proposed number of units
31	Plansets	Plan representation indicating the general des...
32	TIDF Compliance	TIDF compliant or not, this is a new legal req...

	Column name	Description
33	Existing Construction Type	Construction type, existing,as categories \nre...
34	Existing Construction Type Description	Description of the above, for example, \nwood ...
35	Proposed Construction Type	Construction type, proposed, as categories\n r...
36	Proposed Construction Type Description	Description of the above
37	Site Permit	Permit for site
38	Supervisor District	Supervisor District to which the building loca...
39	Neighborhoods - Analysis Boundaries	Neighborhood to which the building location be...
40	Zipcode	Zipcode of building address
41	Location	Location in latitude, longitude pair.
42	Record ID	Some ID, not useful for this

Загрузка набора данных

In [ ]:
```
# Had error about mixed dtypes of cols 22 and 32.
data = pd.read_csv('data/BuildingPermits.csv', sep=",", dtype={'Voluntary Soft-Story R
```

Основные характеристики датасета

In [ ]:
```
data.head()
```

Out[ ]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	St	...
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	Av	...
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	St	...

5 rows × 43 columns

In [ ]:
```
data.tail()
```

Out[ ]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name
198895	M862628	8	otc alterations permit	12/05/2017	0113	017A	1228	NaN	Montgomery
198896	201712055595	8	otc alterations permit	12/05/2017	0271	014	580	NaN	Bush
198897	M863507	8	otc alterations permit	12/06/2017	4318	019	1568	NaN	Indiana
198898	M863747	8	otc alterations permit	12/06/2017	0298	029	795	NaN	Sutter
198899	M864287	8	otc alterations permit	12/07/2017	0160	006	838	NaN	Pacific

5 rows × 43 columns

Размер датасета (кол-во строк, кол-во колонок)

In [ ]:
```
num_of_rows, num_of_columns = data.shape
print(f'Размер датасета: {num_of_rows} строк, {num_of_columns} колонок')
```

```
Размер датасета: 198900 строк, 43 колонок
```

Определение типов

In [ ]:
```
data.dtypes
```

Out[ ]:
```
Permit Number                                object
Permit Type                                   int64
Permit Type Definition                       object
Permit Creation Date                         object
Block                                        object
Lot                                          object
Street Number                                 int64
Street Number Suffix                         object
Street Name                                  object
Street Suffix                                object
Unit                                        float64
Unit Suffix                                  object
Description                                  object
Current Status                               object
Current Status Date                          object
Filed Date                                   object
Issued Date                                  object
Completed Date                               object
First Construction Document Date             object
Structural Notification                      object
Number of Existing Stories                  float64
Number of Proposed Stories                  float64
Voluntary Soft-Story Retrofit                object
Fire Only Permit                             object
Permit Expiration Date                       object
Estimated Cost                              float64
Revised Cost                                float64
Existing Use                                 object
Existing Units                              float64
Proposed Use                                 object
Proposed Units                              float64
Plansets                                    float64
TIDF Compliance                              object
Existing Construction Type                  float64
Existing Construction Type Description       object
Proposed Construction Type                  float64
Proposed Construction Type Description       object
Site Permit                                  object
Supervisor District                         float64
Neighborhoods - Analysis Boundaries          object
Zipcode                                     float64
Location                                     object
Record ID                                     int64
dtype: object
```

Некоторые колонки имеют неверные типы данных, их следует преобразовать.

In [ ]:
```
data = data.astype({'Zipcode': 'category'})
data.dtypes
```

Out[ ]:
```
Permit Number                                object
Permit Type                                   int64
Permit Type Definition                       object
Permit Creation Date                         object
Block                                        object
Lot                                          object
Street Number                                 int64
Street Number Suffix                         object
Street Name                                  object
Street Suffix                                object
Unit                                        float64
Unit Suffix                                  object
Description                                  object
Current Status                               object
Current Status Date                          object
Filed Date                                   object
Issued Date                                  object
Completed Date                               object
First Construction Document Date             object
Structural Notification                      object
Number of Existing Stories                  float64
Number of Proposed Stories                  float64
Voluntary Soft-Story Retrofit                object
Fire Only Permit                             object
Permit Expiration Date                       object
Estimated Cost                              float64
Revised Cost                                float64
Existing Use                                 object
Existing Units                              float64
Proposed Use                                 object
Proposed Units                              float64
Plansets                                    float64
TIDF Compliance                              object
Existing Construction Type                  float64
Existing Construction Type Description       object
Proposed Construction Type                  float64
Proposed Construction Type Description       object
Site Permit                                  object
Supervisor District                         float64
Neighborhoods - Analysis Boundaries          object
Zipcode                                    category
Location                                     object
Record ID                                     int64
dtype: object
```

Проверка на наличие пустых значений

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

Out[ ]:
```
Permit Number                                0
Permit Type                                  0
Permit Type Definition                       0
Permit Creation Date                         0
Block                                        0
Lot                                          0
Street Number                                0
Street Number Suffix                    196684
Street Name                                  0
Street Suffix                             2768
Unit                                    169421
Unit Suffix                             196939
Description                                290
Current Status                               0
Current Status Date                          0
Filed Date                                   0
Issued Date                              14940
Completed Date                          101709
First Construction Document Date         14946
Structural Notification                 191978
Number of Existing Stories               42784
Number of Proposed Stories               42868
Voluntary Soft-Story Retrofit           198865
Fire Only Permit                        180073
Permit Expiration Date                   51880
Estimated Cost                           38066
Revised Cost                              6066
Existing Use                             41114
Existing Units                           51538
Proposed Use                             42439
Proposed Units                           50911
Plansets                                 37309
TIDF Compliance                         198898
Existing Construction Type               43366
Existing Construction Type Description   43366
Proposed Construction Type               43162
Proposed Construction Type Description   43162
Site Permit                             193541
Supervisor District                       1717
Neighborhoods - Analysis Boundaries       1725
Zipcode                                   1716
Location                                  1700
Record ID                                    0
dtype: int64
```

Обработка пропусков в данных

Удаление признаков, не содержащих данных

Удаляем из набора данных колонки, в которых **все** значения неопределены.

In [ ]:
```
data_dropped_na_columns = data.dropna(axis=1, how='all')
(data.shape[1], data_dropped_na_columns.shape[1])
```

Out[ ]:
```
(43, 43)
```

Удаление записей, не содержащих данных

Удаляем из набора данных ряды, в которых **все** значения неопределены.

In [ ]:
```
data_dropped_na_columns_and_rows = data_dropped_na_columns.dropna(axis=0, how='all')
(data.shape, data_dropped_na_columns_and_rows.shape)
```

Out[ ]:
```
((198900, 43), (198900, 43))
```

Удаление признаков не подлежащих восстановлению

In [ ]:
```
# Determines whether columns has numeric values.
def is_numeric(col: pd.Series):
    dt = str(col.dtype)
    return dt=='float64' or dt=='int64'

# Searching for columns with na.
rows_total_count = data_dropped_na_columns_and_rows.shape[0]

indexes_of_num_cols_with_na = []
cols_stats = []

for col_i in data_dropped_na_columns_and_rows.columns:
    col = data_dropped_na_columns_and_rows[col_i]
    null_count_in_col = data_dropped_na_columns_and_rows[col.isnull()].shape[0]

    if null_count_in_col>0 and is_numeric(col):
        perc = round((null_count_in_col / rows_total_count) * 100, 2)
        col_stats = {
            'i': col_i,
            'dtype': col.dtype,
            'null_count': null_count_in_col,
            'perc': perc
        }
        cols_stats.append(col_stats)
        indexes_of_num_cols_with_na.append(col_i)



for col_stats in cols_stats:
    print(f'Колонка {col_stats["i"]}:\
        \n\tТип данных {col_stats["dtype"]}. Количество пустых значений {col_stats["nu
```

```
Колонка Unit:
        Тип данных float64. Количество пустых значений 169421, 85.18%.
Колонка Number of Existing Stories:
        Тип данных float64. Количество пустых значений 42784, 21.51%.
Колонка Number of Proposed Stories:
        Тип данных float64. Количество пустых значений 42868, 21.55%.
Колонка Estimated Cost:
        Тип данных float64. Количество пустых значений 38066, 19.14%.
Колонка Revised Cost:
        Тип данных float64. Количество пустых значений 6066, 3.05%.
Колонка Existing Units:
        Тип данных float64. Количество пустых значений 51538, 25.91%.
Колонка Proposed Units:
        Тип данных float64. Количество пустых значений 50911, 25.6%.
Колонка Plansets:
        Тип данных float64. Количество пустых значений 37309, 18.76%.
Колонка Existing Construction Type:
        Тип данных float64. Количество пустых значений 43366, 21.8%.
Колонка Proposed Construction Type:
        Тип данных float64. Количество пустых значений 43162, 21.7%.
Колонка Supervisor District:
        Тип данных float64. Количество пустых значений 1717, 0.86%.
```

In [ ]:
```
# Get table consisting of numerical columns that have na.
data_num_na_init = data_dropped_na_columns_and_rows[indexes_of_num_cols_with_na]
data_num_na_init
```

Out[ ]:

	Unit	Number of Existing Stories	Number of Proposed Stories	Estimated Cost	Revised Cost	Existing Units	Proposed Units	Plansets	Existing Construction Type
0	NaN	6.0	NaN	4000.0	4000.0	143.0	NaN	2.0	3.0
1	0.0	7.0	NaN	1.0	500.0	NaN	NaN	2.0	3.0
2	NaN	6.0	6.0	20000.0	NaN	39.0	39.0	2.0	1.0
3	0.0	2.0	2.0	2000.0	2000.0	1.0	1.0	2.0	5.0
4	NaN	3.0	NaN	100000.0	100000.0	NaN	NaN	2.0	3.0
...	...	...	...	...	...	...	...	...	...
198895	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
198896	NaN	4.0	4.0	5000.0	5000.0	4.0	4.0	2.0	5.0
198897	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
198898	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
198899	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN

198900 rows × 11 columns

Некоторые колонки содержат слишком много пустых значений (больше 15%), если их попытаться заполнить, результат может сильно отличаться от реальности. Стоит подумать над тем, чтобы их убрать совсем. Для этого проанализируем их значимость для анализа:

- Unit - корпус здания. Мы анализируем право на строительство всего здания, от конкретного корпуса это не зависит. Можно от этого атрибута избавиться.
- Existing Units - аналогично Units.
- Proposed Units - аналогично Units.

Остальные колонки имеют пропуски из-за нехватки информации. Например, если данные собирались после непосредственного принятия решения, некоторые атрибуты, связанные с предварительной оценкой параметров здания не известны. Так, Estimated Cost имеет намного больше пропусков, чем Revised Cost. Для подобных колонок мы не в состоянии восстановить значения на основе других параметров записи.

In [ ]:
```
# Columns that have so many missing values that it's meaningless to restore them.
unrestorable_cols = [col_stats["i"] for col_stats in cols_stats if col_stats["perc"] >

# Get table without unrestorable columns.
data_dropped_na = data_dropped_na_columns_and_rows.drop(columns=unrestorable_cols)
# Get table consisting of numberical columns that have na and that are ok to be impute
data_num_na = data_num_na_init.drop(columns=unrestorable_cols)

data_num_na
```

Out[ ]:

	Revised Cost	Supervisor District
0	4000.0	3.0
1	500.0	3.0
2	NaN	3.0
3	2000.0	3.0
4	100000.0	6.0
...	...	...
198895	1.0	NaN
198896	5000.0	NaN
198897	1.0	NaN
198898	1.0	NaN
198899	1.0	NaN

198900 rows × 2 columns

Заполнение пропущенных значений

Так как в датасете некоторые пропущенные значения относятся к категориальным признакам, например, `Unit Suffix`, мы не можем произвести замену 0. Воспользуемся **импьютацией** (внедрением значений):

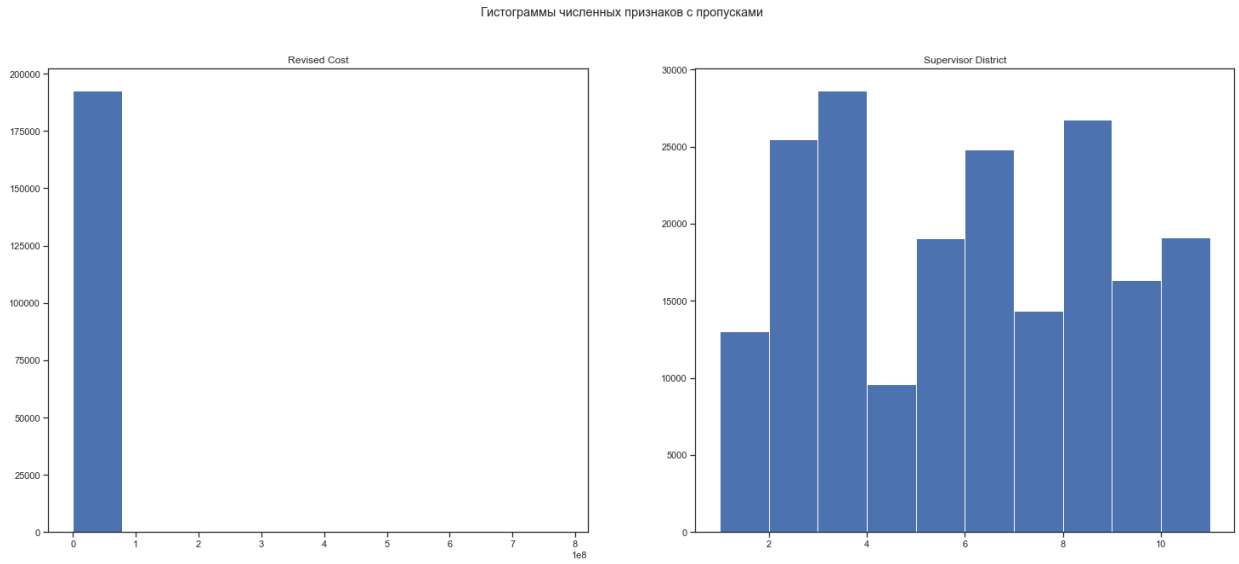
In [ ]:
```
from math import ceil, floor
```

```
NUMBER_OF_COLS = 2
cols = data_num_na.columns
number_of_rows = ceil(len(cols) / NUMBER_OF_COLS)
# Squeeze transforms (0, cols) plot from 2d to 1d. We don't need that (cur_ax = axs[x]
fig, axs = plt.subplots(nrows=number_of_rows, ncols=NUMBER_OF_COLS, figsize=(25, 10 *
fig.suptitle('Гистограммы численных признаков с пропусками')

# Maps 1d index into 2d index (for arranging plots).
def calculate_indexes(i: int, number_of_cols: int):
    return (floor(i / number_of_cols), i % number_of_cols)

for i, col_i in enumerate(cols):
    x, y = calculate_indexes(i, NUMBER_OF_COLS)
    cur_ax = axs[x][y]

    cur_ax.title.set_text(col_i)
    cur_ax.hist(data_num_na[col_i])
```



In [ ]:
```
from sklearn.impute import SimpleImputer, MissingIndicator
# Indicates missing values by mapping them to True.
indicator = MissingIndicator()

def get_mask_missing_values_only(col_i: str):
    data_num_na__col_i = data_num_na[[col_i]]
    return indicator.fit_transform(data_num_na__col_i)
```

In [ ]:
```
from enum import Enum, auto

class AutoName(Enum):
    # Auto generate value the same as the key.
    def _generate_next_value_(name, start, count, last_values):
        return name

class ImputeStrategy(str, AutoName):
    mean = auto()
    median = auto()
    most_frequent = auto()
```

In [ ]:
```
from pandas import DataFrame

def impute_num_col_in_dataset(dataset: DataFrame, column: str, strategy: ImputeStrateg
```

```
    data = dataset[[column]]

    imputer = SimpleImputer(strategy=strategy)
    data_num_imputed = imputer.fit_transform(data)

    # Check what was imputed.
    return data_num_imputed[get_mask_missing_values_only(column)]

# Closure for dataset.
def impute_num_col(column: str, strategy: ImputeStrategy):
    return impute_num_col_in_dataset(data_num_na, column, strategy)
```

Для `Revised Cost` следует применять стратегию моды, т.к. распределение одномодальное.

In [ ]:
```
impute_num_col('Revised Cost', ImputeStrategy.most_frequent)
```

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

Для `Supervisor District` мы наблюдаем многомодальное распределение. Воспользуемся стратегией медианы.

In [ ]:
```
impute_num_col('Supervisor District', ImputeStrategy.median)
```

Out[ ]:
```
array([6., 6., 6., ..., 6., 6., 6.])
```

Обработка пропусков в категориальных данных

In [ ]:
```
# Determines whether columns has categorial values.
def is_categorial(col: pd.Series):
    dt = str(col.dtype)
    return dt=='object' or dt=='categorial'

# Searching for columns with na.
rows_total_count = data_dropped_na.shape[0]

indexes_of_cat_cols_with_na = []
cols_stats = []

for col_i in data_dropped_na.columns:
    col = data_dropped_na[col_i]
    null_count_in_col = data_dropped_na[col.isnull()].shape[0]

    if null_count_in_col>0 and is_categorial(col):
        perc = round((null_count_in_col / rows_total_count) * 100, 2)
        col_stats = {
            'i': col_i,
            'dtype': col.dtype,
            'null_count': null_count_in_col,
            'perc': perc
        }
        cols_stats.append(col_stats)
        indexes_of_cat_cols_with_na.append(col_i)
```

```
for col_stats in cols_stats:
    print(f'Колонка {col_stats["i"]}:\
        \n\tТип данных {col_stats["dtype"]}. Количество пустых значений {col_stats["nu
```

```
Колонка Street Number Suffix:
        Тип данных object. Количество пустых значений 196684, 98.89%.
Колонка Street Suffix:
        Тип данных object. Количество пустых значений 2768, 1.39%.
Колонка Unit Suffix:
        Тип данных object. Количество пустых значений 196939, 99.01%.
Колонка Description:
        Тип данных object. Количество пустых значений 290, 0.15%.
Колонка Issued Date:
        Тип данных object. Количество пустых значений 14940, 7.51%.
Колонка Completed Date:
        Тип данных object. Количество пустых значений 101709, 51.14%.
Колонка First Construction Document Date:
        Тип данных object. Количество пустых значений 14946, 7.51%.
Колонка Structural Notification:
        Тип данных object. Количество пустых значений 191978, 96.52%.
Колонка Voluntary Soft-Story Retrofit:
        Тип данных object. Количество пустых значений 198865, 99.98%.
Колонка Fire Only Permit:
        Тип данных object. Количество пустых значений 180073, 90.53%.
Колонка Permit Expiration Date:
        Тип данных object. Количество пустых значений 51880, 26.08%.
Колонка Existing Use:
        Тип данных object. Количество пустых значений 41114, 20.67%.
Колонка Proposed Use:
        Тип данных object. Количество пустых значений 42439, 21.34%.
Колонка TIDF Compliance:
        Тип данных object. Количество пустых значений 198898, 100.0%.
Колонка Existing Construction Type Description:
        Тип данных object. Количество пустых значений 43366, 21.8%.
Колонка Proposed Construction Type Description:
        Тип данных object. Количество пустых значений 43162, 21.7%.
Колонка Site Permit:
        Тип данных object. Количество пустых значений 193541, 97.31%.
Колонка Neighborhoods - Analysis Boundaries:
        Тип данных object. Количество пустых значений 1725, 0.87%.
Колонка Location:
        Тип данных object. Количество пустых значений 1700, 0.85%.
```

In [ ]:
```
# Финальная версия датасета перед кодированием.
data_cleaned = data_dropped_na
```

Кодирование категориальных признаков

Импорт необходимых инструментов

In [ ]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

Закодируем признак `Street Suffix` целочисленными значениями (label encoding)

In [ ]:
```
# Уникальные значения столбца Street Suffix.
data_cleaned['Street Suffix'].unique()
```

Out[ ]:
```
array(['St', 'Av', 'Tr', 'Ct', 'Bl', 'Wy', 'Dr', nan, 'Rd', 'Cr', 'Pl',
       'Ln', 'Hy', 'Pk', 'Al', 'Pz', 'Wk', 'Rw', 'So', 'Sw', 'No', 'Hl'],
      dtype=object)
```

In [ ]:
```
label_encoder = LabelEncoder()
label_encoded__StreetSuffix = label_encoder.fit_transform(data_cleaned['Street Suffix'

# Уникальные значения в закодированном виде.
np.unique(label_encoded__StreetSuffix)
```

Out[ ]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21])
```

In [ ]:
```
# Обратная трансформация
label_encoder.inverse_transform(np.unique(label_encoded__StreetSuffix))
```

Out[ ]:
```
array(['Al', 'Av', 'Bl', 'Cr', 'Ct', 'Dr', 'Hl', 'Hy', 'Ln', 'No', 'Pk',
       'Pl', 'Pz', 'Rd', 'Rw', 'So', 'St', 'Sw', 'Tr', 'Wk', 'Wy', nan],
      dtype=object)
```

In [ ]:
```
# Заносим закодированный столбец в dataset.
data_label_encoded = data_cleaned.copy()
data_label_encoded['StreetSuffix'] = label_encoded__StreetSuffix

data_label_encoded.head()
```

Out[ ]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	St	...
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	Av	...
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	St	...

5 rows × 35 columns

Закодируем признак Site Permit наборами бинарных значений (с помощью one-hot encoding)

In [ ]:
```
data_label_encoded['Site Permit'].unique()
```

Out[ ]:
```
array([nan, 'Y'], dtype=object)
```

In [ ]:
```
oh_encoder = OneHotEncoder(dtype=np.int64)
```

```
oh_encoded__SitePermit = oh_encoder.fit_transform(data_label_encoded[['Site Permit']])
```

In [ ]:
```
# Обратить внимание на тип закодированных данных (разреженная матрица)
type(oh_encoded__SitePermit)
```

Out[ ]:
```
scipy.sparse._csr.csr_matrix
```

In [ ]:
```
oh_encoded__SitePermit.shape
```

Out[ ]:
```
(198900, 2)
```

In [ ]:
```
# Новые названия признаков.
oh_encoded_columns = oh_encoder.get_feature_names_out(['Site Permit'])
oh_encoded_columns
```

Out[ ]:
```
array(['Site Permit_Y', 'Site Permit_nan'], dtype=object)
```

In [ ]:
```
# Преобразуем разреженную матрицу в pandas DataFrame.
data_oh_encoded: pd.DataFrame = pd.DataFrame.sparse.from_spmatrix(oh_encoded__SitePerm
data_oh_encoded
```

Out[ ]:

	Site Permit_Y	Site Permit_nan
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
...	...	...
198895	0	1
198896	0	1
198897	0	1
198898	0	1
198899	0	1

198900 rows × 2 columns

Повторим кодирование с помощью get_dummies() из библиотеки Pandas

In [ ]:
```
data_encoded_dummies = pd.get_dummies(data_label_encoded, columns=['Site Permit'])
data_encoded_dummies
```

Out[ ]:

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market
...	...	...	...	...	...	...	...	...	...
198895	M862628	8	otc alterations permit	12/05/2017	0113	017A	1228	NaN	Montgomery
198896	201712055595	8	otc alterations permit	12/05/2017	0271	014	580	NaN	Bush
198897	M863507	8	otc alterations permit	12/06/2017	4318	019	1568	NaN	Indiana
198898	M863747	8	otc alterations permit	12/06/2017	0298	029	795	NaN	Sutter
198899	M864287	8	otc alterations permit	12/07/2017	0160	006	838	NaN	Pacific

198900 rows × 35 columns

Масштабирование данных

Импортируем необходимые инструменты

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

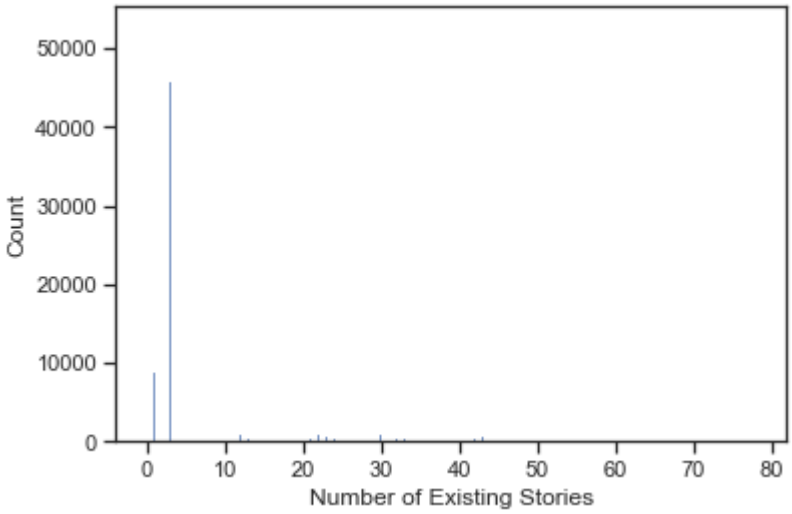
In [ ]:
```
# fig, ax = plt.subplots()

# ax.plot(x, y, linewidth=2.0)

# ax = data['Revised Cost']
```

```
# plt.show()
# sns.lineplot(data['Revised Cost'])
sns.histplot(data['Number of Existing Stories'])
```

Out[ ]:
```
<AxesSubplot:xlabel='Number of Existing Stories', ylabel='Count'>
```

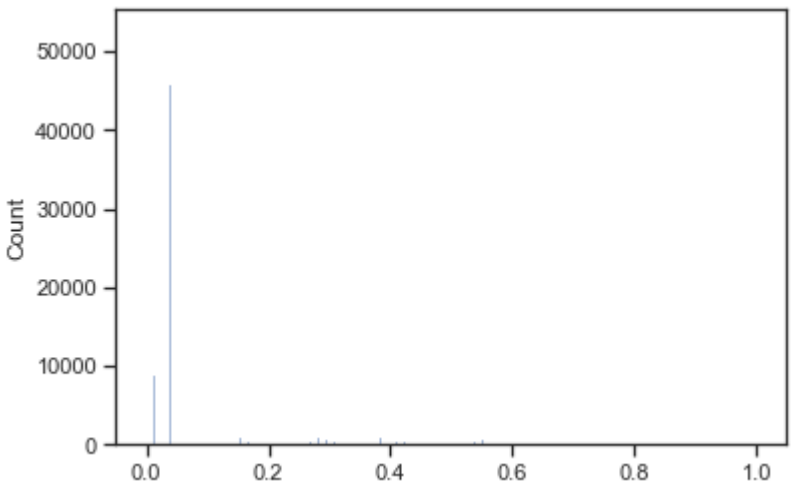


MinMax масштабирование

In [ ]:
```
minmax_scaler = MinMaxScaler()
minmax_scaled_data = minmax_scaler.fit_transform(data[['Number of Existing Stories']])
```

In [ ]:
```
sns.histplot(minmax_scaled_data, legend=False)
```

Out[ ]:
```
<AxesSubplot:ylabel='Count'>
```



Масштабирование на основе Z оценки

In [ ]:
```
standard_scaler = StandardScaler()
standard_scaled_data = standard_scaler.fit_transform(data[['Number of Existing Stories
```

In [ ]:
```
sns.histplot(standard_scaled_data, legend=False)
```

Out[ ]: <AxesSubplot:ylabel='Count'>