

МГТУ им. Н. Э. Баумана, кафедра ИУ5
курс “Технология машинного обучения”

Лабораторная работа №3

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

ВЫПОЛНИЛ:

Ерохин И.А.

Группа: ИУ5-61Б

ПРОВЕРИЛ:

Гапанюк Ю.Е.

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.

Выполненная работа:

Загрузка и первичный анализ данных

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```
In [3]: # Будем использовать только обучающую выборку
data = pd.read_csv('data/train.csv', sep=",")
```

```
In [4]: # размер набора данных
data.shape
```

```
Out[4]: (1460, 81)
```

```
In [5]: # типы колонок
data.dtypes
```

```
Out[5]: Id                int64
MSSubClass               int64
MSZoning                 object
LotFrontage             float64
LotArea                 int64
...
MoSold                  int64
YrSold                  int64
SaleType                object
SaleCondition           object
SalePrice               int64
Length: 81, dtype: object
```

```
In [6]: # проверим есть ли пропущенные значения
data.isnull().sum()
```

```
Out[6]: Id                0
MSSubClass               0
MSZoning                 0
LotFrontage             259
LotArea                 0
...
MoSold                  0
YrSold                  0
SaleType                0
SaleCondition           0
SalePrice               0
Length: 81, dtype: int64
```

```
In [7]: # Первые 5 строк датасета
data.head()
```

```
Out[7]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows x 81 columns

1. Обработка пропусков в данных

1.1. Простые стратегии - удаление или заполнение нулями

```
In [8]: total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))

Всего строк: 1460
```

```
In [10]: # Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

```
Out[10]: ((1460, 81), (1460, 62))
```

```
In [11]: # Удаление строк, содержащих пустые значения
data_new_2 = data.dropna(axis=0, how='any')
(data.shape, data_new_2.shape)
```

```
Out[11]: ((1460, 81), (0, 81))
```

```
In [12]: data.head()
```

```
Out[12]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows x 81 columns

```
In [13]: # Заполнение всех пропущенных значений нулями
# В данном случае это некорректно, так как нулями заполняются в том числе категориальные колонки
data_new_3 = data.fillna(0)
data_new_3.head()
```

```
Out[13]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
0	1	60	RL	65.0	8450	Pave	0	Reg	Lvl	AllPub	...	0	0	0	0	0	
1	2	20	RL	80.0	9600	Pave	0	Reg	Lvl	AllPub	...	0	0	0	0	0	
2	3	60	RL	68.0	11250	Pave	0	IR1	Lvl	AllPub	...	0	0	0	0	0	
3	4	70	RL	60.0	9550	Pave	0	IR1	Lvl	AllPub	...	0	0	0	0	0	
4	5	60	RL	84.0	14260	Pave	0	IR1	Lvl	AllPub	...	0	0	0	0	0	

5 rows x 81 columns

```
In [14]: # Выберем числовые колонки с пропущенными значениями
# Цикл по колонкам датасета
num_cols = []
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 == '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))

Колонка LotFrontage. Тип данных float64. Количество пустых значений 259, 17.74%.
Колонка MasVnrArea. Тип данных float64. Количество пустых значений 8, 0.55%.
Колонка GarageYrBlt. Тип данных float64. Количество пустых значений 81, 5.55%.
```

```
In [15]: # Фильтр по колонкам с пропущенными значениями
data_num = data[num_cols]
data_num
```

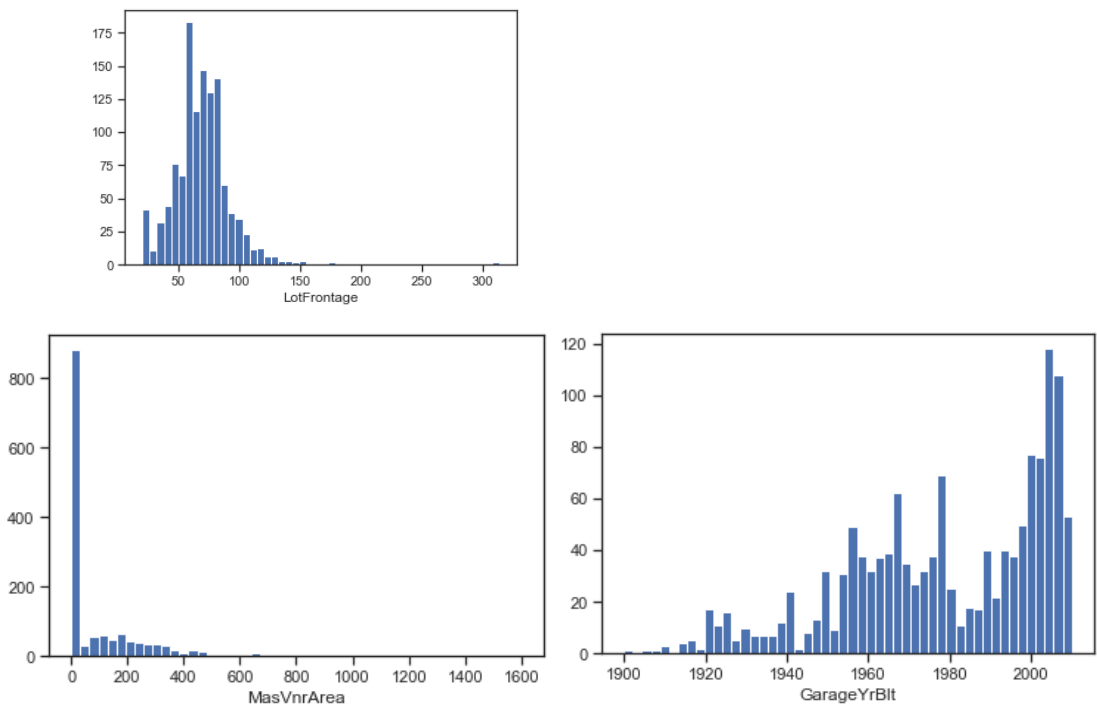
Out[15]:

	LotFrontage	MasVnrArea	GarageYrBlt
0	65.0	196.0	2003.0
1	80.0	0.0	1976.0
2	68.0	162.0	2001.0
3	60.0	0.0	1998.0
4	84.0	350.0	2000.0
...
1455	62.0	0.0	1999.0
1456	85.0	119.0	1978.0
1457	66.0	0.0	1941.0
1458	68.0	0.0	1950.0
1459	75.0	0.0	1965.0

1460 rows x 3 columns

```
In [16]: # Гистограмма по признакам
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()

c:\django_projects\untitled\venv\lib\site-packages\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encountered in greater_equal
    keep = (tmp_a >= first_edge)
c:\django_projects\untitled\venv\lib\site-packages\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encountered in less_equal
    keep &= (tmp_a <= last_edge)
```



```
In [17]: # Фильтр по пустым значениям поля LotFrontage
data[data['MasVnrArea'].isnull()]
```

Out[17]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVn
234	235	60	RL	NaN	7851	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
529	530	20	RL	NaN	32668	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
650	651	60	FV	65.0	8125	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
936	937	20	RL	67.0	10083	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
973	974	20	FV	95.0	11639	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
977	978	120	FV	35.0	4274	Pave	Pave	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
1243	1244	20	RL	107.0	13891	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
1278	1279	60	RL	75.0	9473	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	

8 rows x 81 columns

```
In [18]: # Запоминаем индексы строк с пустыми значениями
flt_index = data[data['MasVnrArea'].isnull()].index
flt_index
```

```
Out[18]: Int64Index([234, 529, 650, 936, 973, 977, 1243, 1278], dtype='int64')
```

```
In [19]: # Проверяем что выводятся нужные строки
data[data.index.isin(flt_index)]
```

```
Out[19]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscV
234	235	60	RL	NaN	7851	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
529	530	20	RL	NaN	32668	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
650	651	60	FV	65.0	8125	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
936	937	20	RL	67.0	10083	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
973	974	20	FV	95.0	11639	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
977	978	120	FV	35.0	4274	Pave	Pave	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
1243	1244	20	RL	107.0	13891	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
1278	1279	60	RL	75.0	9473	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	

8 rows x 81 columns

```
In [20]: # Фильтр по колонке
data_num[data_num.index.isin(flt_index)][['MasVnrArea']]
```

```
Out[20]:
```

234	NaN
529	NaN
650	NaN
936	NaN
973	NaN
977	NaN
1243	NaN
1278	NaN

Name: MasVnrArea, dtype: float64

```
In [27]: data_num_MasVnrArea = data_num[['MasVnrArea']]
data_num_MasVnrArea.head()
```

```
Out[27]:
```

	MasVnrArea
0	196.0
1	0.0
2	162.0
3	0.0
4	350.0

```
In [23]: from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

```
In [29]: # Фильтр для проверки заполнения пустых значений
indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(data_num_MasVnrArea)
mask_missing_values_only
```

```
Out[29]: array([[False],
[False],
[False],
...,
[False],
[False],
[False]])
```

```
In [31]: strategies=['mean', 'median', 'most_frequent']
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(data_num_MasVnrArea)
    return data_num_imp[mask_missing_values_only]
strategies[0], test_num_impute(strategies[0])
```

```
Out[31]: ('mean',
array([103.68526171, 103.68526171, 103.68526171, 103.68526171,
103.68526171, 103.68526171, 103.68526171, 103.68526171]))
```

```
In [32]: strategies[1], test_num_impute(strategies[1])
```

```
Out[32]: ('median', array([0., 0., 0., 0., 0., 0., 0., 0.]))
```

```
In [33]: strategies[2], test_num_impute(strategies[2])
```

```
Out[33]: ('most_frequent', array([0., 0., 0., 0., 0., 0., 0., 0.]))
```

```
In [35]: # Более сложная функция, которая позволяет задавать колонку и вид импутации
def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]

    return column, strategy_param, filled_data.size, filled_data[0], filled_data[filled_data.size-1]

data[['GarageYrBlt']].describe()
```

```
Out[35]:
```

	GarageYrBlt
count	1379.000000
mean	1978.506164
std	24.689725
min	1900.000000
25%	1961.000000
50%	1980.000000
75%	2002.000000
max	2010.000000

```
In [36]: test_num_impute_col(data, 'GarageYrBlt', strategies[0])
```

```
Out[36]: ('GarageYrBlt', 'mean', 81, 1978.5061638868744, 1978.5061638868744)
```

```
In [37]: test_num_impute_col(data, 'GarageYrBlt', strategies[1])
```

```
Out[37]: ('GarageYrBlt', 'median', 81, 1980.0, 1980.0)
```

```
In [38]: test_num_impute_col(data, 'GarageYrBlt', strategies[2])
```

```
Out[38]: ('GarageYrBlt', 'most_frequent', 81, 2005.0, 2005.0)
```

```
In [39]: # Выберем категориальные колонки с пропущенными значениями
# Цикл по колонкам датасета
cat_cols = []
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'):
        cat_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 Alley. Тип данных object. Количество пустых значений 1369, 93.77%.
 Колонка MasVnrType. Тип данных object. Количество пустых значений 8, 0.55%.
 Колонка BsmtQual. Тип данных object. Количество пустых значений 37, 2.53%.
 Колонка BsmtCond. Тип данных object. Количество пустых значений 37, 2.53%.
 Колонка BsmtExposure. Тип данных object. Количество пустых значений 38, 2.6%.
 Колонка BsmtFinType1. Тип данных object. Количество пустых значений 37, 2.53%.
 Колонка BsmtFinType2. Тип данных object. Количество пустых значений 38, 2.6%.
 Колонка Electrical. Тип данных object. Количество пустых значений 1, 0.07%.
 Колонка FireplaceQu. Тип данных object. Количество пустых значений 690, 47.26%.
 Колонка GarageType. Тип данных object. Количество пустых значений 81, 5.55%.
 Колонка GarageFinish. Тип данных object. Количество пустых значений 81, 5.55%.
 Колонка GarageQual. Тип данных object. Количество пустых значений 81, 5.55%.
 Колонка GarageCond. Тип данных object. Количество пустых значений 81, 5.55%.
 Колонка PoolQC. Тип данных object. Количество пустых значений 1453, 99.52%.
 Колонка Fence. Тип данных object. Количество пустых значений 1179, 80.75%.
 Колонка MiscFeature. Тип данных object. Количество пустых значений 1406, 96.3%.

```
In [40]: cat_temp_data = data[['MasVnrType']]
cat_temp_data.head()
```

```
Out[40]:
```

	MasVnrType
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace

```
In [41]: cat_temp_data['MasVnrType'].unique()
```

```
Out[41]: array(['BrkFace', 'None', 'Stone', 'BrkCmn', nan], dtype=object)
```

```
In [42]: cat_temp_data[cat_temp_data['MasVnrType'].isnull()].shape
```

```
Out[42]: (8, 1)
```

```
In [43]: # Импутация наиболее частыми значениями
imp2 = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
data_imp2 = imp2.fit_transform(cat_temp_data)
data_imp2
```

```
Out[43]: array([[ 'BrkFace'],
               [None],
               [ 'BrkFace'],
               ...,
               [None],
               [None],
               [None]], dtype=object)
```

```
In [44]: # Пустые значения отсутствуют
np.unique(data_imp2)
```

```
Out[44]: array([ 'BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

```
In [45]: # Импутация константой
imp3 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='!!!')
data_imp3 = imp3.fit_transform(cat_temp_data)
data_imp3
```

```
Out[45]: array([[ 'BrkFace'],
               [None],
               [ 'BrkFace'],
               ...,
               [None],
               [None],
               [None]], dtype=object)
```

```
In [46]: np.unique(data_imp3)
```

```
Out[46]: array(['!!!', 'BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

```
In [47]: data_imp3[data_imp3=='!!!'].size
```

```
Out[47]: 8
```

```
In [49]: cat_enc = pd.DataFrame({'c1':data_imp2.T[0]})
cat_enc
```

```
Out[49]:
```

	c1
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace
...	...
1455	None
1456	Stone
1457	None
1458	None
1459	None

1460 rows x 1 columns

```
In [51]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
cat_enc_le = le.fit_transform(cat_enc['c1'])
cat_enc['c1'].unique()
```

```
Out[51]: array([ 'BrkFace', 'None', 'Stone', 'BrkCmn'], dtype=object)
```

```
In [52]: np.unique(cat_enc_le)
```

```
Out[52]: array([0, 1, 2, 3])
```

```
In [53]: le.inverse_transform([0, 1, 2, 3])
```

```
Out[53]: array([ 'BrkCmn', 'BrkFace', 'None', 'Stone'], dtype=object)
```

```
In [56]: ohe = OneHotEncoder()  
cat_enc_ohe = ohe.fit_transform(cat_enc[['c1']])  
cat_enc_ohe.shape
```

```
Out[56]: (1460, 1)
```

```
In [57]: cat_enc_ohe.shape
```

```
Out[57]: (1460, 4)
```

```
In [58]: cat_enc_ohe
```

```
Out[58]: <1460x4 sparse matrix of type '<class 'numpy.float64'>'  
with 1460 stored elements in Compressed Sparse Row format>
```

```
In [59]: cat_enc_ohe.todense()[0:10]
```

```
Out[59]: matrix([[0., 1., 0., 0.],  
                [0., 0., 1., 0.],  
                [0., 1., 0., 0.],  
                [0., 0., 1., 0.],  
                [0., 1., 0., 0.],  
                [0., 0., 1., 0.],  
                [0., 0., 0., 1.],  
                [0., 0., 0., 1.],  
                [0., 0., 1., 0.],  
                [0., 0., 1., 0.]])
```

```
In [60]: cat_enc.head(10)
```

```
Out[60]:
```

	c1
0	BrkFace
1	None
2	BrkFace
3	None
4	BrkFace
5	None
6	Stone
7	Stone
8	None
9	None

```
In [61]: pd.get_dummies(cat_enc).head()
```

```
Out[61]:
```

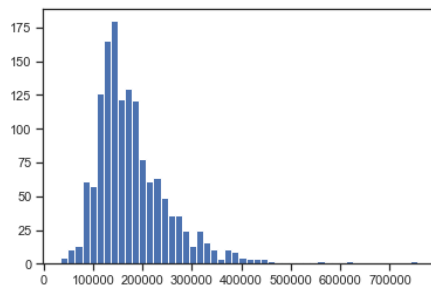
	c1_BrkCmn	c1_BrkFace	c1_None	c1_Stone
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	1	0	0

```
In [62]: pd.get_dummies(cat_temp_data, dummy_na=True).head()
```

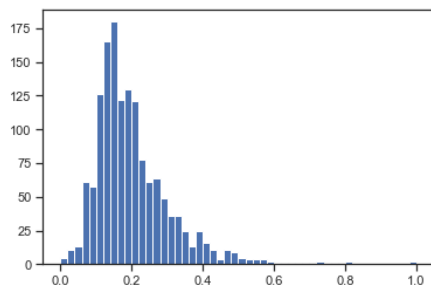
```
Out[62]:
```

	MasVnrType_BrkCmn	MasVnrType_BrkFace	MasVnrType_None	MasVnrType_Stone	MasVnrType_nan
0	0	1	0	0	0
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	1	0	0	0

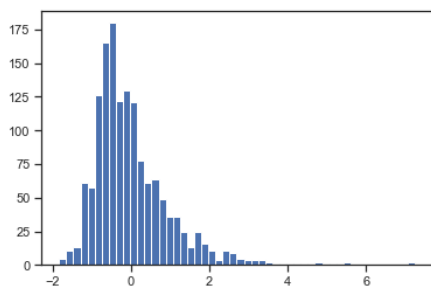

```
In [63]: from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['SalePrice']])
plt.hist(data['SalePrice'], 50)
plt.show()
```



```
In [64]: plt.hist(sc1_data, 50)
plt.show()
```



```
In [65]: sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['SalePrice']])
plt.hist(sc2_data, 50)
plt.show()
```



```
In [66]: sc3 = Normalizer()
sc3_data = sc3.fit_transform(data[['SalePrice']])
plt.hist(sc3_data, 50)
plt.show()
```

