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.

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

Рассмотрим датасет Global Power-Plants:\ Dataset shows list of 35K powerplants with their generation capacity\ и исследуем его.

https://www.kaggle.com/datasets/ramjasmaurya/global-powerplants

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('powerplants (global) - global_power_plants.csv', sep=",")
```

In [4]:
```
data.shape
```

Out[4]:
```
(34936, 16)
```

In [5]:
```
data.dtypes
```

Out[5]:
```
country code                 object
country                      object
name of powerplant           object
capacity in MW              float64
latitude                    float64
longitude                   float64
primary_fuel                 object
secondary fuel               object
other_fuel 1                 object
other_fuel 2                 object
start date                  float64
owner of plant               object
geolocation_source           object
generation_gwh_2020         float64
```

```
generation_data_source            object
estimated_generation_gwh_2020    float64
dtype: object
```

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

Out[6]:
```
country code                         0
country                              0
name of powerplant                   0
capacity in MW                       0
latitude                             0
longitude                            0
primary_fuel                         0
secondary fuel                   32992
other_fuel 1                     34660
other_fuel 2                     34844
start date                       17489
owner of plant                   14068
geolocation_source                 419
generation_gwh_2020              25277
generation_data_source           23536
estimated_generation_gwh_2020     1798
dtype: int64
```

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

Out[7]:

	country code	country	name of powerplant	capacity in MW	latitude	longitude	primary_fuel	secondary fuel	other_fuel 1	other_fuel 2
0	AFG	Afghanistan	Kajaki Hydroelectric Power Plant Afghanistan	33.0	32.322	65.1190	Hydro	NaN	NaN	NaN
1	AFG	Afghanistan	Kandahar DOG	10.0	31.670	65.7950	Solar	NaN	NaN	NaN
2	AFG	Afghanistan	Kandahar JOL	10.0	31.623	65.7920	Solar	NaN	NaN	NaN
3	AFG	Afghanistan	Mahipar Hydroelectric Power Plant Afghanistan	66.0	34.556	69.4787	Hydro	NaN	NaN	NaN
4	AFG	Afghanistan	Naghlu Dam Hydroelectric Power Plant Afghanistan	100.0	34.641	69.7170	Hydro	NaN	NaN	NaN

In [8]:
```
# Удаление колонок, содержащих пустые значения
data_new_1 = data.dropna(axis=1, how='any')
(data.shape, data_new_1.shape)
```

Out[8]:
```
((34936, 16), (34936, 7))
```

In [9]:
```
# 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 / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col,
```

```
Колонка start date. Тип данных float64. Количество пустых значений 17489, 50.06%.
Колонка generation_gwh_2020. Тип данных float64. Количество пустых значений 25277, 72.35%.
Колонка estimated_generation_gwh_2020. Тип данных float64. Количество пустых значений 179
8, 5.15%.
```

In [10]:

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

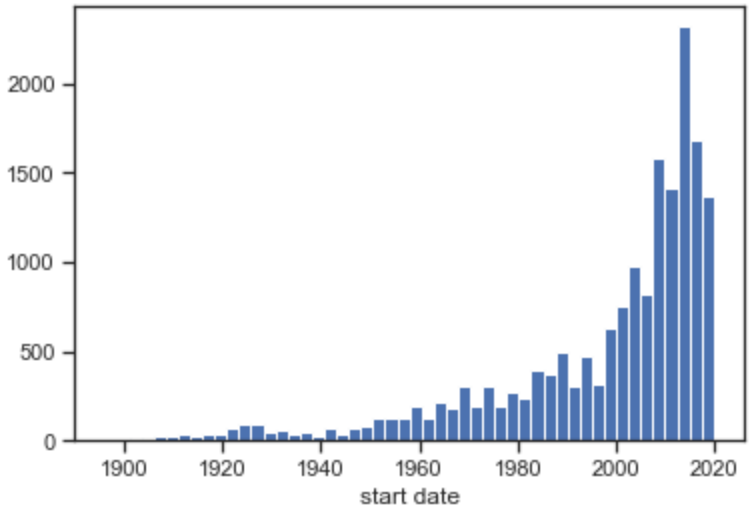
Out[10]:

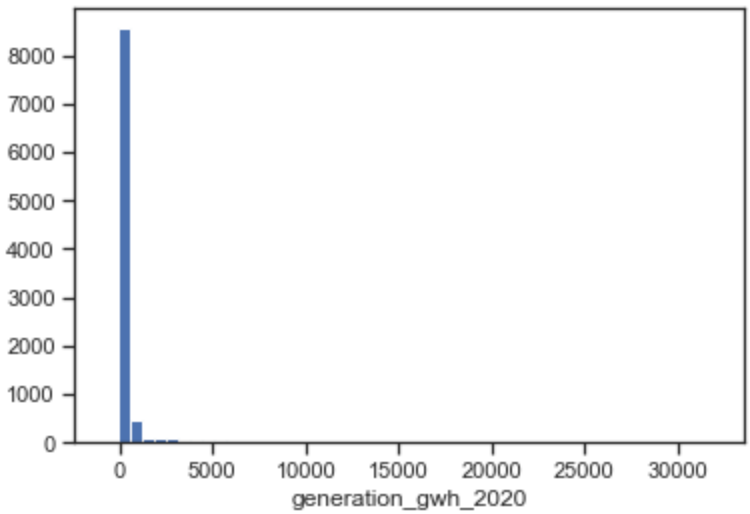
	start date	generation_gwh_2020	estimated_generation_gwh_2020
0	NaN	NaN	119.50
1	NaN	NaN	18.29
2	NaN	NaN	18.72
3	NaN	NaN	174.91
4	NaN	NaN	350.80
...	...	...	...
34931	NaN	NaN	183.79
34932	NaN	NaN	73.51
34933	NaN	NaN	578.32
34934	NaN	NaN	2785.10
34935	NaN	NaN	3960.24

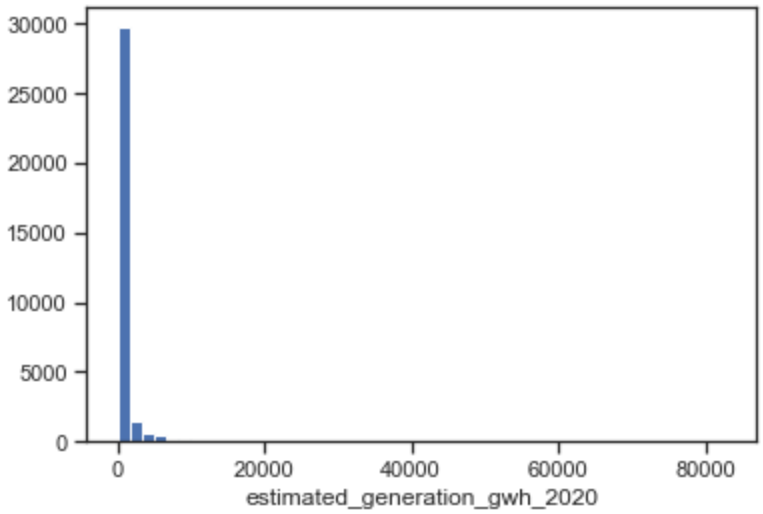
34936 rows × 3 columns

In [12]:

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







In [38]:

```
#введем необходимые функции, чтобы заполнить пропуски в числовых данных
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
strategies=['mean', 'median', 'most_frequent']
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]
```

```
    dataset[column] = data_num_imp
    return column, strategy_param, filled_data.size, filled_data[0], filled_data[filled_da
```

In [39]:
```
data['start date'] = data['start date'].fillna(0)
data.head()
```

Out[39]:

	country code	country	name of powerplant	capacity in MW	latitude	longitude	primary_fuel	secondary fuel	other_fuel 1	other_fuel 2
0	AFG	Afghanistan	Kajaki Hydroelectric Power Plant Afghanistan	33.0	32.322	65.1190	Hydro	NaN	NaN	NaN
1	AFG	Afghanistan	Kandahar DOG	10.0	31.670	65.7950	Solar	NaN	NaN	NaN
2	AFG	Afghanistan	Kandahar JOL	10.0	31.623	65.7920	Solar	NaN	NaN	NaN
3	AFG	Afghanistan	Mahipar Hydroelectric Power Plant Afghanistan	66.0	34.556	69.4787	Hydro	NaN	NaN	NaN
4	AFG	Afghanistan	Naghlu Dam Hydroelectric Power Plant Afghanistan	100.0	34.641	69.7170	Hydro	NaN	NaN	NaN

In [40]:
```
test_num_impute_col(data, 'generation_gwh_2020', strategies[1])
```

Out[40]:
```
('generation_gwh_2020', 'median', 25277, 11.53, 11.53)
```

In [41]:
```
test_num_impute_col(data, 'estimated_generation_gwh_2020', strategies[1])
```

Out[41]:
```
('estimated_generation_gwh_2020', 'median', 1798, 37.59, 37.59)
```

In [42]:
```
data.head()
```

Out[42]:

	country code	country	name of powerplant	capacity in MW	latitude	longitude	primary_fuel	secondary fuel	other_fuel 1	other_fuel 2
0	AFG	Afghanistan	Kajaki Hydroelectric Power Plant Afghanistan	33.0	32.322	65.1190	Hydro	NaN	NaN	NaN
1	AFG	Afghanistan	Kandahar DOG	10.0	31.670	65.7950	Solar	NaN	NaN	NaN
2	AFG	Afghanistan	Kandahar JOL	10.0	31.623	65.7920	Solar	NaN	NaN	NaN
3	AFG	Afghanistan	Mahipar Hydroelectric Power Plant Afghanistan	66.0	34.556	69.4787	Hydro	NaN	NaN	NaN

	country code	country	name of powerplant	capacity in MW	latitude	longitude	primary_fuel	secondary fuel	other_fuel 1	other_fuel 2
4	AFG	Afghanistan	Naghlu Dam Hydroelectric Power Plant Afghanistan	100.0	34.641	69.7170	Hydro	NaN	NaN	NaN

Заполнинили "start date" и "generation_gwh_2020", "estimated_generation_gwh_2020" значениями "0" и "median" соответсвенно

In [44]:
```
# 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 / data.shape[0]) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'.format(col,
```

```
Колонка secondary fuel. Тип данных object. Количество пустых значений 32992, 94.44%.
Колонка other_fuel 1. Тип данных object. Количество пустых значений 34660, 99.21%.
Колонка other_fuel 2. Тип данных object. Количество пустых значений 34844, 99.74%.
Колонка owner of plant. Тип данных object. Количество пустых значений 14068, 40.27%.
Колонка geolocation_source. Тип данных object. Количество пустых значений 419, 1.2%.
Колонка generation_data_source. Тип данных object. Количество пустых значений 23536, 67.3
7%.
```

Как видим удаление колонок в данном случае можно применить к колонкам "secondary_fuel", т.к. очень маленький процент из всей выборки составляют реальные данные. Но чтобы не лишаться информации о "secondary_fuel", заполним эти колонки значение "None"

In [46]:
```
data['secondary fuel'].unique()
```

Out[46]:
```
array([nan, 'Oil', 'Solar', 'Gas', 'Other', 'Hydro', 'Coal', 'Petcoke',
       'Biomass', 'Waste', 'Cogeneration', 'Storage', 'Wind'],
      dtype=object)
```

In [47]:
```
data['other_fuel 1'].unique()
```

Out[47]:
```
array([nan, 'Other', 'Oil', 'Biomass', 'Gas', 'Solar', 'Waste', 'Storage',
       'Hydro', 'Wind', 'Coal', 'Petcoke'], dtype=object)
```

In [48]:
```
data['other_fuel 2'].unique()
```

Out[48]:
```
array([nan, 'Other', 'Gas', 'Solar', 'Biomass', 'Hydro', 'Oil', 'Wind',
       'Storage'], dtype=object)
```

In [ ]:
```
data['secondary fuel'].unique()
```

In [58]:
```
imp = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='None')
data[['secondary fuel']] = imp.fit_transform(data[['secondary fuel']])
data[['other_fuel 1']] = imp.fit_transform(data[['secondary fuel']])
data[['other_fuel 2']] = imp.fit_transform(data[['secondary fuel']])
data[['owner of plant']] = imp.fit_transform(data[['secondary fuel']])
```

```
data[['geolocation_source']] = imp.fit_transform(data[['secondary fuel']])
data[['generation_data_source']] = imp.fit_transform(data[['secondary fuel']])

data.head()
```

Out[58]:

	country code	country	name of powerplant	capacity in MW	latitude	longitude	primary_fuel	secondary fuel	other_fuel 1	other_fuel 2
0	AFG	Afghanistan	Kajaki Hydroelectric Power Plant Afghanistan	33.0	32.322	65.1190	Hydro	None	None	None
1	AFG	Afghanistan	Kandahar DOG	10.0	31.670	65.7950	Solar	None	None	None
2	AFG	Afghanistan	Kandahar JOL	10.0	31.623	65.7920	Solar	None	None	None
3	AFG	Afghanistan	Mahipar Hydroelectric Power Plant Afghanistan	66.0	34.556	69.4787	Hydro	None	None	None
4	AFG	Afghanistan	Naghlu Dam Hydroelectric Power Plant Afghanistan	100.0	34.641	69.7170	Hydro	None	None	None

Закодируем признаки колонки "primary_fuel" целочисленнми значениями

In [59]:
```
from sklearn.preprocessing import LabelEncoder
data['primary_fuel'].unique()
```

Out[59]:
```
array(['Hydro', 'Solar', 'Gas', 'Other', 'Oil', 'Wind', 'Nuclear', 'Coal',
       'Waste', 'Biomass', 'Wave and Tidal', 'Petcoke', 'Geothermal',
       'Storage', 'Cogeneration'], dtype=object)
```

In [60]:
```
le = LabelEncoder()
data1 = le.fit_transform(data['primary_fuel'])
data1
```

Out[60]:
```
array([ 5, 10, 10, ...,  5,  1,  5])
```

In [63]:
```
np.unique(data1)
```

Out[63]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [66]:
```
data1 = le.inverse_transform([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
data1
```

Out[66]:
```
array(['Biomass', 'Coal', 'Cogeneration', 'Gas', 'Geothermal', 'Hydro',
       'Nuclear', 'Oil', 'Other', 'Petcoke', 'Solar', 'Storage', 'Waste',
       'Wave and Tidal', 'Wind'], dtype=object)
```

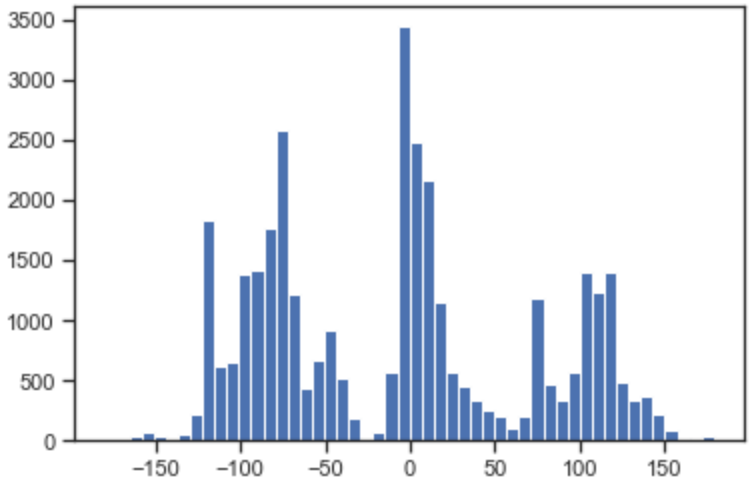
In [68]:
```
pd.get_dummies(data['primary_fuel']).head()
```

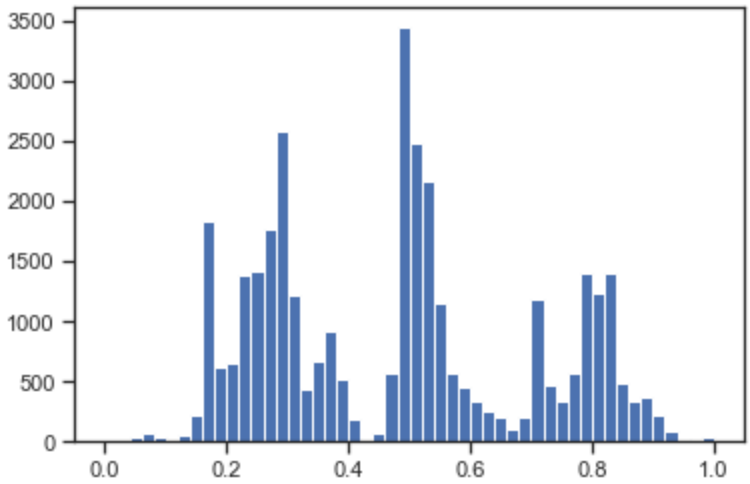
Out[68]:

	Biomass	Coal	Cogeneration	Gas	Geothermal	Hydro	Nuclear	Oil	Other	Petcoke	Solar	Storage	Waste	W T
0	0	0	0	0	0	1	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	0	0	0	0	0	0	0	0	0	0	1	0	0	
3	0	0	0	0	0	1	0	0	0	0	0	0	0	
4	0	0	0	0	0	1	0	0	0	0	0	0	0	

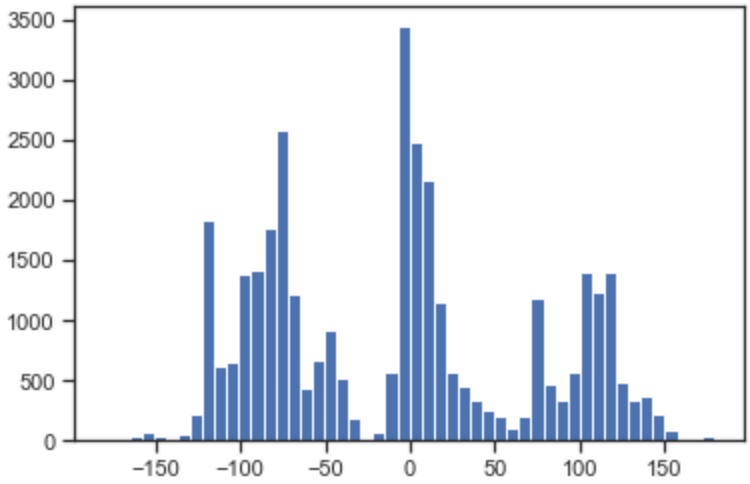
Далее применим масштабирование MINMAX и на основе Z-оценки для колонки "longitude"

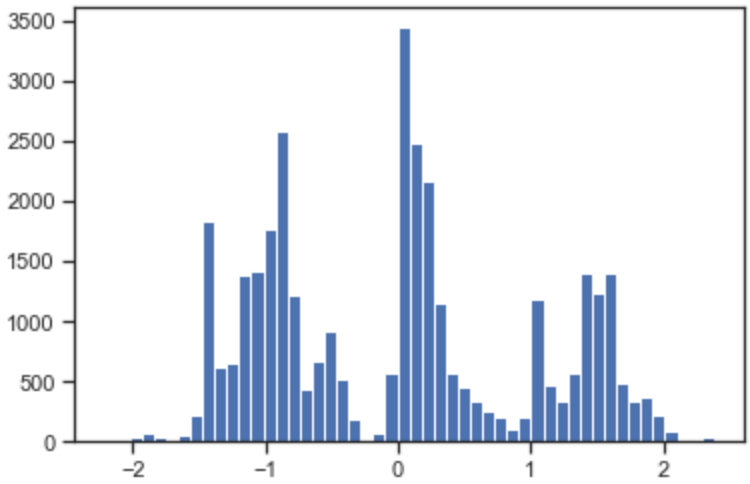
In [79]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[['longitude']])
plt.hist(data['longitude'], 50)
plt.show()
plt.hist(sc1_data, 50)
plt.show()
```





In [80]:
```
sc2 = StandardScaler()
sc2_data = sc2.fit_transform(data[['longitude']])
plt.hist(data['longitude'], 50)
plt.show()
plt.hist(sc2_data, 50)
plt.show()
```


3500
3000
2500
2000
1500
1000
500
0
−150
−100
−50
0
50
100
150


3500
3000
2500
2000
1500
1000
500
0
−2
−1
0
1
2