```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'house-prices-advanced-regression-techniques:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-competitions-data%2Fka

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading house-prices-advanced-regression-techniques, 203809 bytes compressed
[==================================================] 203809 bytes downloaded
Downloaded and uncompressed: house-prices-advanced-regression-techniques
Data source import complete.
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
import matplotlib.pyplot as plt


# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Sa
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
df = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techniques/train.csv",index_col ='Id')
df
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	PoolQC
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	NaN
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	NaN
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	NaN
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN
1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN
1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN
1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN
1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN

1460 rows × 80 columns

Data Preprocessing

Dealing with Nan Values

```
nan_cols = dict()
for j,i in zip(df.columns,np.array(df.isna().sum())):
    if i > 500:
        nan_cols[j] = i
nan_cols
```

```
{'Alley': 1369,
 'MasVnrType': 872,
 'FireplaceQu': 690,
 'PoolQC': 1453,
 'Fence': 1179,
 'MiscFeature': 1406}
```

```
df.drop(list(nan_cols.keys()), axis = 1,inplace = True)
df
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	...	EnclosedPorch
1	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0
2	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub	FR2	Gtl	...	0
3	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub	Inside	Gtl	...	0
4	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub	Corner	Gtl	...	272
5	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub	FR2	Gtl	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1456	60	RL	62.0	7917	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0
1457	20	RL	85.0	13175	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0
1458	70	RL	66.0	9042	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0
1459	20	RL	68.0	9717	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	112
1460	20	RL	75.0	9937	Pave	Reg	Lvl	AllPub	Inside	Gtl	...	0

1460 rows × 74 columns

```
df.dtypes['MSZoning']
```

```
dtype('O')
```

```
nan_rows = dict()
for j,i in zip(df.columns,np.array(df.isna().sum())):
    if i > 0:
        nan_rows[j] = i
nan_rows
```

```
{'LotFrontage': 259,
 'MasVnrArea': 8,
 'BsmtQual': 37,
 'BsmtCond': 37,
 'BsmtExposure': 38,
 'BsmtFinType1': 37,
 'BsmtFinType2': 38,
 'Electrical': 1,
 'GarageType': 81,
 'GarageYrBlt': 81,
 'GarageFinish': 81,
 'GarageQual': 81,
 'GarageCond': 81}
```

```
nan_obj_col = df[list(nan_rows.keys())].select_dtypes(include = 'object').columns
nan_obj_col
```

```
Index(['BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Electrical', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'],
      dtype='object')
```

+ Code + Text

```
for col in nan_obj_col:
    df[col] = df[col].fillna(df[col].value_counts().keys()[0])
nan_rows = dict()
for j,i in zip(df.columns,np.array(df.isna().sum())):
    if i > 0:
        nan_rows[j] = i
nan_rows
```

```
{'LotFrontage': 259, 'MasVnrArea': 8, 'GarageYrBlt': 81}
```

```
for col in nan_rows.keys():
    df[col] = df[col].fillna(df[col].mean())
df.isna().sum().sum()
```

```
0
```

Dealing with duplicates

```
df.duplicated().sum()
```

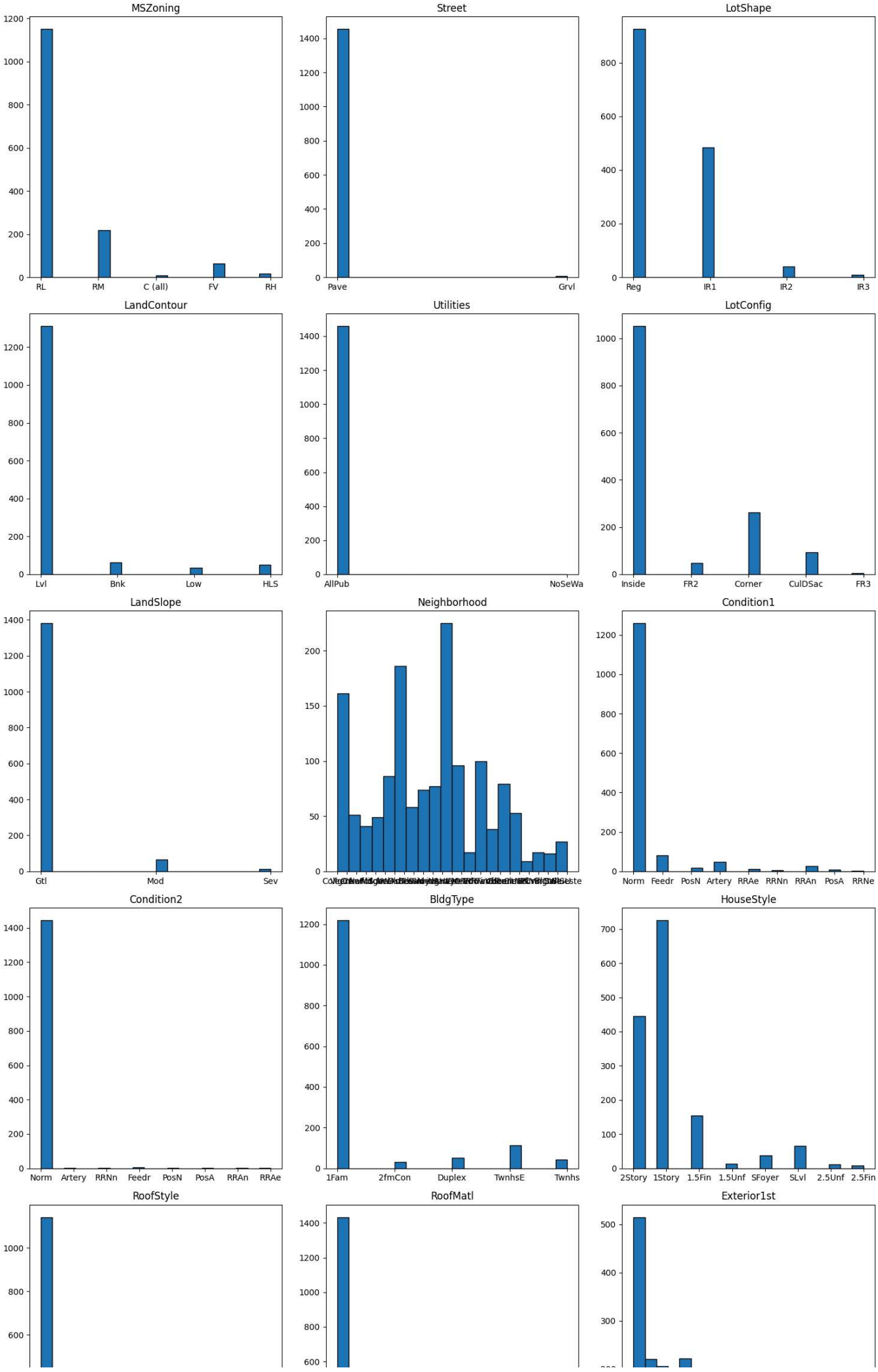
```
0
```

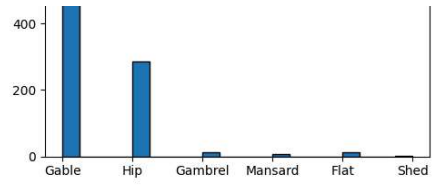
Applying Label Encoder

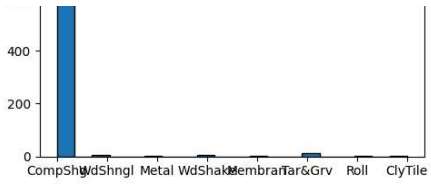
```
obj_cols = df.select_dtypes(include = 'object').columns
obj_cols
```

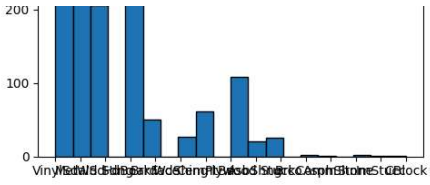
```
Index(['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities',
       'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional',
       'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive',
       'SaleType', 'SaleCondition'],
      dtype='object')
```

```
plt.figure(figsize=(15, 5 * 13))
for i,col in enumerate(obj_cols):
    plt.subplot(13,3,i+1)
    plt.hist(df[col],bins = 20, edgecolor = 'black')
    plt.title(col)
plt.tight_layout()
plt.show()
```

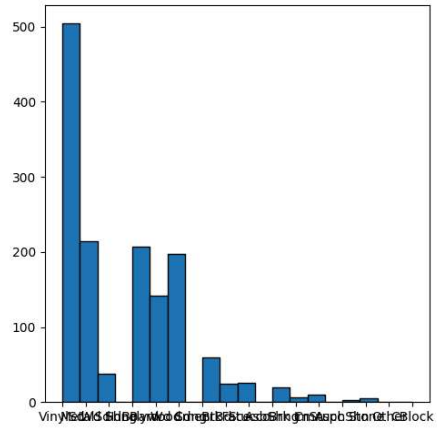

MSZoning
RL
RM
C (all)
FV
RH
Street
Pave
Grvl
LotShape
Reg
IR1
IR2
IR3
LandContour
Lvl
Bnk
Low
HLS
Utilities
AllPub
NoSeWa
LotConfig
Inside
FR2
Corner
CulDSac
FR3
LandSlope
Gtl
Mod
Sev
Neighborhood
Condition1
Norm
Feedr
PosN
Artery
RRAe
RRNn
RRAn
PosA
RRNe
Condition2
Norm
Artery
RRNn
Feedr
PosN
PosA
RRAn
RRAe
BldgType
1Fam
2fmCon
Duplex
TwnhsE
Twnhs
HouseStyle
2Story
1Story
1.5Fin
1.5Unf
SFoyer
SLvl
2.5Unf
2.5Fin
RoofStyle
RoofMatl
Exterior1st



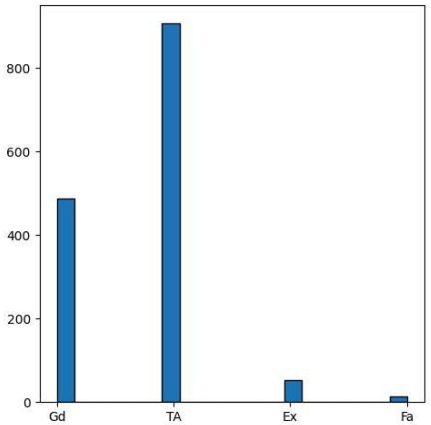




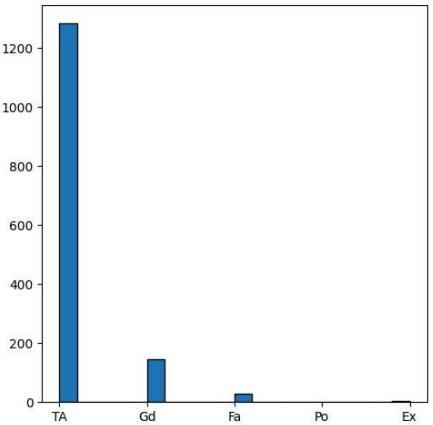
Exterior2nd



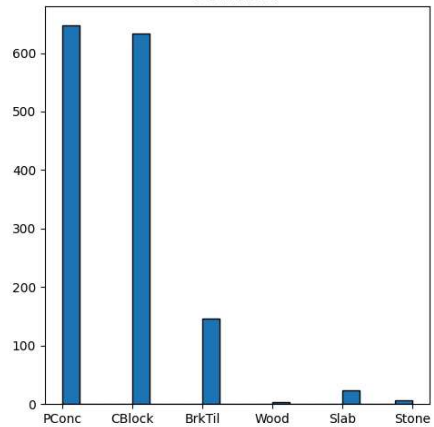
ExterQual



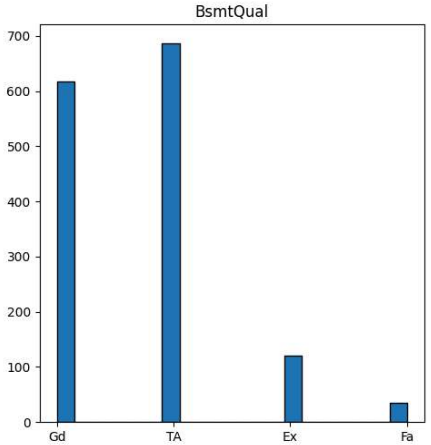
ExterCond



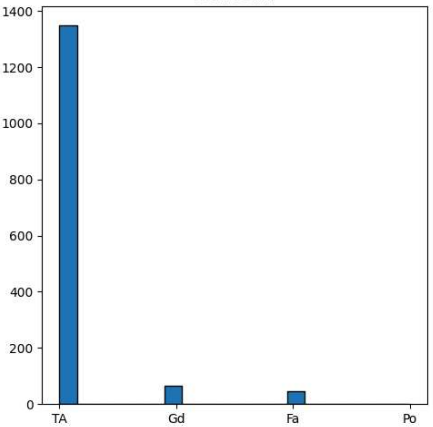
Foundation



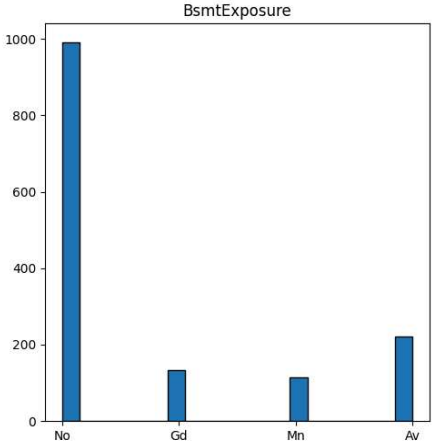
BsmtQual



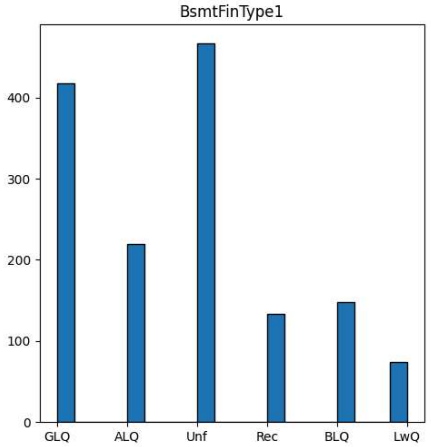
BsmtCond



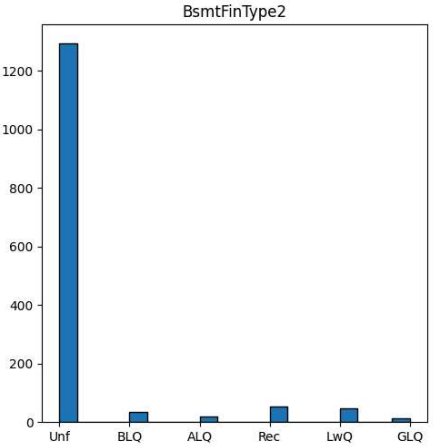
BsmtExposure



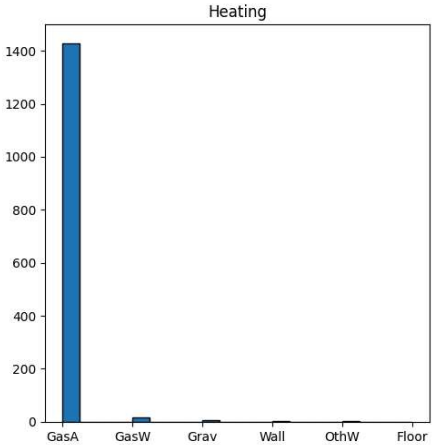
BsmtFinType1



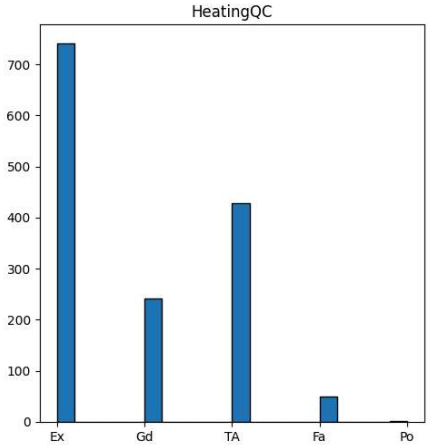
BsmtFinType2



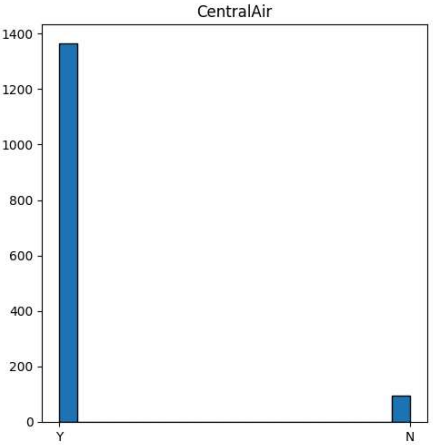
Heating



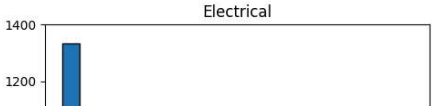
HeatingQC



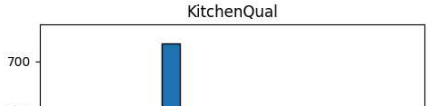
CentralAir



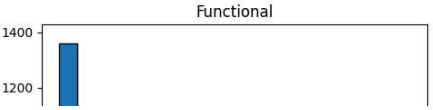
Electrical

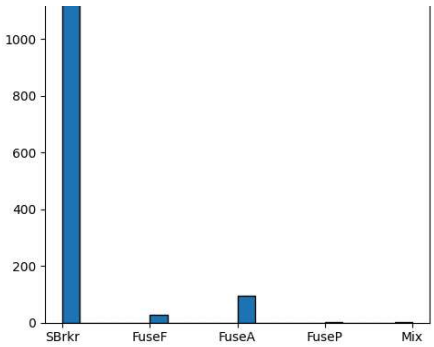


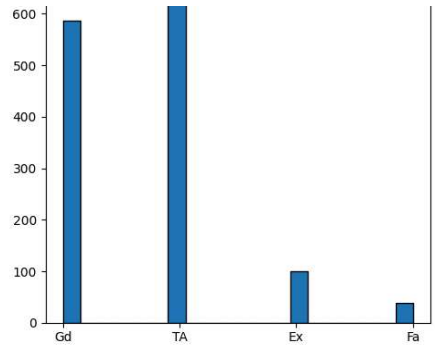
KitchenQual

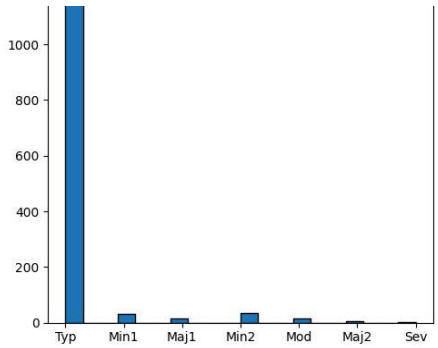


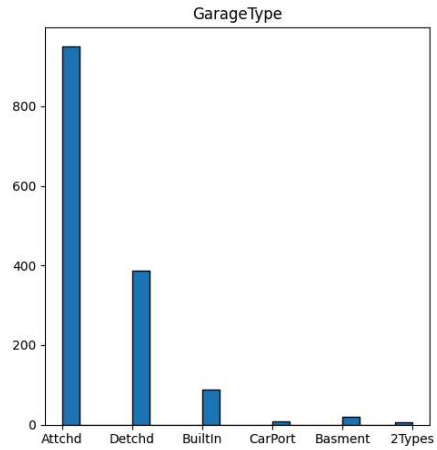
Functional

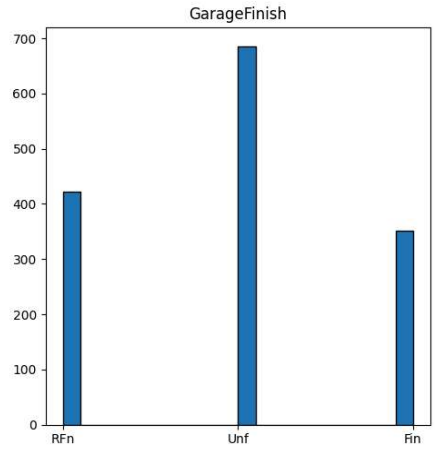


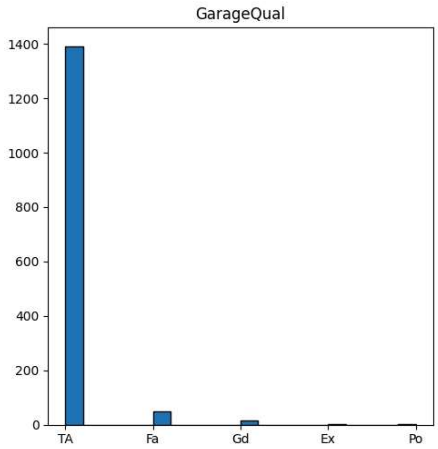

0
200
400
600
800
1000
SBrkr
FuseF
FuseA
FuseP
Mix

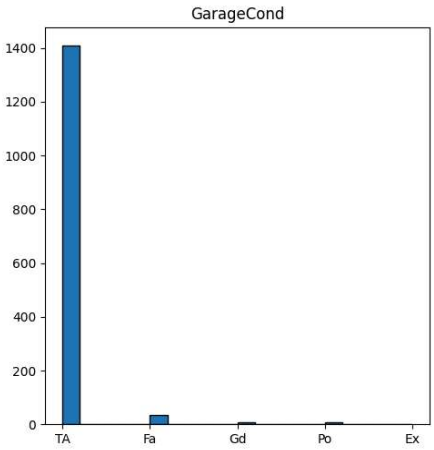

0
100
200
300
400
500
600
Gd
TA
Ex
Fa

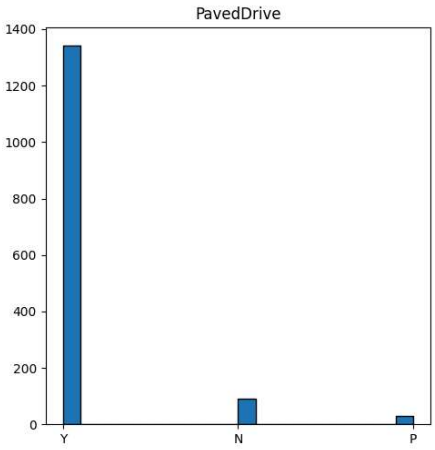

0
200
400
600
800
1000
Typ
Min1
Maj1
Min2
Mod
Maj2
Sev

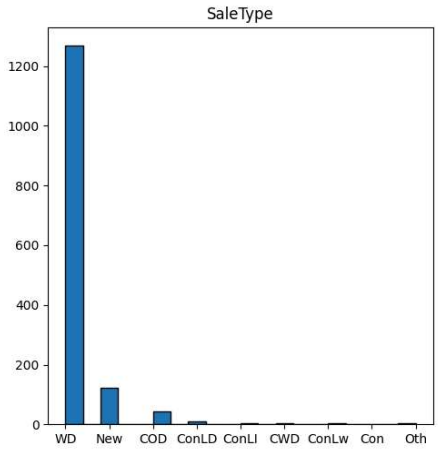

GarageType
0
200
400
600
800
Attchd
Detchd
BuiltIn
CarPort
Basment
2Types

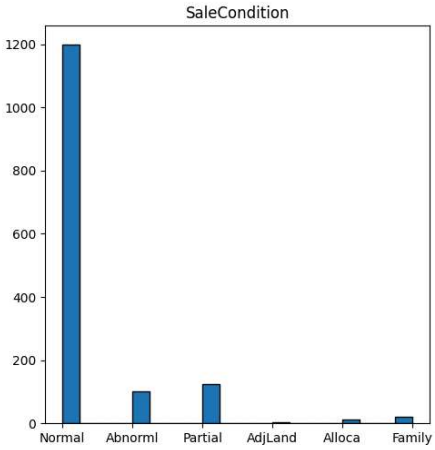

GarageFinish
0
100
200
300
400
500
600
700
RFn
Unf
Fin


GarageQual
0
200
400
600
800
1000
1200
1400
TA
Fa
Gd
Ex
Po


GarageCond
0
200
400
600
800
1000
1200
1400
TA
Fa
Gd
Po
Ex


PavedDrive
0
200
400
600
800
1000
1200
1400
Y
N
P


SaleType
0
200
400
600
800
1000
1200
WD
New
COD
ConLD
ConLI
CWD
ConLw
Con
Oth


SaleCondition
0
200
400
600
800
1000
1200
Normal
Abnorml
Partial
AdjLand
Alloca
Family

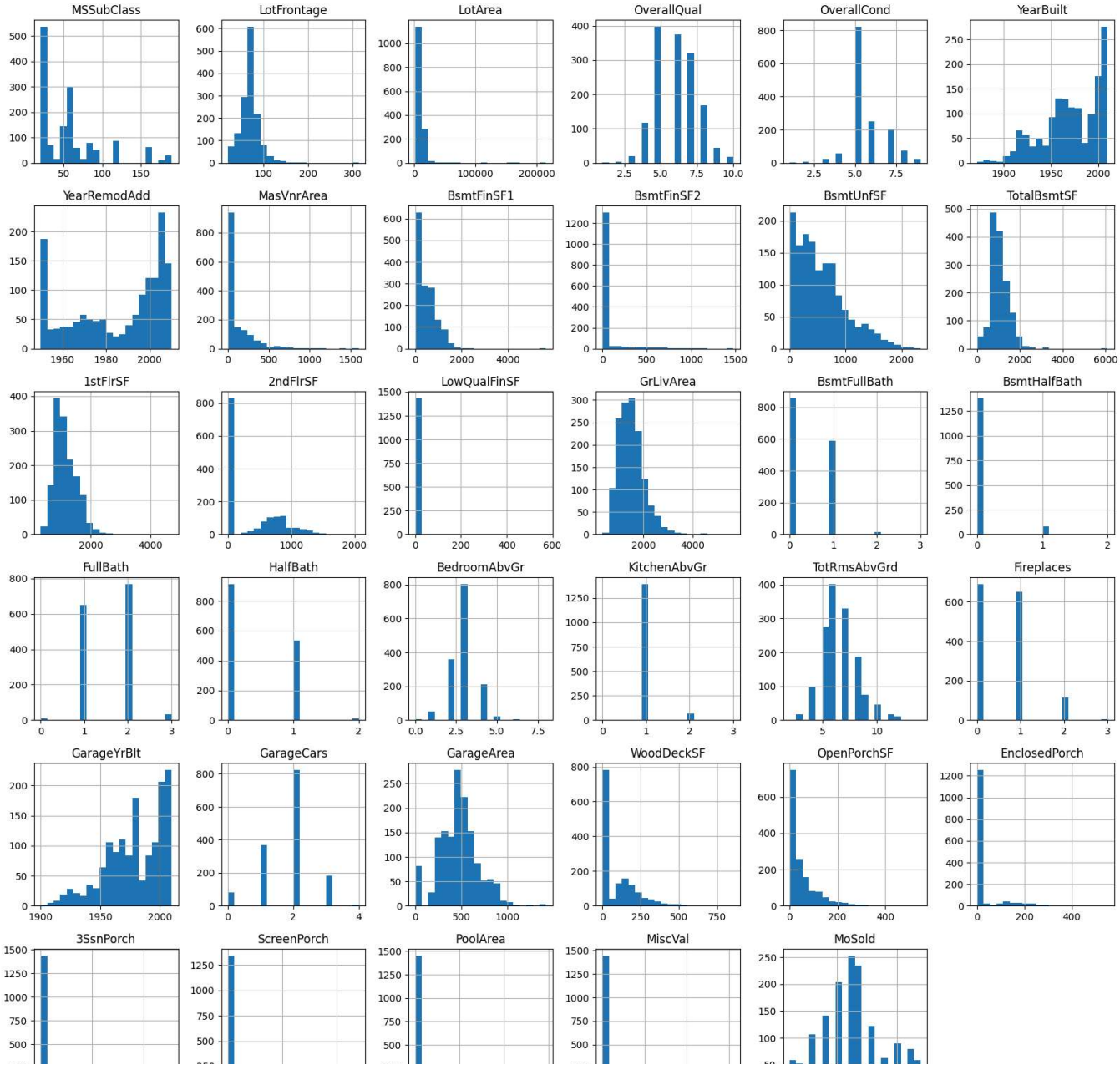
```
from sklearn.preprocessing import LabelEncoder
les = dict()
for col in obj_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    df[col] += 0.5
    les[col] = le
numerical_col =  []
for col in df.columns:
    if col not in obj_cols:
        numerical_col.append(col)
print(numerical_col)
```

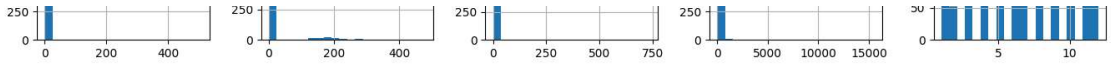
```
['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'Bsm
```

Data Visualization

```
numerical_col = numerical_col[:-1]
df[numerical_col].hist(bins = 20,figsize = (20,20))
```

```
array([[<Axes: title={'center': 'MSSubClass'}>,
        <Axes: title={'center': 'LotFrontage'}>,
        <Axes: title={'center': 'LotArea'}>,
        <Axes: title={'center': 'OverallQual'}>,
        <Axes: title={'center': 'OverallCond'}>,
        <Axes: title={'center': 'YearBuilt'}>],
       [<Axes: title={'center': 'YearRemodAdd'}>,
        <Axes: title={'center': 'MasVnrArea'}>,
        <Axes: title={'center': 'BsmtFinSF1'}>,
        <Axes: title={'center': 'BsmtFinSF2'}>,
        <Axes: title={'center': 'BsmtUnfSF'}>,
        <Axes: title={'center': 'TotalBsmtSF'}>],
       [<Axes: title={'center': '1stFlrSF'}>,
        <Axes: title={'center': '2ndFlrSF'}>,
        <Axes: title={'center': 'LowQualFinSF'}>,
        <Axes: title={'center': 'GrLivArea'}>,
        <Axes: title={'center': 'BsmtFullBath'}>,
        <Axes: title={'center': 'BsmtHalfBath'}>],
       [<Axes: title={'center': 'FullBath'}>,
        <Axes: title={'center': 'HalfBath'}>,
        <Axes: title={'center': 'BedroomAbvGr'}>,
        <Axes: title={'center': 'KitchenAbvGr'}>,
        <Axes: title={'center': 'TotRmsAbvGrd'}>,
        <Axes: title={'center': 'Fireplaces'}>],
       [<Axes: title={'center': 'GarageYrBlt'}>,
        <Axes: title={'center': 'GarageCars'}>,
        <Axes: title={'center': 'GarageArea'}>,
        <Axes: title={'center': 'WoodDeckSF'}>,
        <Axes: title={'center': 'OpenPorchSF'}>,
        <Axes: title={'center': 'EnclosedPorch'}>],
       [<Axes: title={'center': '3SsnPorch'}>,
        <Axes: title={'center': 'ScreenPorch'}>,
        <Axes: title={'center': 'PoolArea'}>,
        <Axes: title={'center': 'MiscVal'}>,
        <Axes: title={'center': 'MoSold'}>, <Axes: >]], dtype=object)
```






```
continous_cols = ['MSSubClass','LotFrontage','LotArea','YearBuilt','TotalBsmtSF','BsmtUnfSF',
                  'BsmtFinSF2','BsmtFinSF1','YearRemodAdd','MasVnrArea','1stFlrSF', '2ndFlrSF',
                  'LowQualFinSF','GrLivArea', 'GarageYrBlt','GarageArea','WoodDeckSF','OpenPorchSF',
                  'EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal','MoSold']
import seaborn as sns

for col in continous_cols:
    plt.boxplot(df[col])
    plt.title(col)
    plt.show()
```