How to Preprocess Data in Python Step-by-Step

Aurthor Shoaib Farooq

```
import pandas as pd
import numpy as np

df = pd.read_csv('house_data.csv')
df
```

```
         Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley
LotShape  \
0         1          60       RL         65.0     8450   Pave   NaN
Reg
1         2          20       RL         80.0     9600   Pave   NaN
Reg
2         3          60       RL         68.0    11250   Pave   NaN
IR1
3         4          70       RL         60.0     9550   Pave   NaN
IR1
4         5          60       RL         84.0    14260   Pave   NaN
IR1
...     ...         ...      ...          ...      ...    ...   ...
...
1455   1456          60       RL         62.0     7917   Pave   NaN
Reg
1456   1457          20       RL         85.0    13175   Pave   NaN
Reg
1457   1458          70       RL         66.0     9042   Pave   NaN
Reg
1458   1459          20       RL         68.0     9717   Pave   NaN
Reg
1459   1460          20       RL         75.0     9937   Pave   NaN
Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature
MiscVal  \
0            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
2            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
3            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
4            Lvl    AllPub  ...        0    NaN    NaN         NaN
0
```

```
...          ...       ...  ...      ...    ...    ...          ...
...
1455         Lvl    AllPub  ...        0    NaN    NaN          NaN
0
1456         Lvl    AllPub  ...        0    NaN  MnPrv          NaN
0
1457         Lvl    AllPub  ...        0    NaN  GdPrv         Shed
2500
1458         Lvl    AllPub  ...        0    NaN    NaN          NaN
0
1459         Lvl    AllPub  ...        0    NaN    NaN          NaN
0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
0         2   2008        WD         Normal     208500
1         5   2007        WD         Normal     181500
2         9   2008        WD         Normal     223500
3         2   2006        WD        Abnorml     140000
4        12   2008        WD         Normal     250000
...     ...    ...       ...            ...        ...
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[1460 rows x 81 columns]

df.shape

(1460, 81)

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
```

```
11  LandSlope       1460 non-null   object
12  Neighborhood    1460 non-null   object
13  Condition1      1460 non-null   object
14  Condition2      1460 non-null   object
15  BldgType        1460 non-null   object
16  HouseStyle      1460 non-null   object
17  OverallQual     1460 non-null   int64
18  OverallCond     1460 non-null   int64
19  YearBuilt       1460 non-null   int64
20  YearRemodAdd    1460 non-null   int64
21  RoofStyle       1460 non-null   object
22  RoofMatl        1460 non-null   object
23  Exterior1st     1460 non-null   object
24  Exterior2nd     1460 non-null   object
25  MasVnrType      588 non-null    object
26  MasVnrArea      1452 non-null   float64
27  ExterQual       1460 non-null   object
28  ExterCond       1460 non-null   object
29  Foundation      1460 non-null   object
30  BsmtQual        1423 non-null   object
31  BsmtCond        1423 non-null   object
32  BsmtExposure    1422 non-null   object
33  BsmtFinType1    1423 non-null   object
34  BsmtFinSF1      1460 non-null   int64
35  BsmtFinType2    1422 non-null   object
36  BsmtFinSF2      1460 non-null   int64
37  BsmtUnfSF       1460 non-null   int64
38  TotalBsmtSF     1460 non-null   int64
39  Heating         1460 non-null   object
40  HeatingQC       1460 non-null   object
41  CentralAir      1460 non-null   object
42  Electrical      1459 non-null   object
43  1stFlrSF        1460 non-null   int64
44  2ndFlrSF        1460 non-null   int64
45  LowQualFinSF    1460 non-null   int64
46  GrLivArea       1460 non-null   int64
47  BsmtFullBath    1460 non-null   int64
48  BsmtHalfBath    1460 non-null   int64
49  FullBath        1460 non-null   int64
50  HalfBath        1460 non-null   int64
51  BedroomAbvGr    1460 non-null   int64
52  KitchenAbvGr    1460 non-null   int64
53  KitchenQual     1460 non-null   object
54  TotRmsAbvGrd    1460 non-null   int64
55  Functional      1460 non-null   object
56  Fireplaces      1460 non-null   int64
57  FireplaceQu     770 non-null    object
58  GarageType      1379 non-null   object
59  GarageYrBlt     1379 non-null   float64
```

```
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB

df.isnull().sum().sum()

np.int64(7829)

df.head(2)

   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
\
0   1          60       RL         65.0     8450   Pave   NaN      Reg

1   2          20       RL         80.0     9600   Pave   NaN      Reg


  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal
MoSold  \
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
5

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500

[2 rows x 81 columns]

df.tail(5)
```

```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley
LotShape  \
1455  1456          60       RL         62.0     7917   Pave   NaN
Reg
1456  1457          20       RL         85.0    13175   Pave   NaN
Reg
1457  1458          70       RL         66.0     9042   Pave   NaN
Reg
1458  1459          20       RL         68.0     9717   Pave   NaN
Reg
1459  1460          20       RL         75.0     9937   Pave   NaN
Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature
MiscVal  \
1455         Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1456         Lvl    AllPub  ...        0    NaN  MnPrv         NaN
0
1457         Lvl    AllPub  ...        0    NaN  GdPrv        Shed
2500
1458         Lvl    AllPub  ...        0    NaN    NaN         NaN
0
1459         Lvl    AllPub  ...        0    NaN    NaN         NaN
0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
1455      8   2007        WD         Normal     175000
1456      2   2010        WD         Normal     210000
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500

[5 rows x 81 columns]

df.describe()

                Id   MSSubClass  LotFrontage        LotArea
OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000
1460.000000
mean    730.500000    56.897260    70.049958   10516.828082
6.099315
std     421.610009    42.300571    24.284752    9981.264932
1.382997
min       1.000000    20.000000    21.000000    1300.000000
1.000000
25%     365.750000    20.000000    59.000000    7553.500000
5.000000
50%     730.500000    50.000000    69.000000    9478.500000
```

```
6.000000
75%    1095.250000    70.000000    80.000000   11601.500000
7.000000
max    1460.000000   190.000000   313.000000  215245.000000
10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea
BsmtFinSF1  ...  \
count  1460.000000  1460.000000   1460.000000  1452.000000
1460.000000  ...
mean      5.575342  1971.267808   1984.865753   103.685262
443.639726  ...
std       1.112799    30.202904     20.645407   181.066207
456.098091  ...
min       1.000000  1872.000000   1950.000000     0.000000
0.000000  ...
25%       5.000000  1954.000000   1967.000000     0.000000
0.000000  ...
50%       5.000000  1973.000000   1994.000000     0.000000
383.500000  ...
75%       6.000000  2000.000000   2004.000000   166.000000
712.250000  ...
max       9.000000  2010.000000   2010.000000  1600.000000
5644.000000  ...

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch
ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000
1460.000000
mean     94.244521    46.660274      21.954110     3.409589
15.060959
std     125.338794    66.256028      61.119149    29.317331
55.757415
min       0.000000     0.000000       0.000000     0.000000
0.000000
25%       0.000000     0.000000       0.000000     0.000000
0.000000
50%       0.000000    25.000000       0.000000     0.000000
0.000000
75%     168.000000    68.000000       0.000000     0.000000
0.000000
max     857.000000   547.000000     552.000000   508.000000
480.000000

          PoolArea       MiscVal       MoSold       YrSold
SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000
1460.000000
mean      2.758904     43.489041     6.321918  2007.815753
180921.195890
```

```
std      40.177307    496.123024     2.703626     1.328095
79442.502883
min       0.000000      0.000000     1.000000  2006.000000
34900.000000
25%       0.000000      0.000000     5.000000  2007.000000
129975.000000
50%       0.000000      0.000000     6.000000  2008.000000
163000.000000
75%       0.000000      0.000000     8.000000  2009.000000
214000.000000
max     738.000000  15500.000000    12.000000  2010.000000
755000.000000

[8 rows x 38 columns]
```

```
df.isnull().sum()
```

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

```
duplicates_mask = df.duplicated()
duplicates_mask.sum()
```

```
np.int64(0)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
```

```
5   Street          1460 non-null   object
6   Alley           91 non-null     object
7   LotShape        1460 non-null   object
8   LandContour     1460 non-null   object
9   Utilities       1460 non-null   object
10  LotConfig       1460 non-null   object
11  LandSlope       1460 non-null   object
12  Neighborhood    1460 non-null   object
13  Condition1      1460 non-null   object
14  Condition2      1460 non-null   object
15  BldgType        1460 non-null   object
16  HouseStyle      1460 non-null   object
17  OverallQual     1460 non-null   int64
18  OverallCond     1460 non-null   int64
19  YearBuilt       1460 non-null   int64
20  YearRemodAdd    1460 non-null   int64
21  RoofStyle       1460 non-null   object
22  RoofMatl        1460 non-null   object
23  Exterior1st     1460 non-null   object
24  Exterior2nd     1460 non-null   object
25  MasVnrType      588 non-null    object
26  MasVnrArea      1452 non-null   float64
27  ExterQual       1460 non-null   object
28  ExterCond       1460 non-null   object
29  Foundation      1460 non-null   object
30  BsmtQual        1423 non-null   object
31  BsmtCond        1423 non-null   object
32  BsmtExposure    1422 non-null   object
33  BsmtFinType1    1423 non-null   object
34  BsmtFinSF1      1460 non-null   int64
35  BsmtFinType2    1422 non-null   object
36  BsmtFinSF2      1460 non-null   int64
37  BsmtUnfSF       1460 non-null   int64
38  TotalBsmtSF     1460 non-null   int64
39  Heating         1460 non-null   object
40  HeatingQC       1460 non-null   object
41  CentralAir      1460 non-null   object
42  Electrical      1459 non-null   object
43  1stFlrSF        1460 non-null   int64
44  2ndFlrSF        1460 non-null   int64
45  LowQualFinSF    1460 non-null   int64
46  GrLivArea       1460 non-null   int64
47  BsmtFullBath    1460 non-null   int64
48  BsmtHalfBath    1460 non-null   int64
49  FullBath        1460 non-null   int64
50  HalfBath        1460 non-null   int64
51  BedroomAbvGr    1460 non-null   int64
52  KitchenAbvGr    1460 non-null   int64
53  KitchenQual     1460 non-null   object
```

```
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
df['LotFrontage'].mean()
```

```
np.float64(70.04995836802665)
```

```
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].mean())
```

```
df['Alley'].value_counts()
```

```
Alley
Grvl    50
Pave    41
Name: count, dtype: int64
```

```
# axis = 1 for column drop
df = df.drop('Alley' , axis = 1)
df
```

```
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street LotShape
\
0        1          60       RL         65.0     8450   Pave      Reg
```

```
1        2          20       RL         80.0     9600   Pave      Reg
2        3          60       RL         68.0    11250   Pave      IR1
3        4          70       RL         60.0     9550   Pave      IR1
4        5          60       RL         84.0    14260   Pave      IR1
...    ...         ...      ...          ...      ...    ...      ...
1455  1456          60       RL         62.0     7917   Pave      Reg
1456  1457          20       RL         85.0    13175   Pave      Reg
1457  1458          70       RL         66.0     9042   Pave      Reg
1458  1459          20       RL         68.0     9717   Pave      Reg
1459  1460          20       RL         75.0     9937   Pave      Reg

     LandContour Utilities LotConfig  ... PoolArea PoolQC  Fence
MiscFeature  \
0            Lvl    AllPub    Inside  ...        0    NaN    NaN
NaN
1            Lvl    AllPub       FR2  ...        0    NaN    NaN
NaN
2            Lvl    AllPub    Inside  ...        0    NaN    NaN
NaN
3            Lvl    AllPub    Corner  ...        0    NaN    NaN
NaN
4            Lvl    AllPub       FR2  ...        0    NaN    NaN
NaN
...          ...       ...       ...  ...      ...    ...    ...
...
1455         Lvl    AllPub    Inside  ...        0    NaN    NaN
NaN
1456         Lvl    AllPub    Inside  ...        0    NaN  MnPrv
NaN
1457         Lvl    AllPub    Inside  ...        0    NaN  GdPrv
Shed
1458         Lvl    AllPub    Inside  ...        0    NaN    NaN
NaN
1459         Lvl    AllPub    Inside  ...        0    NaN    NaN
NaN

     MiscVal MoSold  YrSold  SaleType  SaleCondition  SalePrice
0          0      2    2008        WD         Normal     208500
1          0      5    2007        WD         Normal     181500
```

```
2          0      9    2008        WD         Normal     223500
3          0      2    2006        WD        Abnorml     140000
4          0     12    2008        WD         Normal     250000
...      ...    ...     ...       ...            ...        ...
1455       0      8    2007        WD         Normal     175000
1456       0      2    2010        WD         Normal     210000
1457    2500      5    2010        WD         Normal     266500
1458       0      4    2010        WD         Normal     142125
1459       0      6    2008        WD         Normal     147500

[1460 rows x 80 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
 14  BldgType       1460 non-null   object
 15  HouseStyle     1460 non-null   object
 16  OverallQual    1460 non-null   int64
 17  OverallCond    1460 non-null   int64
 18  YearBuilt      1460 non-null   int64
 19  YearRemodAdd   1460 non-null   int64
 20  RoofStyle      1460 non-null   object
 21  RoofMatl       1460 non-null   object
 22  Exterior1st    1460 non-null   object
 23  Exterior2nd    1460 non-null   object
 24  MasVnrType     588 non-null    object
 25  MasVnrArea     1452 non-null   float64
 26  ExterQual      1460 non-null   object
 27  ExterCond      1460 non-null   object
 28  Foundation     1460 non-null   object
 29  BsmtQual       1423 non-null   object
 30  BsmtCond       1423 non-null   object
```

```
31  BsmtExposure   1422 non-null   object
32  BsmtFinType1   1423 non-null   object
33  BsmtFinSF1     1460 non-null   int64
34  BsmtFinType2   1422 non-null   object
35  BsmtFinSF2     1460 non-null   int64
36  BsmtUnfSF      1460 non-null   int64
37  TotalBsmtSF    1460 non-null   int64
38  Heating        1460 non-null   object
39  HeatingQC      1460 non-null   object
40  CentralAir     1460 non-null   object
41  Electrical     1459 non-null   object
42  1stFlrSF       1460 non-null   int64
43  2ndFlrSF       1460 non-null   int64
44  LowQualFinSF   1460 non-null   int64
45  GrLivArea      1460 non-null   int64
46  BsmtFullBath   1460 non-null   int64
47  BsmtHalfBath   1460 non-null   int64
48  FullBath       1460 non-null   int64
49  HalfBath       1460 non-null   int64
50  BedroomAbvGr   1460 non-null   int64
51  KitchenAbvGr   1460 non-null   int64
52  KitchenQual    1460 non-null   object
53  TotRmsAbvGrd   1460 non-null   int64
54  Functional     1460 non-null   object
55  Fireplaces     1460 non-null   int64
56  FireplaceQu    770 non-null    object
57  GarageType     1379 non-null   object
58  GarageYrBlt    1379 non-null   float64
59  GarageFinish   1379 non-null   object
60  GarageCars     1460 non-null   int64
61  GarageArea     1460 non-null   int64
62  GarageQual     1379 non-null   object
63  GarageCond     1379 non-null   object
64  PavedDrive     1460 non-null   object
65  WoodDeckSF     1460 non-null   int64
66  OpenPorchSF    1460 non-null   int64
67  EnclosedPorch  1460 non-null   int64
68  3SsnPorch      1460 non-null   int64
69  ScreenPorch    1460 non-null   int64
70  PoolArea       1460 non-null   int64
71  PoolQC         7 non-null      object
72  Fence          281 non-null    object
73  MiscFeature    54 non-null     object
74  MiscVal        1460 non-null   int64
75  MoSold         1460 non-null   int64
76  YrSold         1460 non-null   int64
77  SaleType       1460 non-null   object
78  SaleCondition  1460 non-null   object
79  SalePrice      1460 non-null   int64
```

```
dtypes: float64(3), int64(35), object(42)
memory usage: 912.6+ KB
```

```
for i in df.columns:
    if(df[i].dtype == 'object' and df[i].isna().sum()!=0):
        print(i , df[i].dtype , df[i].isna().sum())
```

```
MasVnrType object 872
BsmtQual object 37
BsmtCond object 37
BsmtExposure object 38
BsmtFinType1 object 37
BsmtFinType2 object 38
Electrical object 1
FireplaceQu object 690
GarageType object 81
GarageFinish object 81
GarageQual object 81
GarageCond object 81
PoolQC object 1453
Fence object 1179
MiscFeature object 1406
```

```
# axis = 1 for column drop
df = df.drop([ 'MiscFeature' , 'Fence','PoolQC',
'FireplaceQu','MasVnrType'] , axis = 1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1460 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
 14  BldgType       1460 non-null   object
 15  HouseStyle     1460 non-null   object
 16  OverallQual    1460 non-null   int64
```

```
17  OverallCond    1460 non-null   int64
18  YearBuilt      1460 non-null   int64
19  YearRemodAdd   1460 non-null   int64
20  RoofStyle      1460 non-null   object
21  RoofMatl       1460 non-null   object
22  Exterior1st    1460 non-null   object
23  Exterior2nd    1460 non-null   object
24  MasVnrArea     1452 non-null   float64
25  ExterQual      1460 non-null   object
26  ExterCond      1460 non-null   object
27  Foundation     1460 non-null   object
28  BsmtQual       1423 non-null   object
29  BsmtCond       1423 non-null   object
30  BsmtExposure   1422 non-null   object
31  BsmtFinType1   1423 non-null   object
32  BsmtFinSF1     1460 non-null   int64
33  BsmtFinType2   1422 non-null   object
34  BsmtFinSF2     1460 non-null   int64
35  BsmtUnfSF      1460 non-null   int64
36  TotalBsmtSF    1460 non-null   int64
37  Heating        1460 non-null   object
38  HeatingQC      1460 non-null   object
39  CentralAir     1460 non-null   object
40  Electrical     1459 non-null   object
41  1stFlrSF       1460 non-null   int64
42  2ndFlrSF       1460 non-null   int64
43  LowQualFinSF   1460 non-null   int64
44  GrLivArea      1460 non-null   int64
45  BsmtFullBath   1460 non-null   int64
46  BsmtHalfBath   1460 non-null   int64
47  FullBath       1460 non-null   int64
48  HalfBath       1460 non-null   int64
49  BedroomAbvGr   1460 non-null   int64
50  KitchenAbvGr   1460 non-null   int64
51  KitchenQual    1460 non-null   object
52  TotRmsAbvGrd   1460 non-null   int64
53  Functional     1460 non-null   object
54  Fireplaces     1460 non-null   int64
55  GarageType     1379 non-null   object
56  GarageYrBlt    1379 non-null   float64
57  GarageFinish   1379 non-null   object
58  GarageCars     1460 non-null   int64
59  GarageArea     1460 non-null   int64
60  GarageQual     1379 non-null   object
61  GarageCond     1379 non-null   object
62  PavedDrive     1460 non-null   object
63  WoodDeckSF     1460 non-null   int64
64  OpenPorchSF    1460 non-null   int64
65  EnclosedPorch  1460 non-null   int64
```

```
 66  3SsnPorch      1460 non-null   int64
 67  ScreenPorch    1460 non-null   int64
 68  PoolArea       1460 non-null   int64
 69  MiscVal        1460 non-null   int64
 70  MoSold         1460 non-null   int64
 71  YrSold         1460 non-null   int64
 72  SaleType       1460 non-null   object
 73  SaleCondition  1460 non-null   object
 74  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(37)
memory usage: 855.6+ KB
```

```
for i in df.columns:
    if(df[i].dtype == 'object' and df[i].isna().sum()!=0):
        print(i , df[i].dtype , df[i].isna().sum())
```

```
BsmtQual object 37
BsmtCond object 37
BsmtExposure object 38
BsmtFinType1 object 37
BsmtFinType2 object 38
Electrical object 1
GarageType object 81
GarageFinish object 81
GarageQual object 81
GarageCond object 81
```

```
df.dropna(subset = 'BsmtQual' , axis = 0 , inplace = True)
```

```
df.dropna(subset = 'BsmtExposure' , axis = 0 , inplace = True)
```

```
df.dropna(subset = 'BsmtFinType2' , axis = 0 , inplace = True)
```

```
df.dropna(subset = 'Electrical' , axis = 0 , inplace = True)
```

```
df.dropna(subset = 'GarageType' , axis = 0 , inplace = True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1346 non-null   int64
 1   MSSubClass     1346 non-null   int64
 2   MSZoning       1346 non-null   object
 3   LotFrontage    1346 non-null   float64
 4   LotArea        1346 non-null   int64
 5   Street         1346 non-null   object
 6   LotShape       1346 non-null   object
```

```
7   LandContour    1346 non-null   object
8   Utilities      1346 non-null   object
9   LotConfig      1346 non-null   object
10  LandSlope      1346 non-null   object
11  Neighborhood   1346 non-null   object
12  Condition1     1346 non-null   object
13  Condition2     1346 non-null   object
14  BldgType       1346 non-null   object
15  HouseStyle     1346 non-null   object
16  OverallQual    1346 non-null   int64
17  OverallCond    1346 non-null   int64
18  YearBuilt      1346 non-null   int64
19  YearRemodAdd   1346 non-null   int64
20  RoofStyle      1346 non-null   object
21  RoofMatl       1346 non-null   object
22  Exterior1st    1346 non-null   object
23  Exterior2nd    1346 non-null   object
24  MasVnrArea     1338 non-null   float64
25  ExterQual      1346 non-null   object
26  ExterCond      1346 non-null   object
27  Foundation     1346 non-null   object
28  BsmtQual       1346 non-null   object
29  BsmtCond       1346 non-null   object
30  BsmtExposure   1346 non-null   object
31  BsmtFinType1   1346 non-null   object
32  BsmtFinSF1     1346 non-null   int64
33  BsmtFinType2   1346 non-null   object
34  BsmtFinSF2     1346 non-null   int64
35  BsmtUnfSF      1346 non-null   int64
36  TotalBsmtSF    1346 non-null   int64
37  Heating        1346 non-null   object
38  HeatingQC      1346 non-null   object
39  CentralAir     1346 non-null   object
40  Electrical     1346 non-null   object
41  1stFlrSF       1346 non-null   int64
42  2ndFlrSF       1346 non-null   int64
43  LowQualFinSF   1346 non-null   int64
44  GrLivArea      1346 non-null   int64
45  BsmtFullBath   1346 non-null   int64
46  BsmtHalfBath   1346 non-null   int64
47  FullBath       1346 non-null   int64
48  HalfBath       1346 non-null   int64
49  BedroomAbvGr   1346 non-null   int64
50  KitchenAbvGr   1346 non-null   int64
51  KitchenQual    1346 non-null   object
52  TotRmsAbvGrd   1346 non-null   int64
53  Functional     1346 non-null   object
54  Fireplaces     1346 non-null   int64
55  GarageType     1346 non-null   object
```

```
 56  GarageYrBlt    1346 non-null   float64
 57  GarageFinish   1346 non-null   object
 58  GarageCars     1346 non-null   int64
 59  GarageArea     1346 non-null   int64
 60  GarageQual     1346 non-null   object
 61  GarageCond     1346 non-null   object
 62  PavedDrive     1346 non-null   object
 63  WoodDeckSF     1346 non-null   int64
 64  OpenPorchSF    1346 non-null   int64
 65  EnclosedPorch  1346 non-null   int64
 66  3SsnPorch      1346 non-null   int64
 67  ScreenPorch    1346 non-null   int64
 68  PoolArea       1346 non-null   int64
 69  MiscVal        1346 non-null   int64
 70  MoSold         1346 non-null   int64
 71  YrSold         1346 non-null   int64
 72  SaleType       1346 non-null   object
 73  SaleCondition  1346 non-null   object
 74  SalePrice      1346 non-null   int64
dtypes: float64(3), int64(35), object(37)
memory usage: 799.2+ KB
```

```
for i in df.columns:
    if(df[i].dtype == 'int64' and df[i].isna().sum()!=0):
        print(i , df[i].dtype , df[i].isna().sum())

for i in df.columns:
    if(df[i].dtype == 'float64' and df[i].isna().sum()!=0):
        print(i , df[i].dtype , df[i].isna().sum())
```

```
MasVnrArea float64 8
```

```
df['LotFrontage']  =
df['LotFrontage'].fillna(df['LotFrontage'].mean())
df['MasVnrArea']  = df['MasVnrArea'].fillna(df['MasVnrArea'].mean())

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 75 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1346 non-null   int64
 1   MSSubClass     1346 non-null   int64
 2   MSZoning       1346 non-null   object
 3   LotFrontage    1346 non-null   float64
 4   LotArea        1346 non-null   int64
 5   Street         1346 non-null   object
 6   LotShape       1346 non-null   object
 7   LandContour    1346 non-null   object
```

```
8   Utilities      1346 non-null   object
9   LotConfig      1346 non-null   object
10  LandSlope      1346 non-null   object
11  Neighborhood   1346 non-null   object
12  Condition1     1346 non-null   object
13  Condition2     1346 non-null   object
14  BldgType       1346 non-null   object
15  HouseStyle     1346 non-null   object
16  OverallQual    1346 non-null   int64
17  OverallCond    1346 non-null   int64
18  YearBuilt      1346 non-null   int64
19  YearRemodAdd   1346 non-null   int64
20  RoofStyle      1346 non-null   object
21  RoofMatl       1346 non-null   object
22  Exterior1st    1346 non-null   object
23  Exterior2nd    1346 non-null   object
24  MasVnrArea     1346 non-null   float64
25  ExterQual      1346 non-null   object
26  ExterCond      1346 non-null   object
27  Foundation     1346 non-null   object
28  BsmtQual       1346 non-null   object
29  BsmtCond       1346 non-null   object
30  BsmtExposure   1346 non-null   object
31  BsmtFinType1   1346 non-null   object
32  BsmtFinSF1     1346 non-null   int64
33  BsmtFinType2   1346 non-null   object
34  BsmtFinSF2     1346 non-null   int64
35  BsmtUnfSF      1346 non-null   int64
36  TotalBsmtSF    1346 non-null   int64
37  Heating        1346 non-null   object
38  HeatingQC      1346 non-null   object
39  CentralAir     1346 non-null   object
40  Electrical     1346 non-null   object
41  1stFlrSF       1346 non-null   int64
42  2ndFlrSF       1346 non-null   int64
43  LowQualFinSF   1346 non-null   int64
44  GrLivArea      1346 non-null   int64
45  BsmtFullBath   1346 non-null   int64
46  BsmtHalfBath   1346 non-null   int64
47  FullBath       1346 non-null   int64
48  HalfBath       1346 non-null   int64
49  BedroomAbvGr   1346 non-null   int64
50  KitchenAbvGr   1346 non-null   int64
51  KitchenQual    1346 non-null   object
52  TotRmsAbvGrd   1346 non-null   int64
53  Functional     1346 non-null   object
54  Fireplaces     1346 non-null   int64
55  GarageType     1346 non-null   object
56  GarageYrBlt    1346 non-null   float64
```

```
 57  GarageFinish   1346 non-null   object
 58  GarageCars     1346 non-null   int64
 59  GarageArea     1346 non-null   int64
 60  GarageQual     1346 non-null   object
 61  GarageCond     1346 non-null   object
 62  PavedDrive     1346 non-null   object
 63  WoodDeckSF     1346 non-null   int64
 64  OpenPorchSF    1346 non-null   int64
 65  EnclosedPorch  1346 non-null   int64
 66  3SsnPorch      1346 non-null   int64
 67  ScreenPorch    1346 non-null   int64
 68  PoolArea       1346 non-null   int64
 69  MiscVal        1346 non-null   int64
 70  MoSold         1346 non-null   int64
 71  YrSold         1346 non-null   int64
 72  SaleType       1346 non-null   object
 73  SaleCondition  1346 non-null   object
 74  SalePrice      1346 non-null   int64
dtypes: float64(3), int64(35), object(37)
memory usage: 799.2+ KB
```

```
for i in df.columns:
    if(df[i].dtype == 'int64' and (df[i]==0).sum()!=0):
        print(i , df[i].dtype , (df[i]==0).sum())
```

```
BsmtFinSF1 int64 394
BsmtFinSF2 int64 1184
BsmtUnfSF int64 73
2ndFlrSF int64 753
LowQualFinSF int64 1328
BsmtFullBath int64 766
BsmtHalfBath int64 1266
FullBath int64 8
HalfBath int64 811
BedroomAbvGr int64 6
Fireplaces int64 595
WoodDeckSF int64 675
OpenPorchSF int64 575
EnclosedPorch int64 1162
3SsnPorch int64 1323
ScreenPorch int64 1230
PoolArea int64 1339
MiscVal int64 1299
```

```
# axis = 1 for column drop
df = df.drop(['MiscVal','PoolArea','ScreenPorch', '3SsnPorch' ,
'BsmtFinSF2' ,  'LowQualFinSF' ,'BsmtHalfBath' ,
```

```
'HalfBath' ,'EnclosedPorch'] , axis = 1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 66 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Id              1346 non-null   int64  
 1   MSSubClass      1346 non-null   int64  
 2   MSZoning        1346 non-null   object 
 3   LotFrontage     1346 non-null   float64
 4   LotArea         1346 non-null   int64  
 5   Street          1346 non-null   object 
 6   LotShape        1346 non-null   object 
 7   LandContour     1346 non-null   object 
 8   Utilities       1346 non-null   object 
 9   LotConfig       1346 non-null   object 
 10  LandSlope       1346 non-null   object 
 11  Neighborhood    1346 non-null   object 
 12  Condition1      1346 non-null   object 
 13  Condition2      1346 non-null   object 
 14  BldgType        1346 non-null   object 
 15  HouseStyle      1346 non-null   object 
 16  OverallQual     1346 non-null   int64  
 17  OverallCond     1346 non-null   int64  
 18  YearBuilt       1346 non-null   int64  
 19  YearRemodAdd    1346 non-null   int64  
 20  RoofStyle       1346 non-null   object 
 21  RoofMatl        1346 non-null   object 
 22  Exterior1st     1346 non-null   object 
 23  Exterior2nd     1346 non-null   object 
 24  MasVnrArea      1346 non-null   float64
 25  ExterQual       1346 non-null   object 
 26  ExterCond       1346 non-null   object 
 27  Foundation      1346 non-null   object 
 28  BsmtQual        1346 non-null   object 
 29  BsmtCond        1346 non-null   object 
 30  BsmtExposure    1346 non-null   object 
 31  BsmtFinType1    1346 non-null   object 
 32  BsmtFinSF1      1346 non-null   int64  
 33  BsmtFinType2    1346 non-null   object 
 34  BsmtUnfSF       1346 non-null   int64  
 35  TotalBsmtSF     1346 non-null   int64  
 36  Heating         1346 non-null   object 
 37  HeatingQC       1346 non-null   object 
 38  CentralAir      1346 non-null   object 
 39  Electrical      1346 non-null   object 
 40  1stFlrSF        1346 non-null   int64  
 41  2ndFlrSF        1346 non-null   int64  
```

```
 42  GrLivArea      1346 non-null   int64
 43  BsmtFullBath   1346 non-null   int64
 44  FullBath       1346 non-null   int64
 45  BedroomAbvGr   1346 non-null   int64
 46  KitchenAbvGr   1346 non-null   int64
 47  KitchenQual    1346 non-null   object
 48  TotRmsAbvGrd   1346 non-null   int64
 49  Functional     1346 non-null   object
 50  Fireplaces     1346 non-null   int64
 51  GarageType     1346 non-null   object
 52  GarageYrBlt    1346 non-null   float64
 53  GarageFinish   1346 non-null   object
 54  GarageCars     1346 non-null   int64
 55  GarageArea     1346 non-null   int64
 56  GarageQual     1346 non-null   object
 57  GarageCond     1346 non-null   object
 58  PavedDrive     1346 non-null   object
 59  WoodDeckSF     1346 non-null   int64
 60  OpenPorchSF    1346 non-null   int64
 61  MoSold         1346 non-null   int64
 62  YrSold         1346 non-null   int64
 63  SaleType       1346 non-null   object
 64  SaleCondition  1346 non-null   object
 65  SalePrice      1346 non-null   int64
dtypes: float64(3), int64(26), object(37)
memory usage: 704.5+ KB
```

```
for i in df.columns:
    if(df[i].dtype == 'int64' and (df[i]==0).sum()!=0):
        print(i , df[i].dtype , (df[i]==0).sum())
```

```
BsmtFinSF1 int64 394
BsmtUnfSF int64 73
2ndFlrSF int64 753
BsmtFullBath int64 766
FullBath int64 8
BedroomAbvGr int64 6
Fireplaces int64 595
WoodDeckSF int64 675
OpenPorchSF int64 575
```

```
df['BsmtFinSF1'].mean()
df['BsmtFinSF1'] = df['BsmtFinSF1'].replace( 0,
df['BsmtFinSF1'].mean())

df['BsmtUnfSF'] = df['BsmtUnfSF'].replace( 0, df['BsmtUnfSF'].mean())
df['2ndFlrSF'] = df['2ndFlrSF'].replace( 0, df['2ndFlrSF'].mean())
df['BsmtFullBath'] = df['BsmtFullBath'].replace( 0,
df['BsmtFullBath'].mean())
df['FullBath'] = df['FullBath'].replace( 0, df['FullBath'].mean())
```

```
df['BedroomAbvGr'] = df['BedroomAbvGr'].replace( 0,
df['BedroomAbvGr'].mean())
df['Fireplaces'] = df['Fireplaces'].replace( 0,
df['Fireplaces'].mean())
df['WoodDeckSF'] = df['WoodDeckSF'].replace( 0,
df['WoodDeckSF'].mean())
df['OpenPorchSF'] = df['OpenPorchSF'].replace( 0,
df['OpenPorchSF'].mean())

for i in df.columns:
    if(df[i].dtype == 'int64' and (df[i]==0).sum()!=0):
        print(i , df[i].dtype , (df[i]==0).sum())

for i in df.columns:
    if(df[i].dtype == 'float64' and (df[i]==0).sum()!=0):
        print(i , df[i].dtype , (df[i]==0).sum())
```

```
MasVnrArea float64 760
```

```
df['MasVnrArea'] = df['MasVnrArea'].replace( 0,
df['MasVnrArea'].mean())

for i in df.columns:
    if(df[i].dtype == 'object'):
        print(i , df[i].dtype)
```

```
MSZoning object
Street object
LotShape object
LandContour object
Utilities object
LotConfig object
LandSlope object
Neighborhood object
Condition1 object
Condition2 object
BldgType object
HouseStyle object
RoofStyle object
RoofMatl object
Exterior1st object
Exterior2nd object
ExterQual object
ExterCond object
Foundation object
BsmtQual object
BsmtCond object
BsmtExposure object
BsmtFinType1 object
BsmtFinType2 object
Heating object
```

```
HeatingQC object
CentralAir object
Electrical object
KitchenQual object
Functional object
GarageType object
GarageFinish object
GarageQual object
GarageCond object
PavedDrive object
SaleType object
SaleCondition object
```

```
for i in df.columns:
    if(df[i].dtype == 'object'):
        print(i , df[i].value_counts())
```

```
MSZoning MSZoning
RL         1071
RM          191
FV           65
RH           11
C (all)       8
Name: count, dtype: int64
Street Street
Pave    1341
Grvl       5
Name: count, dtype: int64
LotShape LotShape
Reg    835
IR1    461
IR2     40
IR3     10
Name: count, dtype: int64
LandContour LandContour
Lvl    1214
Bnk      52
HLS      48
Low      32
Name: count, dtype: int64
Utilities Utilities
AllPub    1345
NoSeWa       1
Name: count, dtype: int64
LotConfig LotConfig
Inside     963
Corner     245
CulDSac     91
FR2         43
FR3          4
```

```
Name: count, dtype: int64
LandSlope LandSlope
Gtl    1273
Mod      61
Sev      12
Name: count, dtype: int64
Neighborhood Neighborhood
NAmes      209
CollgCr    147
OldTown    100
Somerst     86
Gilbert     78
NridgHt     76
NWAmes      73
Edwards     70
Sawyer      69
SawyerW     54
Crawfor     51
BrkSide     47
Mitchel     42
NoRidge     41
Timber      37
IDOTRR      29
ClearCr     26
StoneBr     25
SWISU       20
Blmngtn     17
BrDale      15
MeadowV     12
Veenker     11
NPkVill      9
Blueste      2
Name: count, dtype: int64
Condition1 Condition1
Norm      1170
Feedr       63
Artery      43
RRAn        26
PosN        19
RRAe        10
PosA         8
RRNn         5
RRNe         2
Name: count, dtype: int64
Condition2 Condition2
Norm      1332
Feedr        5
Artery       2
RRNn         2
```

```
PosN         2
PosA         1
RRAn         1
RRAe         1
Name: count, dtype: int64
BldgType BldgType
1Fam      1145
TwnhsE     113
Twnhs       38
Duplex      28
2fmCon      22
Name: count, dtype: int64
HouseStyle HouseStyle
1Story    662
2Story    429
1.5Fin    134
SLvl       64
SFoyer     30
1.5Unf     11
2.5Unf     10
2.5Fin      6
Name: count, dtype: int64
RoofStyle RoofStyle
Gable      1044
Hip         273
Flat         11
Gambrel      10
Mansard       6
Shed          2
Name: count, dtype: int64
RoofMatl RoofMatl
CompShg    1322
Tar&Grv       9
WdShngl       6
WdShake       5
Metal         1
Membran       1
Roll          1
ClyTile       1
Name: count, dtype: int64
Exterior1st Exterior1st
VinylSd    491
HdBoard    211
MetalSd    201
Wd Sdng    184
Plywood    100
CemntBd     54
BrkFace     44
Stucco      21
```

```
WdShing     20
AsbShng     15
Stone        2
BrkComm      1
ImStucc      1
CBlock       1
Name: count, dtype: int64
Exterior2nd Exterior2nd
VinylSd    480
MetalSd    197
HdBoard    197
Wd Sdng    176
Plywood    127
CmentBd     53
Wd Shng     32
Stucco      23
BrkFace     22
AsbShng     16
ImStucc     10
Brk Cmn      6
Stone        3
AsphShn      2
Other        1
CBlock       1
Name: count, dtype: int64
ExterQual ExterQual
TA    803
Gd    484
Ex     52
Fa      7
Name: count, dtype: int64
ExterCond ExterCond
TA    1191
Gd     137
Fa      16
Ex       2
Name: count, dtype: int64
Foundation Foundation
PConc     628
CBlock    580
BrkTil    129
Stone       6
Wood        3
Name: count, dtype: int64
BsmtQual BsmtQual
Gd    598
TA    595
Ex    121
Fa     32
```

```
Name: count, dtype: int64
BsmtCond BsmtCond
TA    1244
Gd      63
Fa      38
Po       1
Name: count, dtype: int64
BsmtExposure BsmtExposure
No    894
Av    213
Gd    128
Mn    111
Name: count, dtype: int64
BsmtFinType1 BsmtFinType1
GLQ    407
Unf    394
ALQ    209
BLQ    141
Rec    126
LwQ     69
Name: count, dtype: int64
BsmtFinType2 BsmtFinType2
Unf    1184
Rec      53
LwQ      46
BLQ      32
ALQ      19
GLQ      12
Name: count, dtype: int64
Heating Heating
GasA    1326
GasW      16
Grav       3
OthW       1
Name: count, dtype: int64
HeatingQC HeatingQC
Ex    711
TA    381
Gd    217
Fa     36
Po      1
Name: count, dtype: int64
CentralAir CentralAir
Y    1285
N      61
Name: count, dtype: int64
Electrical Electrical
SBrkr    1250
FuseA      76
```

```
FuseF      17
FuseP       2
Mix         1
Name: count, dtype: int64
KitchenQual KitchenQual
TA    651
Gd    574
Ex     98
Fa     23
Name: count, dtype: int64
Functional Functional
Typ     1261
Min2      30
Min1      28
Maj1      11
Mod       11
Maj2       4
Sev        1
Name: count, dtype: int64
GarageType GarageType
Attchd     859
Detchd     369
BuiltIn     86
Basment     19
CarPort      7
2Types       6
Name: count, dtype: int64
GarageFinish GarageFinish
Unf    580
RFn    417
Fin    349
Name: count, dtype: int64
GarageQual GarageQual
TA    1278
Fa      48
Gd      14
Ex       3
Po       3
Name: count, dtype: int64
GarageCond GarageCond
TA    1295
Fa      33
Gd       9
Po       7
Ex       2
Name: count, dtype: int64
PavedDrive PavedDrive
Y    1265
N      54
```

```
P      27
Name: count, dtype: int64
SaleType SaleType
WD       1163
New       120
COD        42
ConLD       6
ConLI       4
CWD         4
ConLw       4
Con         2
Oth         1
Name: count, dtype: int64
SaleCondition SaleCondition
Normal     1108
Partial     123
Abnorml      86
Family       20
Alloca        8
AdjLand       1
Name: count, dtype: int64
```

```
# df['GarageCond'] = df['GarageCond'].replace({'TA' : 0 ,
#                             'Fa' : 1 ,
#                             'Gd' : 2 ,
#                             'Po' : 3 ,
#                            'Ex' : 4 })

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for i in df.columns:
    if(df[i].dtype == 'object'):
        df[i] = le.fit_transform(df[i])
        print(i , df[i].value_counts())
```

```
MSZoning MSZoning
3    1071
4     191
1      65
2      11
0       8
Name: count, dtype: int64
Street Street
1    1341
0       5
Name: count, dtype: int64
LotShape LotShape
3    835
```

```
0    461
1     40
2     10
Name: count, dtype: int64
LandContour LandContour
3    1214
0      52
1      48
2      32
Name: count, dtype: int64
Utilities Utilities
0    1345
1       1
Name: count, dtype: int64
LotConfig LotConfig
4    963
0    245
1     91
2     43
3      4
Name: count, dtype: int64
LandSlope LandSlope
0    1273
1      61
2      12
Name: count, dtype: int64
Neighborhood Neighborhood
12    209
5     147
17    100
21     86
8      78
16     76
14     73
7      70
19     69
20     54
6      51
3      47
11     42
15     41
23     37
9      29
4      26
22     25
18     20
0      17
2      15
10     12
```

```
24     11
13      9
1       2
Name: count, dtype: int64
Condition1 Condition1
2    1170
1      63
0      43
6      26
4      19
5      10
3       8
8       5
7       2
Name: count, dtype: int64
Condition2 Condition2
2    1332
1       5
0       2
7       2
4       2
3       1
6       1
5       1
Name: count, dtype: int64
BldgType BldgType
0    1145
4     113
3      38
2      28
1      22
Name: count, dtype: int64
HouseStyle HouseStyle
2    662
5    429
0    134
7     64
6     30
1     11
4     10
3      6
Name: count, dtype: int64
RoofStyle RoofStyle
1    1044
3     273
0      11
2      10
4       6
5       2
```

```
Name: count, dtype: int64
RoofMatl RoofMatl
1    1322
5       9
7       6
6       5
3       1
2       1
4       1
0       1
Name: count, dtype: int64
Exterior1st Exterior1st
11    491
5     211
7     201
12    184
8     100
4      54
2      44
10     21
13     20
0      15
9       2
1       1
6       1
3       1
Name: count, dtype: int64
Exterior2nd Exterior2nd
13    480
8     197
6     197
14    176
10    127
5      53
15     32
12     23
3      22
0      16
7      10
2       6
11      3
1       2
9       1
4       1
Name: count, dtype: int64
ExterQual ExterQual
3    803
2    484
0     52
```

```
1      7
Name: count, dtype: int64
ExterCond ExterCond
3    1191
2     137
1      16
0       2
Name: count, dtype: int64
Foundation Foundation
2    628
1    580
0    129
3      6
4      3
Name: count, dtype: int64
BsmtQual BsmtQual
2    598
3    595
0    121
1     32
Name: count, dtype: int64
BsmtCond BsmtCond
3    1244
1      63
0      38
2       1
Name: count, dtype: int64
BsmtExposure BsmtExposure
3    894
0    213
1    128
2    111
Name: count, dtype: int64
BsmtFinType1 BsmtFinType1
2    407
5    394
0    209
1    141
4    126
3     69
Name: count, dtype: int64
BsmtFinType2 BsmtFinType2
5    1184
4      53
3      46
1      32
0      19
2      12
Name: count, dtype: int64
```

```
Heating Heating
0    1326
1      16
2       3
3       1
Name: count, dtype: int64
HeatingQC HeatingQC
0    711
4    381
2    217
1     36
3      1
Name: count, dtype: int64
CentralAir CentralAir
1    1285
0      61
Name: count, dtype: int64
Electrical Electrical
4    1250
0      76
1      17
2       2
3       1
Name: count, dtype: int64
KitchenQual KitchenQual
3    651
2    574
0     98
1     23
Name: count, dtype: int64
Functional Functional
6    1261
3      30
2      28
0      11
4      11
1       4
5       1
Name: count, dtype: int64
GarageType GarageType
1    859
5    369
3     86
2     19
4      7
0      6
Name: count, dtype: int64
GarageFinish GarageFinish
2    580
```

```
1    417
0    349
Name: count, dtype: int64
GarageQual GarageQual
4    1278
1      48
2      14
0       3
3       3
Name: count, dtype: int64
GarageCond GarageCond
4    1295
1      33
2       9
3       7
0       2
Name: count, dtype: int64
PavedDrive PavedDrive
2    1265
0      54
1      27
Name: count, dtype: int64
SaleType SaleType
8    1163
6     120
0      42
3       6
4       4
1       4
5       4
2       2
7       1
Name: count, dtype: int64
SaleCondition SaleCondition
4    1108
5     123
0      86
3      20
2       8
1       1
Name: count, dtype: int64

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 66 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1346 non-null   int64
```

```
1   MSSubClass     1346 non-null   int64
2   MSZoning       1346 non-null   int64
3   LotFrontage    1346 non-null   float64
4   LotArea        1346 non-null   int64
5   Street         1346 non-null   int64
6   LotShape       1346 non-null   int64
7   LandContour    1346 non-null   int64
8   Utilities      1346 non-null   int64
9   LotConfig      1346 non-null   int64
10  LandSlope      1346 non-null   int64
11  Neighborhood   1346 non-null   int64
12  Condition1     1346 non-null   int64
13  Condition2     1346 non-null   int64
14  BldgType       1346 non-null   int64
15  HouseStyle     1346 non-null   int64
16  OverallQual    1346 non-null   int64
17  OverallCond    1346 non-null   int64
18  YearBuilt      1346 non-null   int64
19  YearRemodAdd   1346 non-null   int64
20  RoofStyle      1346 non-null   int64
21  RoofMatl       1346 non-null   int64
22  Exterior1st    1346 non-null   int64
23  Exterior2nd    1346 non-null   int64
24  MasVnrArea     1346 non-null   float64
25  ExterQual      1346 non-null   int64
26  ExterCond      1346 non-null   int64
27  Foundation     1346 non-null   int64
28  BsmtQual       1346 non-null   int64
29  BsmtCond       1346 non-null   int64
30  BsmtExposure   1346 non-null   int64
31  BsmtFinType1   1346 non-null   int64
32  BsmtFinSF1     1346 non-null   float64
33  BsmtFinType2   1346 non-null   int64
34  BsmtUnfSF      1346 non-null   float64
35  TotalBsmtSF    1346 non-null   int64
36  Heating        1346 non-null   int64
37  HeatingQC      1346 non-null   int64
38  CentralAir     1346 non-null   int64
39  Electrical     1346 non-null   int64
40  1stFlrSF       1346 non-null   int64
41  2ndFlrSF       1346 non-null   float64
42  GrLivArea      1346 non-null   int64
43  BsmtFullBath   1346 non-null   float64
44  FullBath       1346 non-null   float64
45  BedroomAbvGr   1346 non-null   float64
46  KitchenAbvGr   1346 non-null   int64
47  KitchenQual    1346 non-null   int64
48  TotRmsAbvGrd   1346 non-null   int64
49  Functional     1346 non-null   int64
```

```
 50  Fireplaces     1346 non-null   float64
 51  GarageType     1346 non-null   int64
 52  GarageYrBlt    1346 non-null   float64
 53  GarageFinish   1346 non-null   int64
 54  GarageCars     1346 non-null   int64
 55  GarageArea     1346 non-null   int64
 56  GarageQual     1346 non-null   int64
 57  GarageCond     1346 non-null   int64
 58  PavedDrive     1346 non-null   int64
 59  WoodDeckSF     1346 non-null   float64
 60  OpenPorchSF    1346 non-null   float64
 61  MoSold         1346 non-null   int64
 62  YrSold         1346 non-null   int64
 63  SaleType       1346 non-null   int64
 64  SaleCondition  1346 non-null   int64
 65  SalePrice      1346 non-null   int64
dtypes: float64(12), int64(54)
memory usage: 704.5 KB
```

```
df.isna().sum().sum()
```

```
np.int64(0)
```

```
df.shape
```

```
(1346, 66)
```

```
# for column in missing_data.columns.values.tolist():
#     print(column)
#     print (missing_data[column].value_counts())
#     print("")
```

Correlation

Understanding the correlation between various features in the dataset

Positive Correlation

Negative Correlation

```
correlation = df.corr()
correlation
```

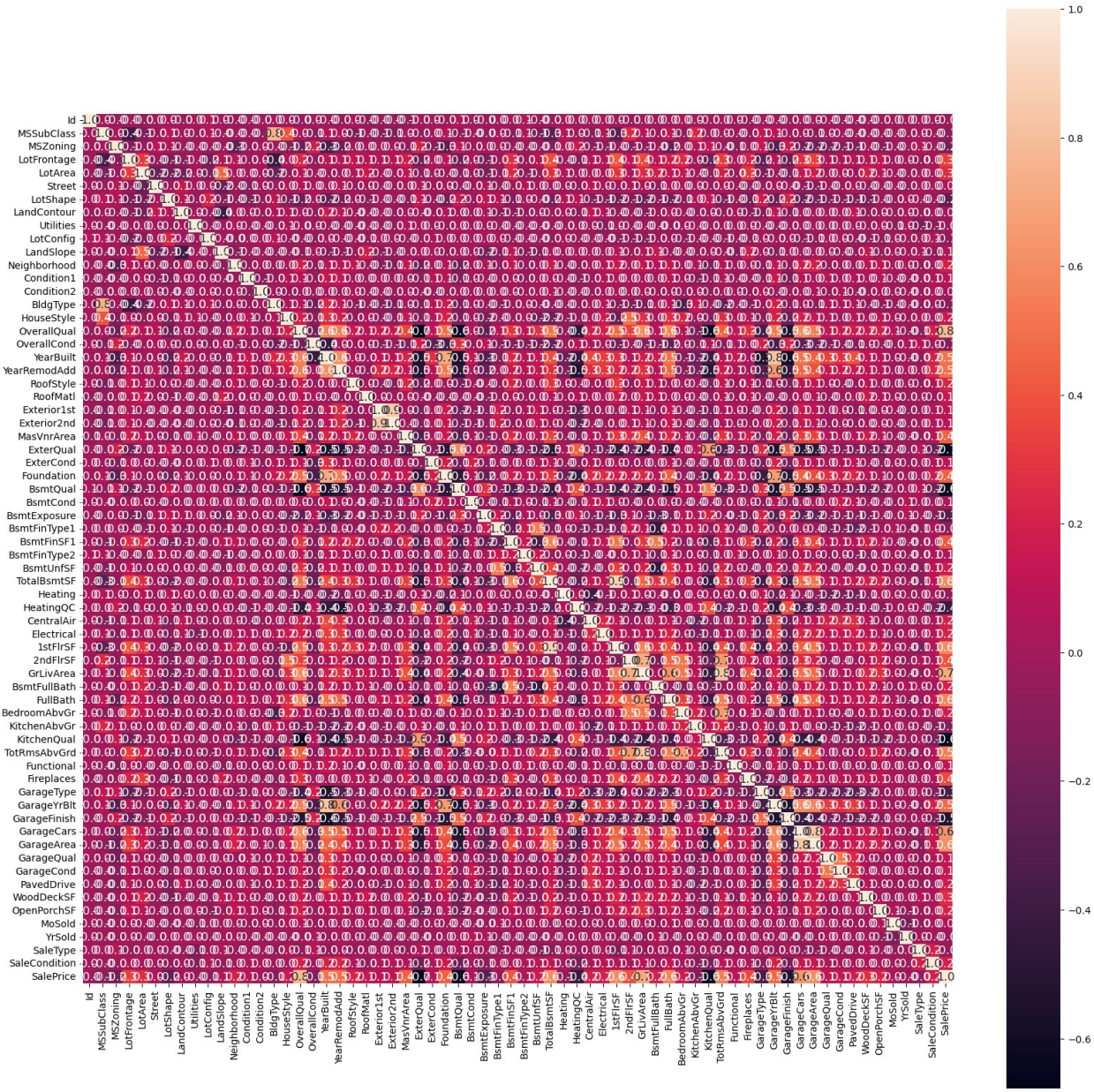
```
                    Id  MSSubClass  MSZoning  LotFrontage
LotArea  \
```

```
Id              1.000000    0.016151 -0.009639    -0.011789 -0.036507

MSSubClass      0.016151    1.000000  0.017811    -0.360201 -0.135995

MSZoning       -0.009639    0.017811  1.000000    -0.088991 -0.023951

LotFrontage    -0.011789   -0.360201 -0.088991     1.000000  0.299898

LotArea        -0.036507   -0.135995 -0.023951     0.299898  1.000000

...                  ...         ...       ...          ...       ...

MoSold          0.023310   -0.015948 -0.019094     0.014158 -0.001991

YrSold          0.000018   -0.021460 -0.022157     0.010726 -0.013415

SaleType        0.013126    0.010772  0.095048    -0.024890  0.014859

SaleCondition   0.001819   -0.010778 -0.008889     0.046891  0.028344

SalePrice      -0.026857   -0.081120 -0.162067     0.329520  0.253854


                 Street  LotShape  LandContour  Utilities
LotConfig  ...  \
Id              0.006110  0.034822    -0.014571   0.013790
0.062671  ...
MSSubClass     -0.020624  0.099092     0.004291  -0.023882
0.064806  ...
MSZoning        0.098863  0.056129    -0.026716  -0.000837  -
0.018868  ...
LotFrontage    -0.034653 -0.132124    -0.071211  -0.000732  -
0.169329  ...
LotArea        -0.214624 -0.163527    -0.168455   0.009667  -
0.122189  ...
...                  ...       ...          ...        ...        ...
...
MoSold         -0.001450 -0.043293    -0.018705  -0.053831
0.015179  ...
YrSold         -0.036469  0.033646     0.018702   0.024490  -
0.006659  ...
SaleType        0.019299 -0.007536    -0.027785  -0.129632
0.008239  ...
SaleCondition   0.011925 -0.030437     0.006286  -0.098010
0.060510  ...
SalePrice       0.042129 -0.248841     0.002661  -0.017098  -
0.063082  ...

                GarageQual  GarageCond  PavedDrive  WoodDeckSF
OpenPorchSF  \
```

```
Id              -0.002595   -0.009759   -0.017654   -0.014014    -
0.003071
MSSubClass       0.003734   -0.008269   -0.011511   -0.027921    -
0.013863
MSZoning        -0.161304   -0.092422   -0.118304   -0.003165    -
0.095421
LotFrontage      0.059072    0.051667    0.075024    0.079092
0.130532
LotArea          0.022035    0.030891   -0.015280    0.198028
0.082442
...                   ...         ...         ...         ...
...
MoSold           0.012376    0.003900   -0.011095    0.003403
0.059998
YrSold           0.040502    0.031819    0.001113    0.018712    -
0.052525
SaleType        -0.031875   -0.016595   -0.047573    0.069541    -
0.015687
SaleCondition    0.057811    0.029204    0.021956    0.039943
0.048593
SalePrice        0.121029    0.147901    0.176629    0.267777
0.219385

                  MoSold    YrSold  SaleType  SaleCondition  SalePrice

Id              0.023310  0.000018  0.013126       0.001819  -0.026857

MSSubClass     -0.015948 -0.021460  0.010772      -0.010778  -0.081120

MSZoning       -0.019094 -0.022157  0.095048      -0.008889  -0.162067

LotFrontage     0.014158  0.010726 -0.024890       0.046891   0.329520

LotArea        -0.001991 -0.013415  0.014859       0.028344   0.253854

...                  ...       ...       ...            ...        ...

MoSold          1.000000 -0.145241 -0.042233       0.031745   0.042637

YrSold         -0.145241  1.000000  0.007127      -0.006552  -0.023478

SaleType       -0.042233  0.007127  1.000000       0.199077  -0.047806

SaleCondition   0.031745 -0.006552  0.199077       1.000000   0.203785

SalePrice       0.042637 -0.023478 -0.047806       0.203785   1.000000


[66 rows x 66 columns]
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# constructing a heatmap to nderstand the correlation
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```




```
G1 = df.iloc[ : , :19]
G1.shape
```

```
(1346, 19)
```

```
G2 = df.iloc[:, 19:38]
G2.shape
```

```
(1346, 19)
```

```
G3 = df.iloc[:, 38:57]
G3.shape
```

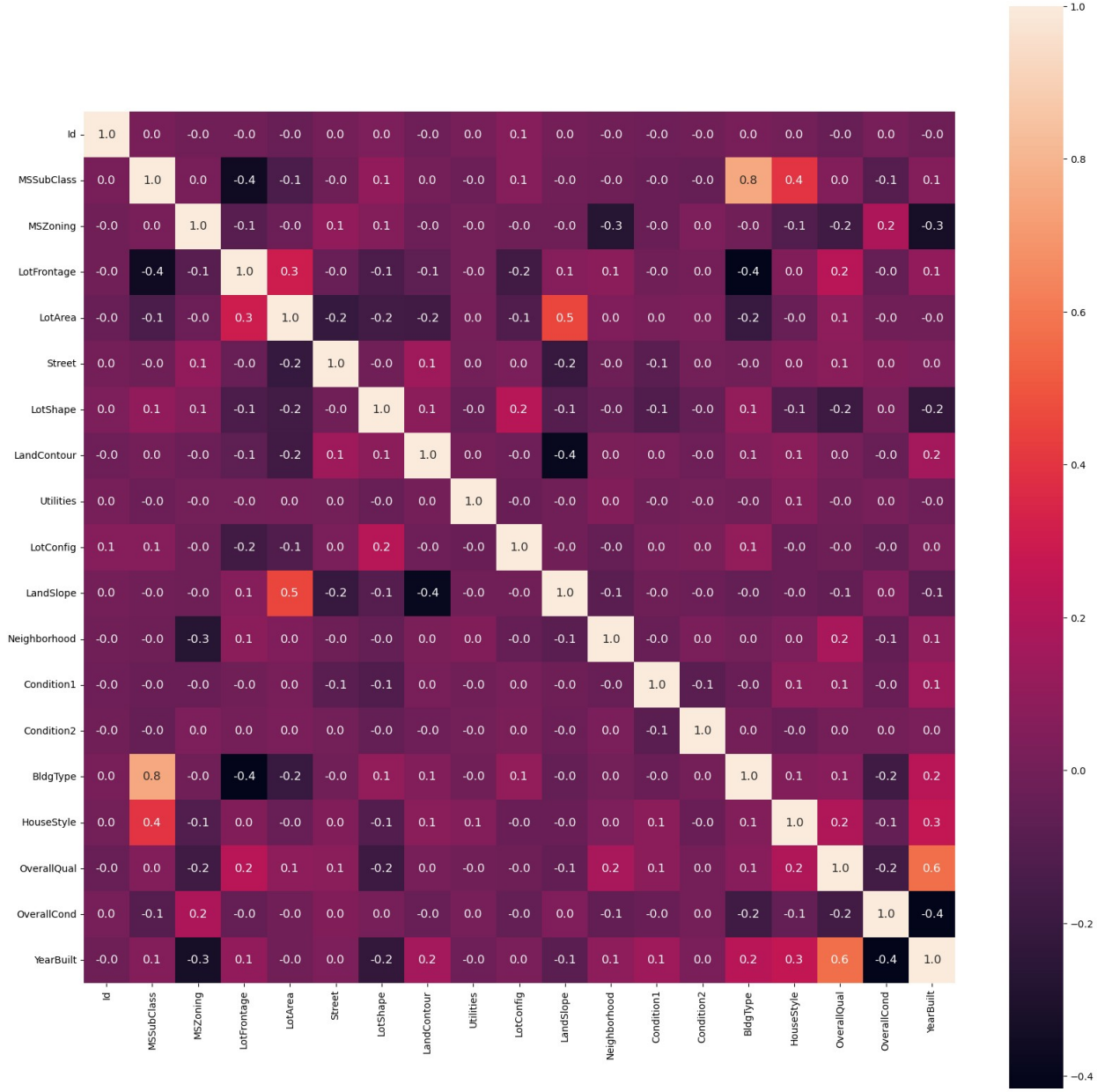
```
(1346, 19)
```

```
G4 = df.iloc[:, 57:66]
G4.shape
```

```
(1346, 9)
```

```
# constructing a heatmap to nderstand the correlation
correlation = G1.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```



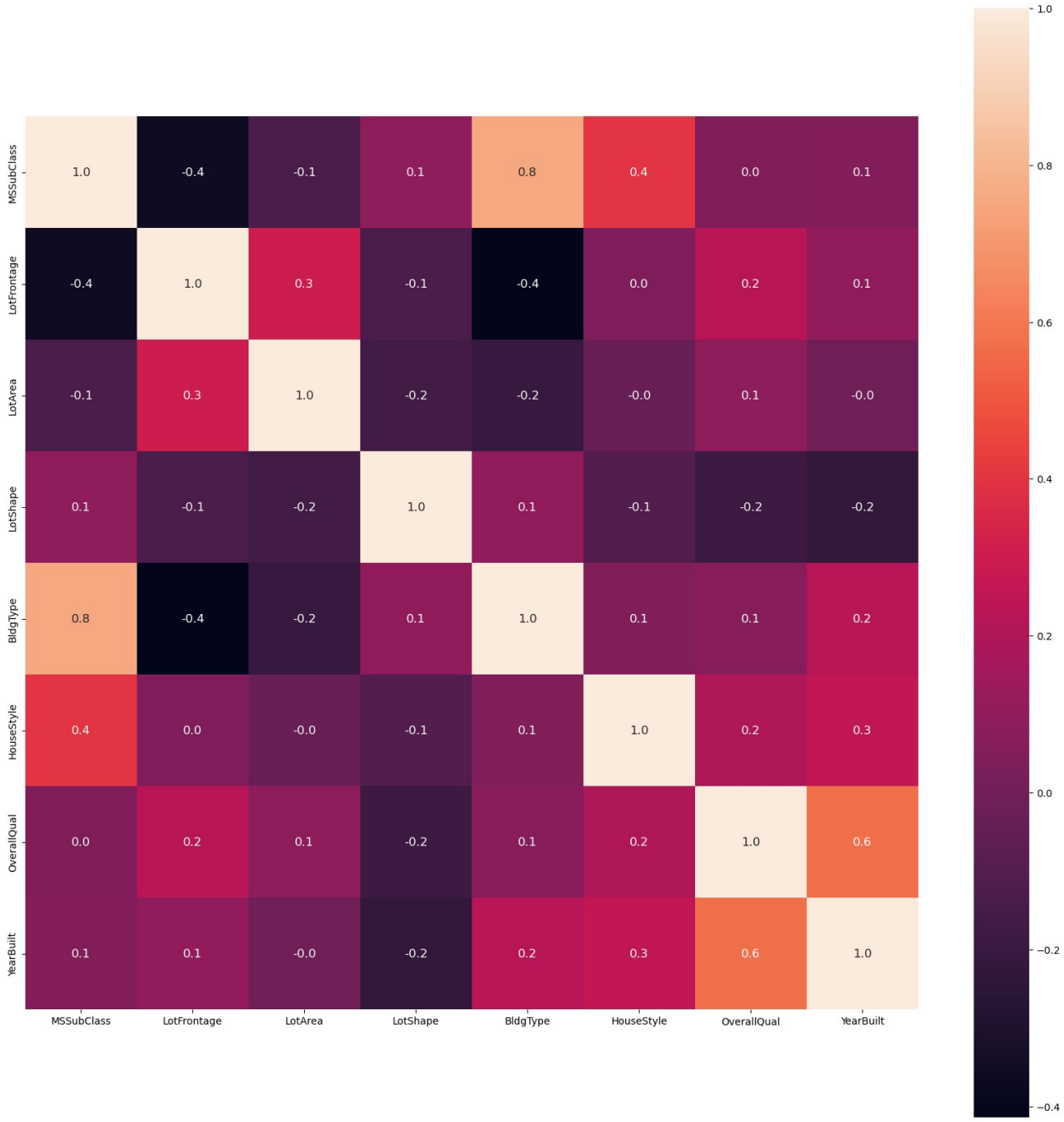

```
# axis = 1 for column drop
G1 =
G1.drop(['OverallCond','Condition2','Condition1','Neighborhood','LandS
lope' , 'Id' ,'MSZoning' , 'Street' , 'LandContour' , 'Utilities' ,
'LotConfig' ] , axis = 1)

G1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
```

```
---  ------       --------------  -----
 0   MSSubClass   1346 non-null   int64
 1   LotFrontage  1346 non-null   float64
 2   LotArea      1346 non-null   int64
 3   LotShape     1346 non-null   int64
 4   BldgType     1346 non-null   int64
 5   HouseStyle   1346 non-null   int64
 6   OverallQual  1346 non-null   int64
 7   YearBuilt    1346 non-null   int64
dtypes: float64(1), int64(7)
memory usage: 94.6 KB
```

```
# constructing a heatmap to nderstand the correlation
correlation = G1.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```




```
# axis = 1 for column drop
df =
df.drop(['OverallCond','Condition2','Condition1','Neighborhood','LandS
lope' , 'Id' ,'MSZoning' , 'Street' , 'LandContour' , 'Utilities' ,
'LotConfig' ] , axis = 1)

df.info()
```

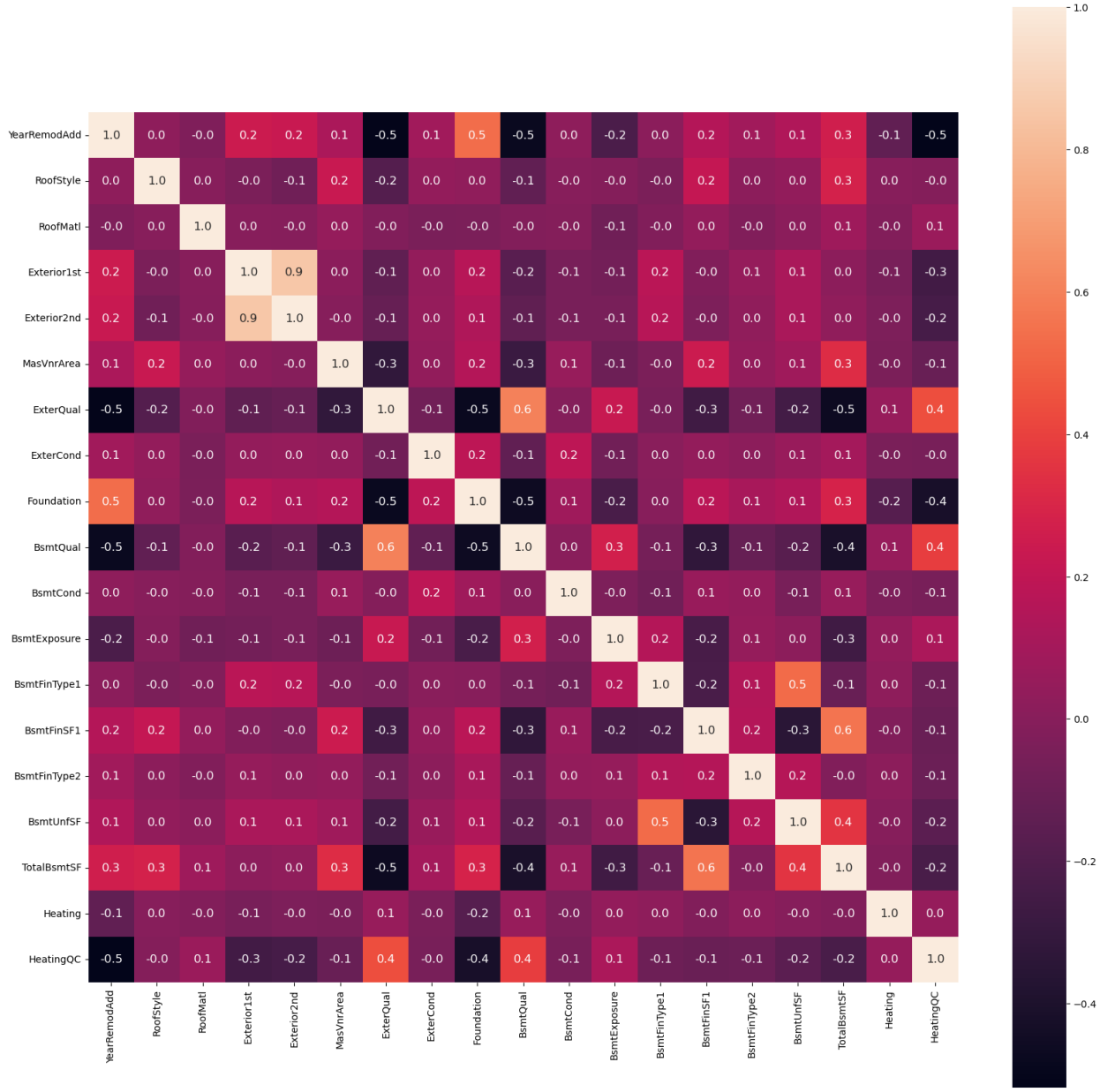
```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
```

```
Data columns (total 55 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   MSSubClass      1346 non-null   int64
 1   LotFrontage     1346 non-null   float64
 2   LotArea         1346 non-null   int64
 3   LotShape        1346 non-null   int64
 4   BldgType        1346 non-null   int64
 5   HouseStyle      1346 non-null   int64
 6   OverallQual     1346 non-null   int64
 7   YearBuilt       1346 non-null   int64
 8   YearRemodAdd    1346 non-null   int64
 9   RoofStyle       1346 non-null   int64
 10  RoofMatl        1346 non-null   int64
 11  Exterior1st     1346 non-null   int64
 12  Exterior2nd     1346 non-null   int64
 13  MasVnrArea      1346 non-null   float64
 14  ExterQual       1346 non-null   int64
 15  ExterCond       1346 non-null   int64
 16  Foundation      1346 non-null   int64
 17  BsmtQual        1346 non-null   int64
 18  BsmtCond        1346 non-null   int64
 19  BsmtExposure    1346 non-null   int64
 20  BsmtFinType1    1346 non-null   int64
 21  BsmtFinSF1      1346 non-null   float64
 22  BsmtFinType2    1346 non-null   int64
 23  BsmtUnfSF       1346 non-null   float64
 24  TotalBsmtSF     1346 non-null   int64
 25  Heating         1346 non-null   int64
 26  HeatingQC       1346 non-null   int64
 27  CentralAir      1346 non-null   int64
 28  Electrical      1346 non-null   int64
 29  1stFlrSF        1346 non-null   int64
 30  2ndFlrSF        1346 non-null   float64
 31  GrLivArea       1346 non-null   int64
 32  BsmtFullBath    1346 non-null   float64
 33  FullBath        1346 non-null   float64
 34  BedroomAbvGr    1346 non-null   float64
 35  KitchenAbvGr    1346 non-null   int64
 36  KitchenQual     1346 non-null   int64
 37  TotRmsAbvGrd    1346 non-null   int64
 38  Functional      1346 non-null   int64
 39  Fireplaces      1346 non-null   float64
 40  GarageType      1346 non-null   int64
 41  GarageYrBlt     1346 non-null   float64
 42  GarageFinish    1346 non-null   int64
 43  GarageCars      1346 non-null   int64
 44  GarageArea      1346 non-null   int64
 45  GarageQual      1346 non-null   int64
```

```
 46  GarageCond     1346 non-null   int64
 47  PavedDrive     1346 non-null   int64
 48  WoodDeckSF     1346 non-null   float64
 49  OpenPorchSF    1346 non-null   float64
 50  MoSold         1346 non-null   int64
 51  YrSold         1346 non-null   int64
 52  SaleType       1346 non-null   int64
 53  SaleCondition  1346 non-null   int64
 54  SalePrice      1346 non-null   int64
dtypes: float64(12), int64(43)
memory usage: 588.9 KB
```

```
# constructing a heatmap to nderstand the correlation
correlation = G2.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```



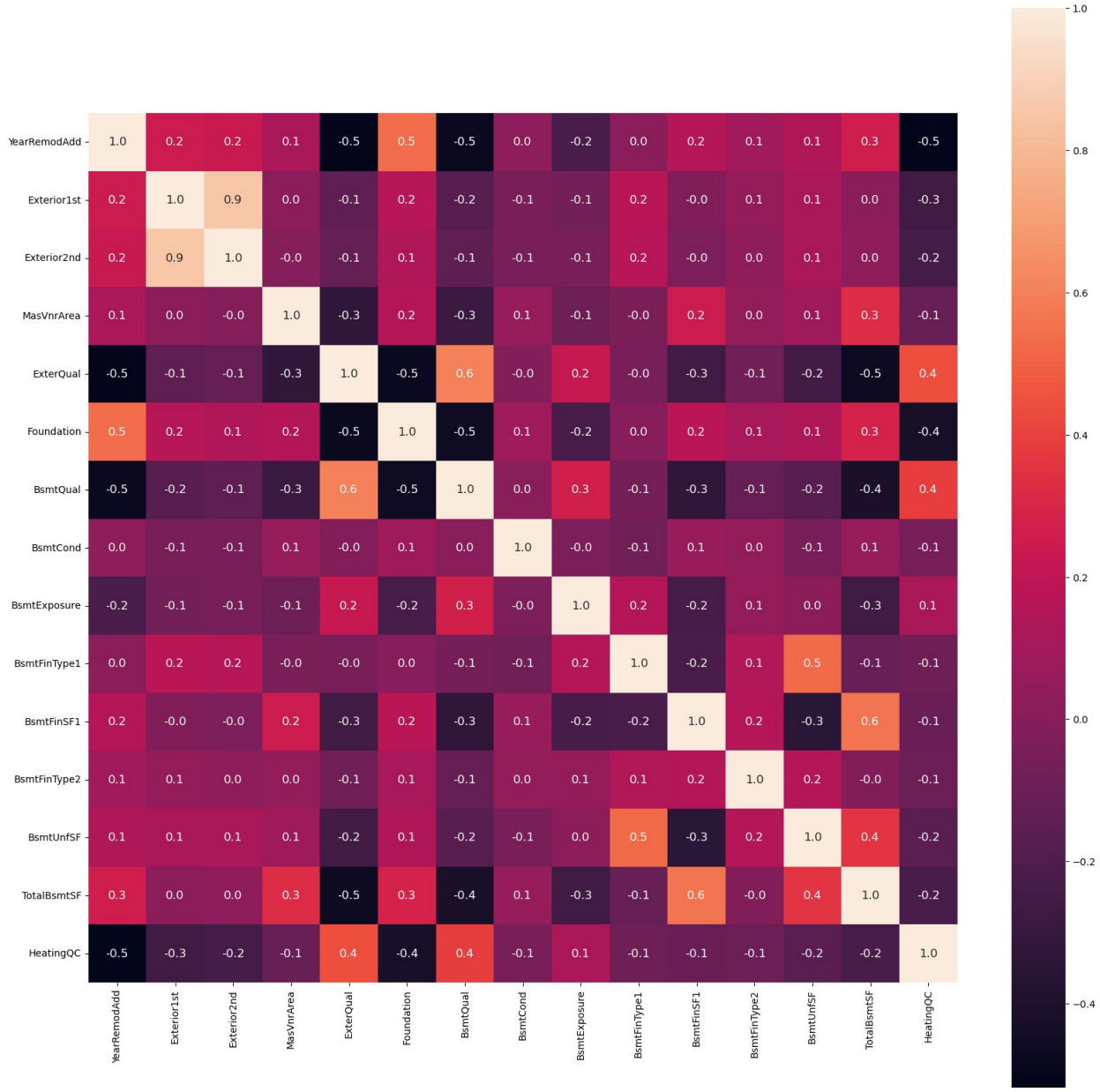

```
# axis = 1 for column drop
G2 = G2.drop([ 'Heating', 'ExterCond','RoofMatl' ,'RoofStyle'] , axis
= 1)

G2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   YearRemodAdd  1346 non-null   int64
 1   Exterior1st   1346 non-null   int64
```

```
 2   Exterior2nd   1346 non-null   int64
 3   MasVnrArea    1346 non-null   float64
 4   ExterQual     1346 non-null   int64
 5   Foundation    1346 non-null   int64
 6   BsmtQual      1346 non-null   int64
 7   BsmtCond      1346 non-null   int64
 8   BsmtExposure  1346 non-null   int64
 9   BsmtFinType1  1346 non-null   int64
 10  BsmtFinSF1    1346 non-null   float64
 11  BsmtFinType2  1346 non-null   int64
 12  BsmtUnfSF     1346 non-null   float64
 13  TotalBsmtSF   1346 non-null   int64
 14  HeatingQC     1346 non-null   int64
dtypes: float64(3), int64(12)
memory usage: 168.2 KB
```

```
# axis = 1 for column drop
df = df.drop([ 'Heating', 'ExterCond','RoofMatl' ,'RoofStyle'] , axis
= 1)

# constructing a heatmap to nderstand the correlation
correlation = G2.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```



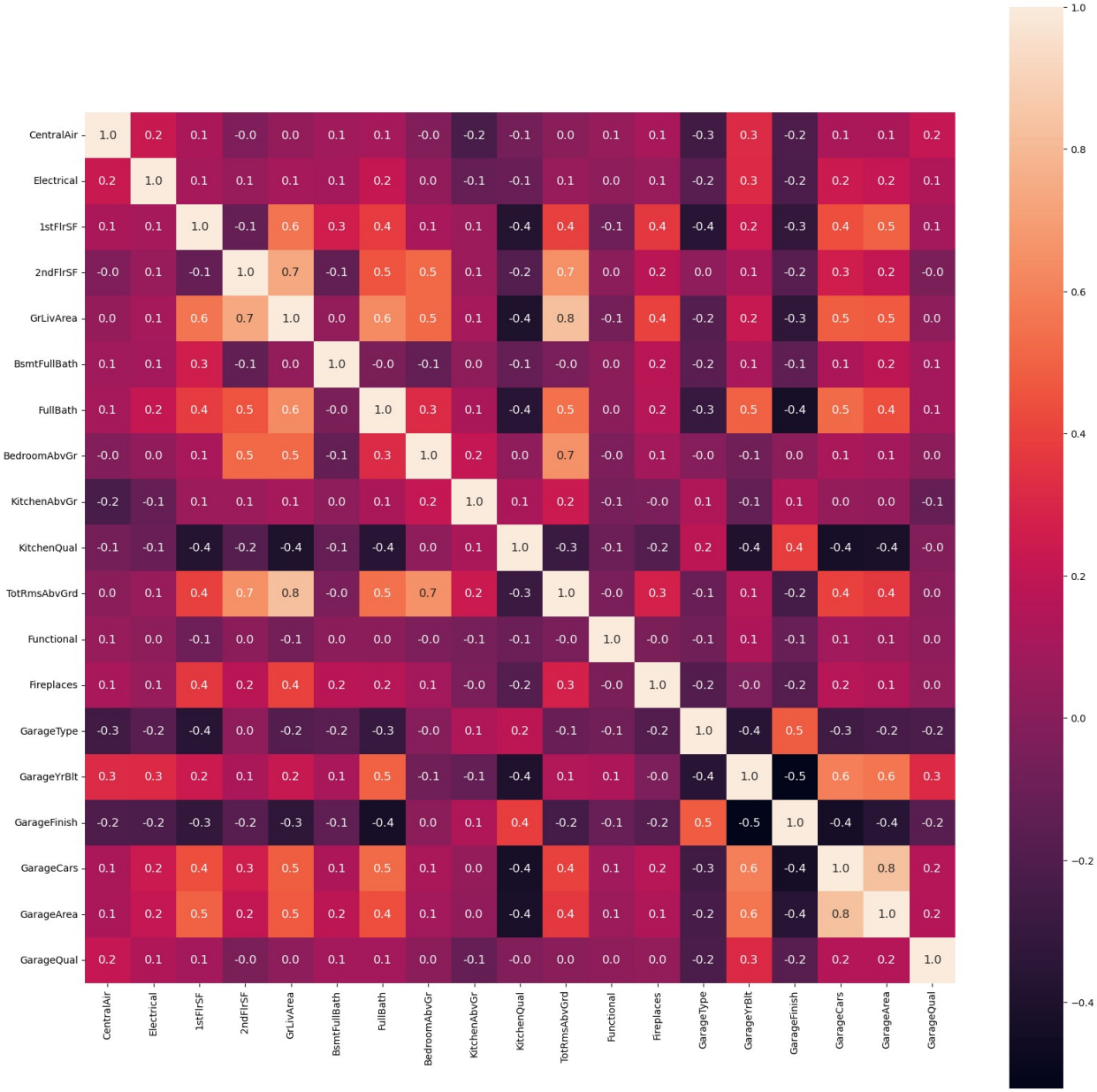

```
G2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   YearRemodAdd   1346 non-null   int64  
 1   Exterior1st    1346 non-null   int64  
 2   Exterior2nd    1346 non-null   int64  
 3   MasVnrArea     1346 non-null   float64
 4   ExterQual      1346 non-null   int64  
```

```
 5   Foundation    1346 non-null   int64
 6   BsmtQual      1346 non-null   int64
 7   BsmtCond      1346 non-null   int64
 8   BsmtExposure  1346 non-null   int64
 9   BsmtFinType1  1346 non-null   int64
 10  BsmtFinSF1    1346 non-null   float64
 11  BsmtFinType2  1346 non-null   int64
 12  BsmtUnfSF     1346 non-null   float64
 13  TotalBsmtSF   1346 non-null   int64
 14  HeatingQC     1346 non-null   int64
dtypes: float64(3), int64(12)
memory usage: 168.2 KB
```

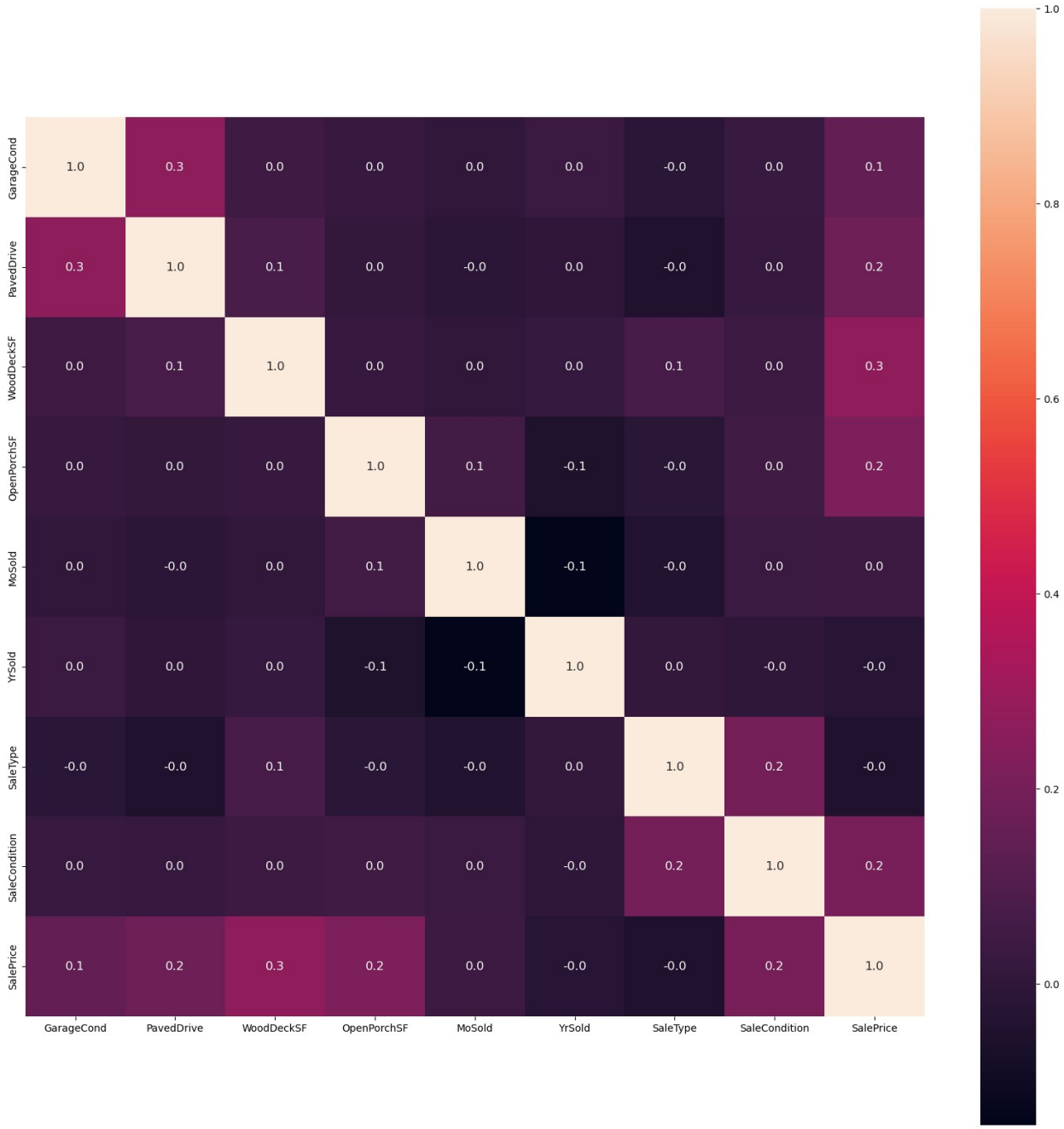
```
# constructing a heatmap to nderstand the correlation
correlation = G3.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```




```
# constructing a heatmap to nderstand the correlation
correlation = G4.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```



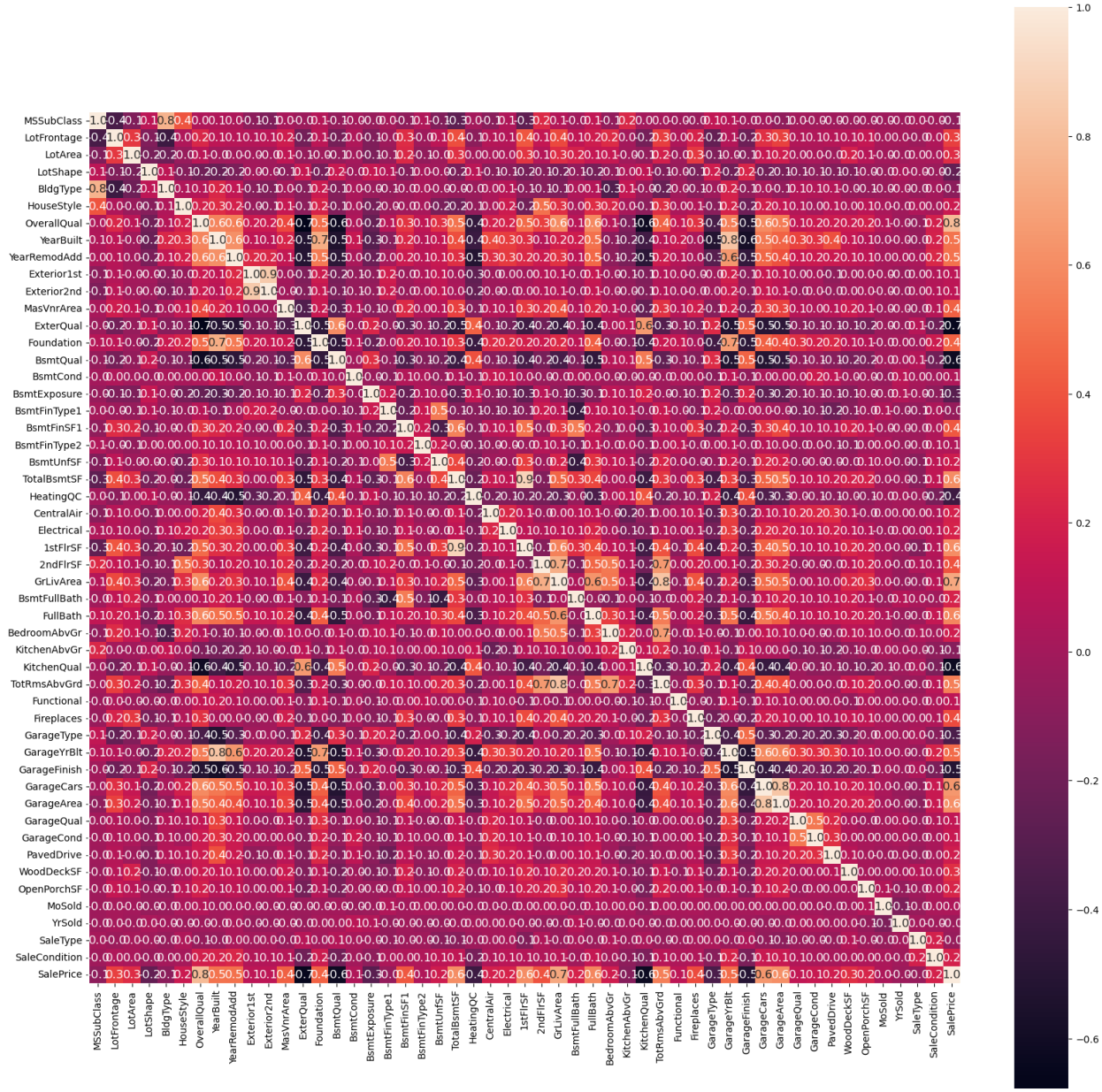

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 51 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   MSSubClass      1346 non-null   int64
 1   LotFrontage     1346 non-null   float64
 2   LotArea         1346 non-null   int64
```

```
3   LotShape        1346 non-null   int64
4   BldgType        1346 non-null   int64
5   HouseStyle      1346 non-null   int64
6   OverallQual     1346 non-null   int64
7   YearBuilt       1346 non-null   int64
8   YearRemodAdd    1346 non-null   int64
9   Exterior1st     1346 non-null   int64
10  Exterior2nd     1346 non-null   int64
11  MasVnrArea      1346 non-null   float64
12  ExterQual       1346 non-null   int64
13  Foundation      1346 non-null   int64
14  BsmtQual        1346 non-null   int64
15  BsmtCond        1346 non-null   int64
16  BsmtExposure    1346 non-null   int64
17  BsmtFinType1    1346 non-null   int64
18  BsmtFinSF1      1346 non-null   float64
19  BsmtFinType2    1346 non-null   int64
20  BsmtUnfSF       1346 non-null   float64
21  TotalBsmtSF     1346 non-null   int64
22  HeatingQC       1346 non-null   int64
23  CentralAir      1346 non-null   int64
24  Electrical      1346 non-null   int64
25  1stFlrSF        1346 non-null   int64
26  2ndFlrSF        1346 non-null   float64
27  GrLivArea       1346 non-null   int64
28  BsmtFullBath    1346 non-null   float64
29  FullBath        1346 non-null   float64
30  BedroomAbvGr    1346 non-null   float64
31  KitchenAbvGr    1346 non-null   int64
32  KitchenQual     1346 non-null   int64
33  TotRmsAbvGrd    1346 non-null   int64
34  Functional      1346 non-null   int64
35  Fireplaces      1346 non-null   float64
36  GarageType      1346 non-null   int64
37  GarageYrBlt     1346 non-null   float64
38  GarageFinish    1346 non-null   int64
39  GarageCars      1346 non-null   int64
40  GarageArea      1346 non-null   int64
41  GarageQual      1346 non-null   int64
42  GarageCond      1346 non-null   int64
43  PavedDrive      1346 non-null   int64
44  WoodDeckSF      1346 non-null   float64
45  OpenPorchSF     1346 non-null   float64
46  MoSold          1346 non-null   int64
47  YrSold          1346 non-null   int64
48  SaleType        1346 non-null   int64
49  SaleCondition   1346 non-null   int64
50  SalePrice       1346 non-null   int64
```

```
dtypes: float64(12), int64(39)
memory usage: 546.8 KB
```

```
# constructing a heatmap to nderstand the correlation
correlation = df.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```



```
G1 = df.iloc[ : , :20]
G1.shape
```

```
(1346, 20)
```

```
G2 = df.iloc[ : ,20 :40]
G2.shape
```

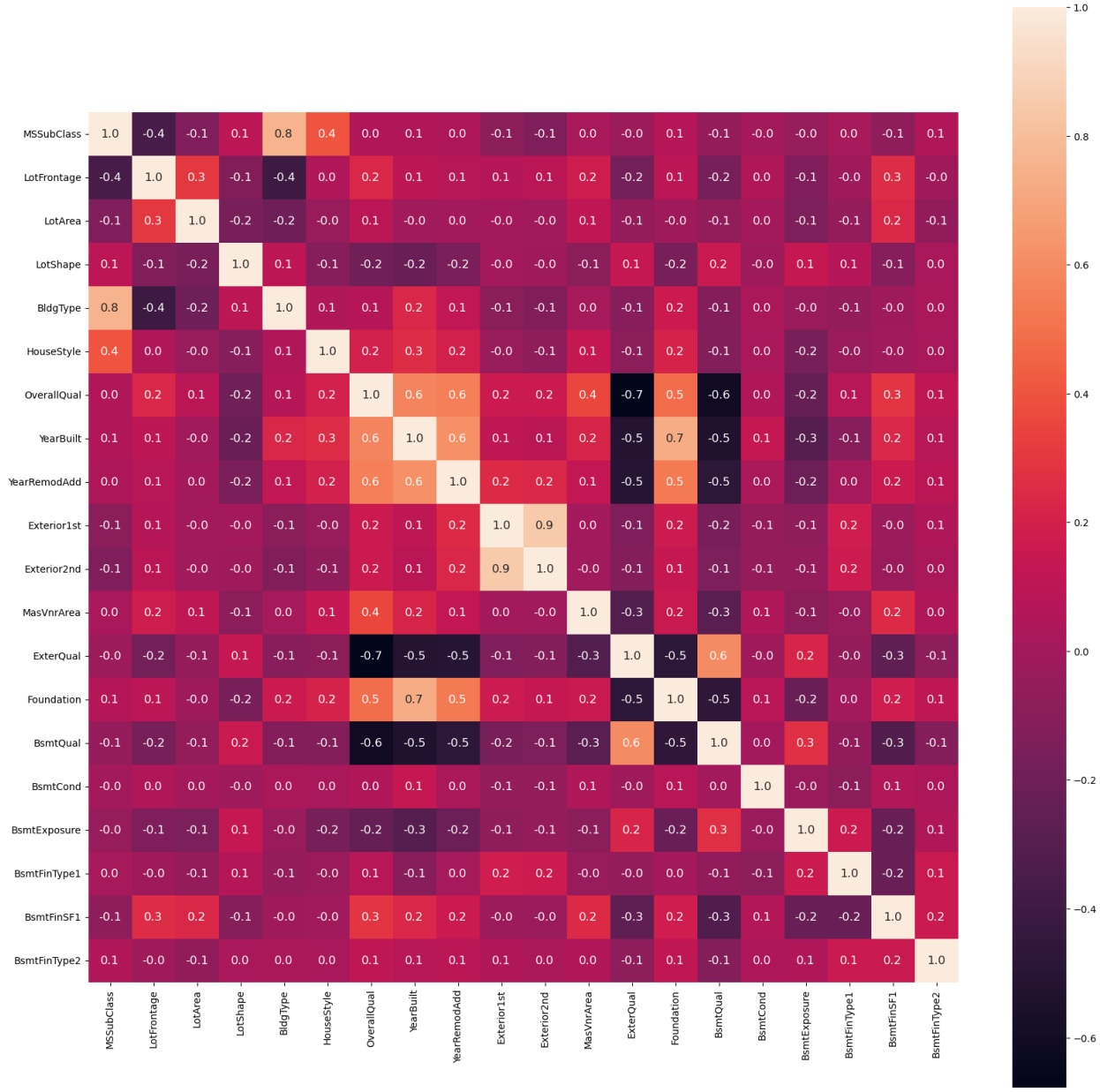
```
(1346, 20)
```

```
G3 = df.iloc[ : ,40 :51]
G3.shape
```

```
(1346, 11)
```

```
# constructing a heatmap to nderstand the correlation
correlation = G1.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```

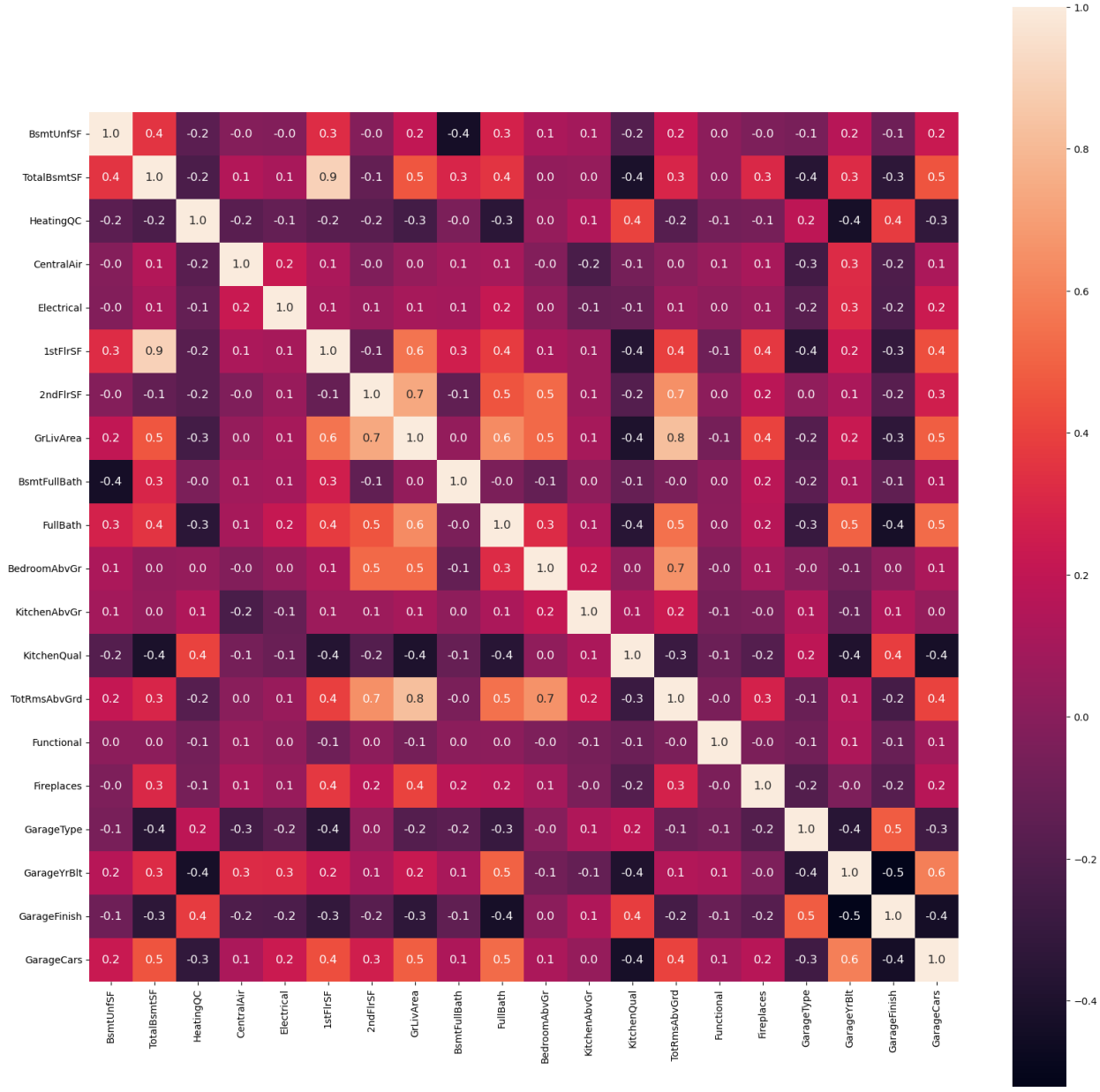



```
G1.info()

<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MSSubClass     1346 non-null   int64  
 1   LotFrontage    1346 non-null   float64
 2   LotArea        1346 non-null   int64  
 3   LotShape       1346 non-null   int64  
 4   BldgType       1346 non-null   int64  
```

```
 5   HouseStyle    1346 non-null   int64
 6   OverallQual   1346 non-null   int64
 7   YearBuilt     1346 non-null   int64
 8   YearRemodAdd  1346 non-null   int64
 9   Exterior1st   1346 non-null   int64
 10  Exterior2nd   1346 non-null   int64
 11  MasVnrArea    1346 non-null   float64
 12  ExterQual     1346 non-null   int64
 13  Foundation    1346 non-null   int64
 14  BsmtQual      1346 non-null   int64
 15  BsmtCond      1346 non-null   int64
 16  BsmtExposure  1346 non-null   int64
 17  BsmtFinType1  1346 non-null   int64
 18  BsmtFinSF1    1346 non-null   float64
 19  BsmtFinType2  1346 non-null   int64
dtypes: float64(3), int64(17)
memory usage: 220.8 KB
```

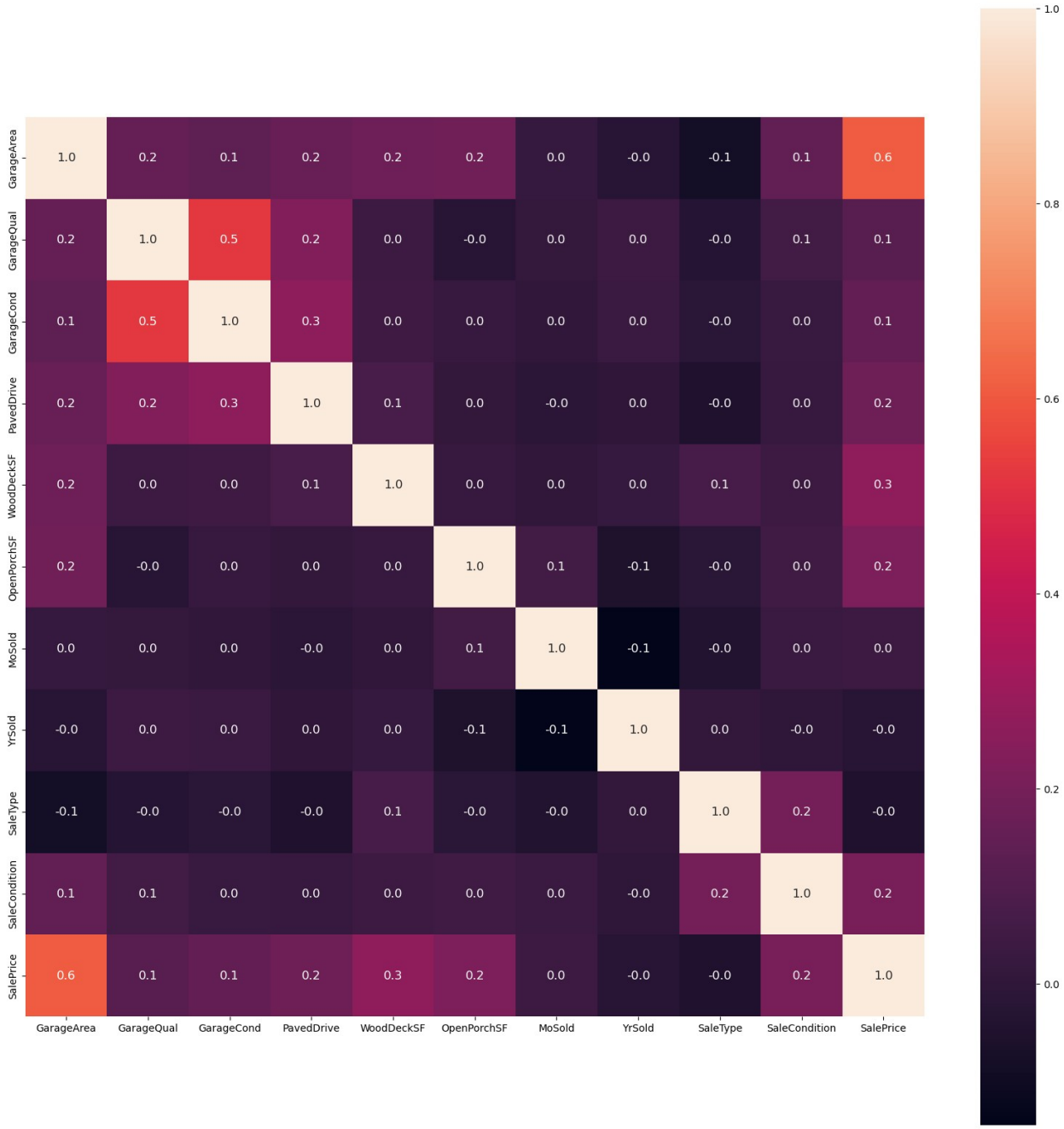
```
# G1 = G1.drop('BsmtCond' , axis=1)
df = df.drop('BsmtCond' , axis=1)


# constructing a heatmap to nderstand the correlation
correlation = G2.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```




```
G2 = G2.drop('Functional' , axis = 1)

df = df.drop('Functional' , axis = 1)

# G3 = df.iloc[ : ,30 :50]
correlation = G3.corr()
plt.figure(figsize=(20,20))
sns.heatmap(correlation, square=True, fmt='.1f', annot=True,
annot_kws={'size':12})
```

```
<Axes: >
```




```
G3 = G3.drop(['SaleCondition','SaleType', 'YrSold', 'MoSold'] , axis = 1)

df = df.drop(['SaleCondition','SaleType', 'YrSold', 'MoSold'] , axis = 1)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1346 entries, 0 to 1459
```

```
Data columns (total 45 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MSSubClass     1346 non-null   int64  
 1   LotFrontage    1346 non-null   float64
 2   LotArea        1346 non-null   int64  
 3   LotShape       1346 non-null   int64  
 4   BldgType       1346 non-null   int64  
 5   HouseStyle     1346 non-null   int64  
 6   OverallQual    1346 non-null   int64  
 7   YearBuilt      1346 non-null   int64  
 8   YearRemodAdd   1346 non-null   int64  
 9   Exterior1st    1346 non-null   int64  
 10  Exterior2nd    1346 non-null   int64  
 11  MasVnrArea     1346 non-null   float64
 12  ExterQual      1346 non-null   int64  
 13  Foundation     1346 non-null   int64  
 14  BsmtQual       1346 non-null   int64  
 15  BsmtExposure   1346 non-null   int64  
 16  BsmtFinType1   1346 non-null   int64  
 17  BsmtFinSF1     1346 non-null   float64
 18  BsmtFinType2   1346 non-null   int64  
 19  BsmtUnfSF      1346 non-null   float64
 20  TotalBsmtSF    1346 non-null   int64  
 21  HeatingQC      1346 non-null   int64  
 22  CentralAir     1346 non-null   int64  
 23  Electrical     1346 non-null   int64  
 24  1stFlrSF       1346 non-null   int64  
 25  2ndFlrSF       1346 non-null   float64
 26  GrLivArea      1346 non-null   int64  
 27  BsmtFullBath   1346 non-null   float64
 28  FullBath       1346 non-null   float64
 29  BedroomAbvGr   1346 non-null   float64
 30  KitchenAbvGr   1346 non-null   int64  
 31  KitchenQual    1346 non-null   int64  
 32  TotRmsAbvGrd   1346 non-null   int64  
 33  Fireplaces     1346 non-null   float64
 34  GarageType     1346 non-null   int64  
 35  GarageYrBlt    1346 non-null   float64
 36  GarageFinish   1346 non-null   int64  
 37  GarageCars     1346 non-null   int64  
 38  GarageArea     1346 non-null   int64  
 39  GarageQual     1346 non-null   int64  
 40  GarageCond     1346 non-null   int64  
 41  PavedDrive     1346 non-null   int64  
 42  WoodDeckSF     1346 non-null   float64
 43  OpenPorchSF    1346 non-null   float64
 44  SalePrice      1346 non-null   int64  
```

```
dtypes: float64(12), int64(33)
memory usage: 483.7 KB
```

```
df['BsmtQual'].value_counts()
```

```
BsmtQual
2    598
3    595
0    121
1     32
Name: count, dtype: int64
```

```
df = df[df['BsmtQual']!=1]
```

```
df.shape
```

```
(1314, 45)
```

Predict type of HeatingQC

Data Spliting

```
X=df.drop("BsmtQual", axis = 1)
Y = df["BsmtQual"]
```

```
X.info()
X.shape
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1314 entries, 0 to 1459
Data columns (total 44 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   MSSubClass    1314 non-null   int64
 1   LotFrontage   1314 non-null   float64
 2   LotArea       1314 non-null   int64
 3   LotShape      1314 non-null   int64
 4   BldgType      1314 non-null   int64
 5   HouseStyle    1314 non-null   int64
 6   OverallQual   1314 non-null   int64
 7   YearBuilt     1314 non-null   int64
 8   YearRemodAdd  1314 non-null   int64
 9   Exterior1st   1314 non-null   int64
 10  Exterior2nd   1314 non-null   int64
 11  MasVnrArea    1314 non-null   float64
```

```
 12  ExterQual     1314 non-null   int64
 13  Foundation    1314 non-null   int64
 14  BsmtExposure  1314 non-null   int64
 15  BsmtFinType1  1314 non-null   int64
 16  BsmtFinSF1    1314 non-null   float64
 17  BsmtFinType2  1314 non-null   int64
 18  BsmtUnfSF     1314 non-null   float64
 19  TotalBsmtSF   1314 non-null   int64
 20  HeatingQC     1314 non-null   int64
 21  CentralAir    1314 non-null   int64
 22  Electrical    1314 non-null   int64
 23  1stFlrSF      1314 non-null   int64
 24  2ndFlrSF      1314 non-null   float64
 25  GrLivArea     1314 non-null   int64
 26  BsmtFullBath  1314 non-null   float64
 27  FullBath      1314 non-null   float64
 28  BedroomAbvGr  1314 non-null   float64
 29  KitchenAbvGr  1314 non-null   int64
 30  KitchenQual   1314 non-null   int64
 31  TotRmsAbvGrd  1314 non-null   int64
 32  Fireplaces    1314 non-null   float64
 33  GarageType    1314 non-null   int64
 34  GarageYrBlt   1314 non-null   float64
 35  GarageFinish  1314 non-null   int64
 36  GarageCars    1314 non-null   int64
 37  GarageArea    1314 non-null   int64
 38  GarageQual    1314 non-null   int64
 39  GarageCond    1314 non-null   int64
 40  PavedDrive    1314 non-null   int64
 41  WoodDeckSF    1314 non-null   float64
 42  OpenPorchSF   1314 non-null   float64
 43  SalePrice     1314 non-null   int64
dtypes: float64(12), int64(32)
memory usage: 462.0 KB

(1314, 44)

Y.info()
Y.shape

<class 'pandas.core.series.Series'>
Index: 1314 entries, 0 to 1459
Series name: BsmtQual
Non-Null Count  Dtype
--------------  -----
1314 non-null   int64
dtypes: int64(1)
memory usage: 20.5 KB

(1314,)
```

```
from sklearn.model_selection  import train_test_split
X_TRAIN , X_TEST , Y_TRAIN, Y_TEST = train_test_split(X,Y, test_size =
0.25, random_state=30)
print("Size of Train X =  " , len(X_TRAIN))
print("Size of Train Y =  " , len(Y_TRAIN))
print("Size of Test X =  " , len(X_TEST))
print("Size of Test Y =  " , len(Y_TEST))
```

```
Size of Train X =   985
Size of Train Y =   985
Size of Test X =   329
Size of Test Y =   329
```

GaussianNB Base Classifier

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
# fit the regressor with X and Y data
gnb.fit(X_TRAIN, Y_TRAIN)
```

```
GaussianNB()
```

```
# accuracy for prediction on training data
training_data_prediction = gnb.predict(X_TRAIN)
print(training_data_prediction)
```

```
[3 0 2 2 2 2 3 3 3 2 3 2 2 3 2 3 3 2 0 3 2 3 0 3 2 2 2 3 3 3 3 2 3 3 3
3 3
 3 3 0 3 2 3 0 0 2 2 2 3 3 0 3 0 3 0 3 3 2 3 0 0 3 3 2 2 3 3 3 3 2 3 3
2 0
 3 2 2 3 3 2 2 3 3 0 3 3 3 2 3 2 0 2 2 3 3 3 3 3 3 3 2 3 2 3 3 3 2 3 3
2 3
 3 2 3 2 3 2 3 3 3 2 2 2 2 0 3 2 3 2 0 2 3 3 3 0 0 3 3 2 3 3 2 2 3 2 3
3 3
 3 0 3 3 3 3 3 2 2 3 2 0 2 2 3 3 3 3 0 0 3 2 2 3 3 3 3 3 3 2 2 0 0 2 3
0 3
 3 0 2 3 2 3 3 2 2 3 2 3 3 2 2 2 3 2 2 0 3 3 3 3 3 2 2 0 3 0 3 0 2 3 3
3 2
 0 3 3 3 0 3 2 0 2 3 2 2 3 2 2 2 2 3 2 0 3 2 3 0 3 3 0 2 2 3 3 3 3 0 2
3 2
 0 3 2 3 2 2 3 3 2 2 3 2 2 2 3 0 3 2 2 3 2 2 2 0 3 3 3 3 2 3 2 3 2 0 3
3 2
 0 2 2 2 3 2 2 0 3 2 3 0 3 3 2 3 3 3 3 3 2 2 2 2 3 2 3 3 0 3 2 3 2 3 3
3 2
 0 2 2 2 2 3 3 3 3 3 2 0 0 2 3 3 3 3 2 3 0 0 2 2 3 2 3 2 0 2 2 0 2 3 2
3 3
 0 2 0 2 2 2 3 3 3 0 3 2 2 3 3 3 3 2 3 2 3 0 2 3 3 2 2 2 2 2 3 2 2 2 0
3 2
```

```
 2 0 2 0 3 2 3 3 3 2 2 3 2 3 0 3 2 0 3 3 3 3 3 2 0 3 3 0 0 2 3 3 2 3 2
2 3
 3 3 0 3 2 3 3 3 2 2 2 2 3 3 2 2 2 3 2 0 3 3 2 3 3 2 2 3 0 0 2 2 3 2 3
3 0
 3 3 3 2 2 2 2 3 2 2 2 3 2 2 3 3 3 3 0 3 0 3 2 3 3 3 3 3 2 3 3 2 2 3 3
3 3
 0 3 3 3 2 2 3 3 3 3 2 3 3 0 0 3 2 0 2 3 0 0 0 3 0 0 3 2 0 2 3 3 0 3 2
2 3
 2 0 2 3 3 2 2 3 2 3 2 2 3 2 2 3 2 0 3 2 3 0 2 3 3 3 3 2 3 3 0 3 2 3 3
0 3
 2 0 3 2 3 3 2 2 2 3 0 3 0 2 2 3 2 3 3 3 0 2 3 3 3 3 3 3 3 2 3 3 2 2 2
2 2
 3 2 0 2 3 3 3 2 2 2 0 2 3 3 2 3 2 3 3 2 3 3 2 2 2 0 3 2 3 3 2 3 2 0 3
3 2
 3 3 2 2 2 3 3 3 2 3 0 3 3 0 3 0 2 3 3 3 2 3 2 2 3 2 3 3 2 2 3 2 2 3 2
2 3
 3 3 2 3 3 3 0 3 3 2 2 3 0 3 3 3 3 0 2 0 2 3 2 3 3 2 2 2 3 3 3 2 2 3 2
2 3
 3 3 3 3 3 3 2 3 2 3 3 2 3 0 3 3 2 3 0 0 3 2 3 3 3 3 3 3 2 2 3 3 0 3 2
3 3
 3 2 3 2 2 0 0 3 3 3 2 3 3 0 2 2 2 2 3 0 3 3 3 3 3 2 3 3 3 3 2 2 3 2 3
3 3
 3 0 2 2 2 2 3 0 3 3 3 2 2 2 2 3 3 3 2 0 3 2 2 3 2 0 0 3 2 3 3 2 2 0 3
3 2
 3 3 3 2 3 0 2 3 2 3 3 2 2 3 3 3 2 3 2 3 2 2 2 0 2 3 3 3 2 3 0 2 2 3 3
3 3
 2 2 3 3 2 2 2 3 3 2 3 2 0 2 2 3 3 0 2 2 3 3 3 3 2 2 0 2 2 3 3 2 3 2 3
2 2
 2 3 3 3 2 2 3 2 2 2 2 3 3 2 3 0 0 2 0 3 0 2 3 3 3 2 3 0 0 0 3 2 2 3 0
2 3
 2 3 2 3 0 2 0 2 2 3 2 0 0 3 3 0 2 3 2 3 3 3 2]
```

Prediction on Train Data

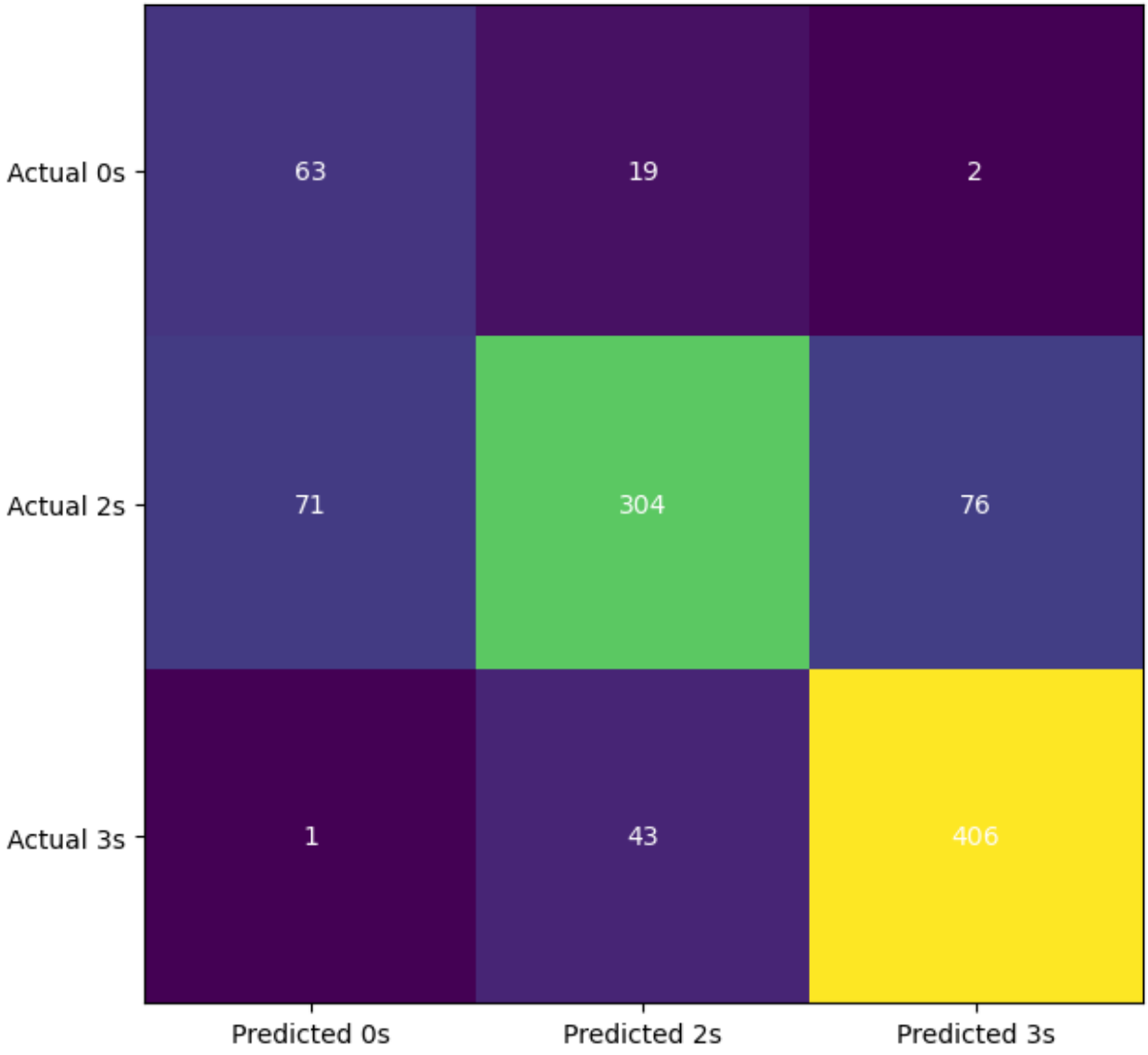
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TRAIN, training_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

```
The Accuracy of Prediction is  0.7847715736040609
```

```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(Y_TRAIN, training_data_prediction)
```

```
array([[ 63,  19,   2],
       [ 71, 304,  76],
       [  1,  43, 406]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TRAIN, training_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```



```
report = classification_report(Y_TRAIN, training_data_prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.47      0.75      0.58        84
           2       0.83      0.67      0.74       451
           3       0.84      0.90      0.87       450

    accuracy                           0.78       985
   macro avg       0.71      0.78      0.73       985
weighted avg       0.80      0.78      0.79       985
```

Testing

```
# accuracy for prediction on training data
testing_data_prediction = gnb.predict(X_TEST)
print(testing_data_prediction)
```

```
[3 0 2 3 2 3 2 3 2 3 2 2 3 0 2 3 2 3 0 3 3 3 3 0 2 3 2 2 0 3 0 2 2 2 2
0 0
 3 0 0 3 3 3 2 3 2 2 3 3 3 3 3 0 3 3 3 3 0 2 0 3 3 2 0 3 3 3 2 3 3 3 2
3 2
 3 2 0 3 2 2 3 3 3 3 2 3 2 2 2 0 2 0 3 2 3 3 3 3 2 3 2 2 3 2 3 2 3 2 3
0 3
 2 3 2 0 3 3 0 3 2 0 2 3 3 2 2 2 3 3 2 3 2 3 0 2 3 3 3 3 2 2 3 3 0 3 3
0 2
 2 2 2 3 0 2 3 2 0 0 3 3 3 2 2 3 2 3 2 3 0 3 2 3 3 3 2 0 2 3 3 2 0 2 3
3 3
 3 3 2 3 2 0 2 3 3 3 2 2 0 3 3 3 3 2 3 2 2 0 3 2 3 3 3 3 3 3 2 2 0 3 3
3 3
 2 3 2 2 2 3 0 2 3 3 0 2 3 3 3 3 3 3 3 2 3 2 2 2 0 3 0 2 0 0 3 2 2 2 2
2 3
 3 2 2 3 2 3 2 3 3 0 3 3 0 2 0 2 2 0 2 3 2 3 3 2 2 2 3 2 3 3 3 2 3 3 3
3 2
 2 3 0 2 2 2 2 2 3 2 3 2 2 2 3 0 2 3 2 2 3 2 3 2 3 2 3 3 3 0 3 3 2]
```

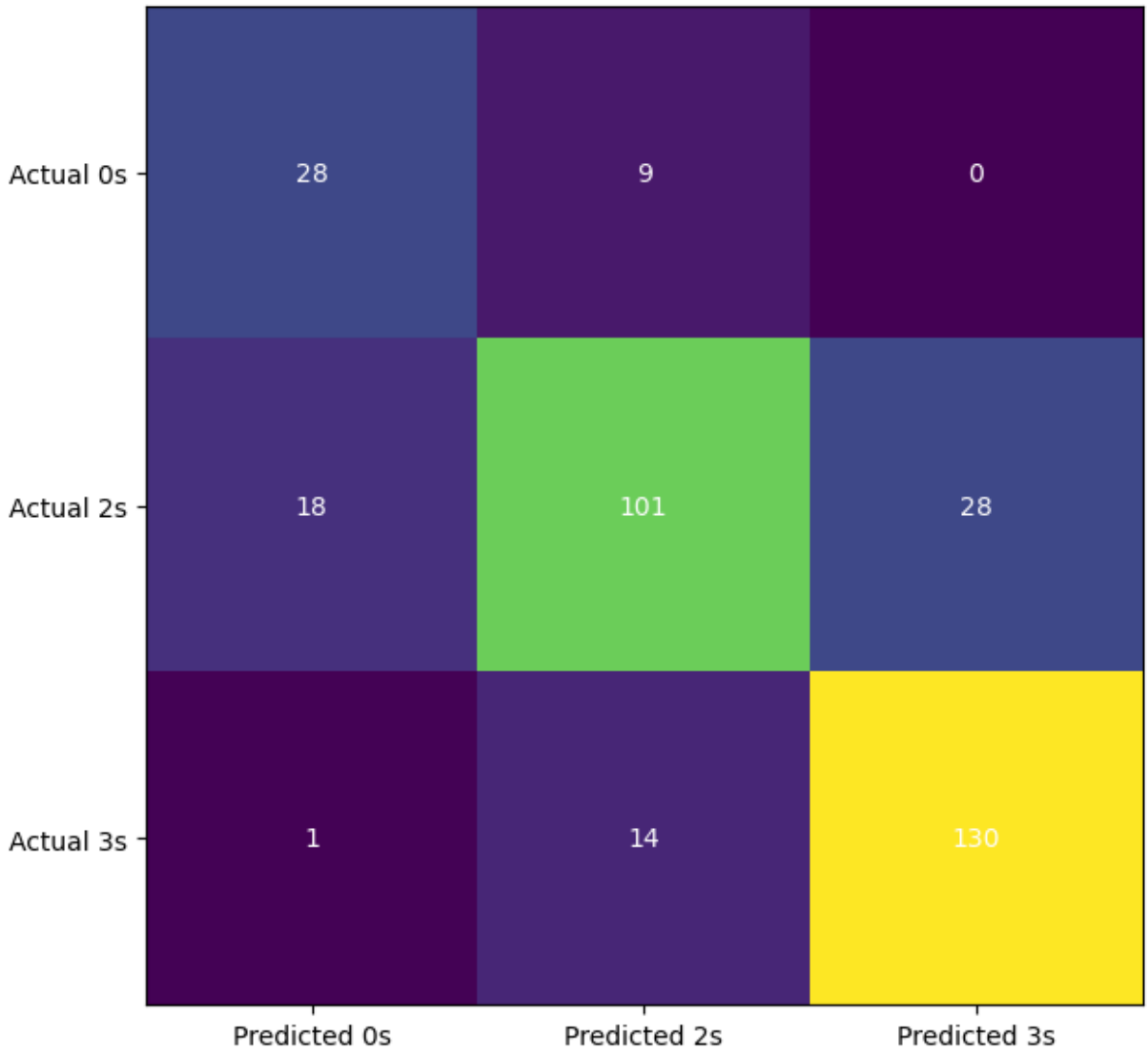
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TEST, testing_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

```
The Accuracy of Prediction is  0.7872340425531915
```

```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(Y_TEST, testing_data_prediction)
```

```
array([[ 28,   9,   0],
       [ 18, 101,  28],
       [  1,  14, 130]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TEST, testing_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TEST, testing_data_prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.60      0.76      0.67        37
           2       0.81      0.69      0.75       147
           3       0.82      0.90      0.86       145

    accuracy                           0.79       329
   macro avg       0.74      0.78      0.76       329
weighted avg       0.79      0.79      0.79       329
```

Logestic Regresssion

```
from sklearn.linear_model import LogisticRegression
shoaib = LogisticRegression(solver='liblinear', random_state=10)

shoaib.fit(X_TRAIN, Y_TRAIN)

LogisticRegression(random_state=10, solver='liblinear')

shoaib.classes_

array([0, 2, 3])

shoaib.intercept_

array([-5.74677728e-06, -1.68690127e-01,  3.91250054e-04])

shoaib.coef_

array([[ 7.10629709e-03,  2.33000403e-03, -8.18556130e-05,
        -1.13807573e-04,  4.33570746e-04, -2.23524332e-04,
         4.41128070e-04,  6.09409095e-04, -4.49568593e-03,
         1.55557571e-03,  1.48943633e-03,  6.52579066e-04,
        -5.98886528e-04,  1.85230637e-04, -2.31621934e-04,
         5.09010446e-04,  7.24064246e-04, -1.96211517e-04,
         4.08594496e-04,  3.24008899e-03, -6.80789660e-04,
         7.57547838e-06, -9.80390417e-05, -4.23114426e-03,
        -2.06137473e-03,  3.01025949e-04, -3.84405022e-05,
         2.40379609e-05, -2.65645842e-04, -1.95381252e-05,
        -5.65047378e-04,  1.43316047e-04,  8.25962238e-06,
        -1.45286723e-04, -3.91292144e-04, -4.58866874e-04,
         8.30876384e-05,  2.58291117e-03,  8.49880392e-05,
        -5.48617713e-06,  1.06128519e-05,  7.30273863e-04,
         4.10297159e-03,  2.39217694e-05],
       [-6.66040317e-03, -3.19204856e-03,  2.43895563e-05,
        -9.15280693e-02,  2.95231127e-01,  2.20444588e-01,
         4.39125257e-01,  2.99022557e-02, -1.80597859e-02,
        -2.70323797e-02,  6.11221505e-02, -1.98498017e-03,
         3.55168838e-01,  9.87158021e-01, -1.67892209e-01,
        -9.31577759e-02,  3.92817155e-04, -8.12615578e-02,
         9.80954742e-04, -6.38909599e-04, -9.06792117e-02,
        -4.17807736e-01,  1.12427215e-01,  7.96558159e-04,
         2.57993736e-03, -2.61542076e-04,  4.08543462e-01,
         1.46374409e+00, -3.62467711e-01, -1.06789971e+00,
         1.37601188e-01, -2.48961883e-01, -2.58430128e-01,
        -7.93209214e-02, -1.30848268e-02, -7.73238210e-02,
         7.08390157e-01, -1.89606224e-03, -1.53813230e-01,
        -3.04428249e-02,  1.96642028e-02,  1.09149072e-03,
        -3.04729871e-03, -1.13124477e-05],
       [-1.53048890e-02, -4.00618543e-03,  2.39926041e-05,
         7.02939252e-03, -1.23882454e-02, -1.38778992e-02,
```

```
        -1.52937410e-02, -5.32210491e-02,  4.52754476e-02,
        -4.46262715e-02, -4.28976790e-02,  2.26973560e-03,
         1.07990256e-02, -1.60776157e-02,  1.53168762e-02,
        -9.23550933e-03, -1.67805737e-03,  4.50476176e-03,
        -1.58516765e-03, -5.62518215e-04,  2.77272133e-02,
         3.42976256e-04, -4.54858775e-03,  1.49173793e-03,
        -1.40652092e-03,  1.64223618e-03,  5.50791587e-04,
        -9.80916110e-03,  1.12264008e-02,  1.73663674e-03,
         1.10221474e-02,  1.09042339e-02,  3.64395700e-03,
         8.60399943e-03,  1.16291757e-02,  1.32292935e-02,
        -4.78187920e-03, -2.50091353e-04,  1.02682941e-03,
         1.65898445e-03, -1.72148358e-04, -2.65938443e-03,
         3.19121120e-03, -4.22631721e-05]])
```

Prediction on Train Data

```
# accuracy for prediction on training data
training_data_prediction = shoaib.predict(X_TRAIN)
print(training_data_prediction)
```

```
[3 0 2 2 2 2 3 3 3 2 3 2 2 3 2 3 3 2 0 3 2 3 0 3 3 2 2 3 3 3 3 2 3 3 2
3 3
 3 3 0 3 2 3 0 2 2 3 2 3 3 0 3 2 3 2 2 3 2 3 2 2 2 3 2 2 3 3 3 3 2 3 3
2 2
 2 2 2 3 3 2 2 3 3 0 2 3 3 3 3 2 2 2 2 2 3 3 3 3 3 3 3 3 2 3 3 2 2 3 3
2 2
 3 2 3 2 3 3 3 3 3 3 3 2 2 0 3 2 3 2 2 2 2 2 3 2 0 3 3 2 3 3 2 3 3 3 3
3 3
 3 2 3 3 3 3 3 2 2 2 2 2 2 2 3 3 3 3 2 2 3 2 2 3 3 3 3 2 3 2 2 0 2 2 3
2 3
 3 2 2 3 2 2 3 2 2 3 2 3 3 3 2 2 3 2 2 2 3 3 3 3 3 2 3 2 3 2 3 2 2 3 3
3 2
 2 3 3 3 0 3 2 0 3 3 2 2 3 2 2 2 2 3 2 0 3 2 3 2 3 3 2 2 2 3 3 3 2 2 2
3 2
 0 3 2 3 2 0 3 2 3 2 3 2 2 3 3 0 3 2 2 2 2 2 2 2 3 3 3 3 2 3 2 3 2 0 3
3 3
 0 2 3 2 3 2 2 2 2 2 2 0 2 3 2 3 3 3 3 3 2 2 2 2 3 2 3 3 2 3 2 2 2 3 3
3 2
 2 2 2 2 2 3 3 3 3 3 2 2 2 3 3 3 3 3 2 3 2 0 2 2 2 2 3 2 0 2 2 2 2 3 2
3 3
 2 2 0 2 2 2 2 3 3 0 3 2 3 3 3 2 2 2 3 2 3 2 2 3 3 3 2 2 2 2 3 2 2 2 2
3 2
 2 2 2 0 3 2 3 3 2 2 3 3 2 3 0 3 2 2 2 3 2 3 3 2 2 2 3 0 0 2 3 3 2 3 2
2 3
 3 3 0 3 3 3 3 3 2 2 3 2 3 3 2 2 2 3 2 0 3 3 2 2 3 2 2 3 2 2 2 2 3 2 2
3 2
 3 2 3 2 2 2 2 3 2 2 2 3 2 2 3 2 3 2 2 3 0 3 2 3 3 2 3 3 2 3 3 3 2 3 3
3 3
 0 3 3 3 2 2 3 3 3 2 2 3 3 2 0 3 2 2 2 3 2 2 2 3 2 2 3 3 2 2 3 2 0 3 2
```

```
2 2
 0 0 3 2 3 2 2 3 2 3 2 2 3 2 2 3 2 0 3 2 3 0 2 3 3 3 3 2 3 3 2 3 3 3 3
0 3
 2 2 3 2 3 3 2 2 2 3 0 3 0 3 2 3 3 3 3 2 2 2 3 3 3 3 3 3 2 3 3 3 3 2
2 2
 3 2 2 2 3 3 3 2 2 3 0 2 2 3 2 3 2 3 3 2 3 3 2 2 2 0 3 2 3 3 2 3 2 0 3
3 2
 3 3 2 2 2 3 3 3 2 3 2 3 3 2 3 2 2 3 3 2 2 3 3 2 3 3 3 3 2 2 3 2 2 3 2
2 3
 3 3 2 3 3 3 2 3 3 3 2 3 2 3 3 3 3 2 2 2 2 3 2 3 3 2 2 2 3 3 3 2 2 3 3
2 3
 3 3 3 3 3 3 2 3 2 3 3 2 3 0 2 3 2 3 0 0 3 2 3 3 3 3 3 3 2 3 2 3 0 3 2
3 3
 3 2 3 2 2 2 0 3 3 3 3 3 3 2 2 2 2 2 2 2 2 3 3 3 2 2 3 3 3 3 2 2 3 2 3
3 3
 3 2 2 2 2 2 3 0 3 3 3 2 2 2 2 3 3 3 2 2 3 2 2 3 2 2 2 3 3 3 3 2 2 2 3
3 2
 3 3 3 2 3 2 2 3 2 3 3 2 2 3 2 3 2 3 3 2 2 2 2 0 2 3 3 3 2 3 2 2 3 3 3
3 3
 2 2 3 3 2 2 2 3 3 2 3 3 2 2 3 3 3 0 2 2 3 3 2 3 2 2 2 2 2 3 3 2 3 2 3
2 2
 2 3 3 3 2 2 3 2 2 3 2 2 3 2 3 2 2 2 2 3 0 3 3 3 2 2 3 2 2 3 3 2 2 3 2
2 3
 2 2 3 3 2 2 3 2 3 3 2 2 3 3 3 3 2 3 2 2 3 3 2]
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TRAIN, training_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

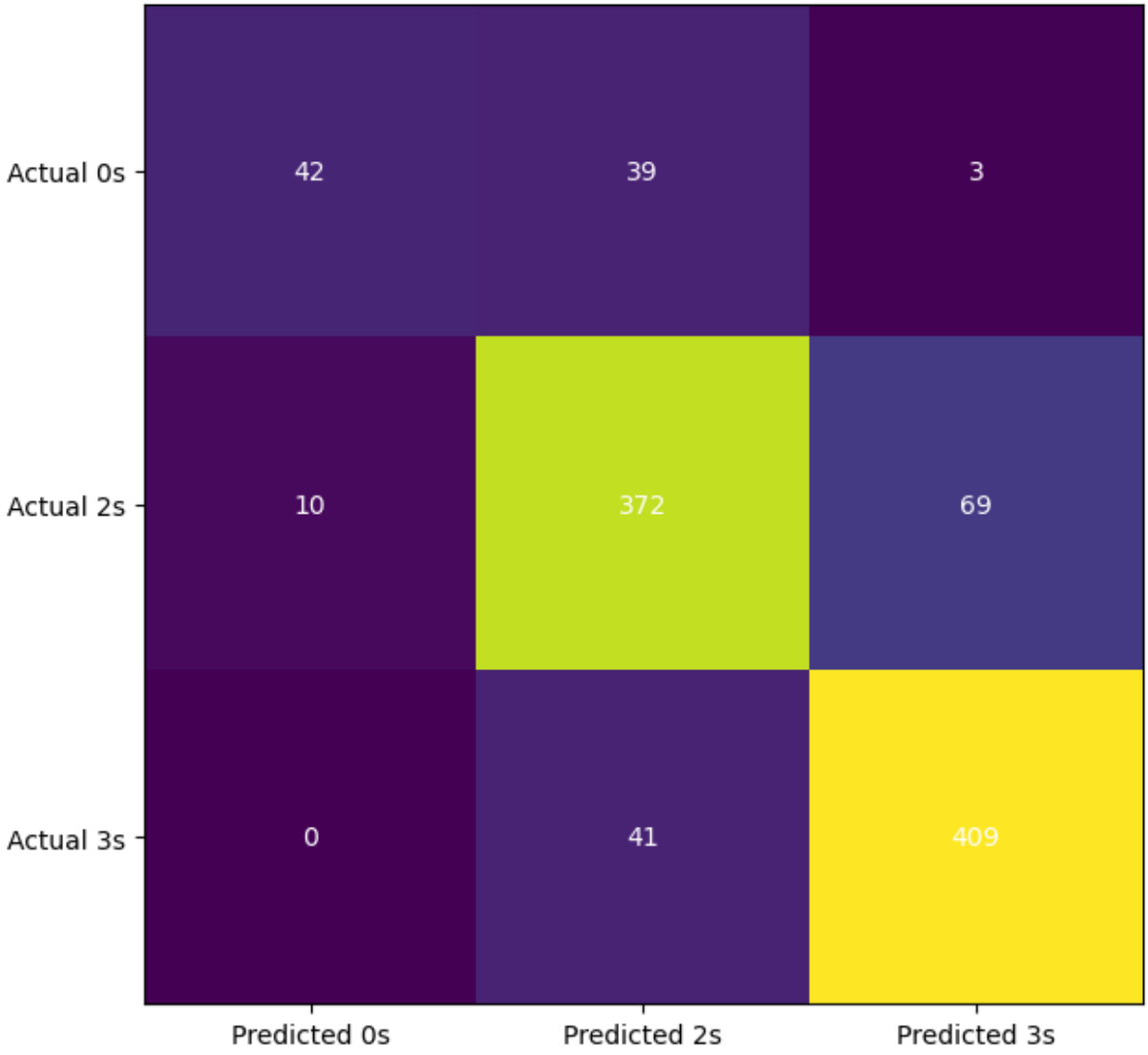
```
The Accuracy of Prediction is  0.8355329949238579
```

```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(Y_TRAIN, training_data_prediction)
```

```
array([[ 42,  39,   3],
       [ 10, 372,  69],
       [  0,  41, 409]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TRAIN, training_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
```

```
color='white')
plt.show()
```




```
report = classification_report(Y_TRAIN, training_data_prediction)
print(report)
```

	precision	recall	f1-score	support
0	0.81	0.50	0.62	84
2	0.82	0.82	0.82	451
3	0.85	0.91	0.88	450
accuracy			0.84	985
macro avg	0.83	0.74	0.77	985

```
weighted avg       0.83      0.84      0.83       985
```

Testing

```
# accuracy for prediction on training data
testing_data_prediction = shoaib.predict(X_TEST)
print(testing_data_prediction)
```

```
[3 2 2 3 2 2 3 3 2 3 2 3 2 2 2 3 2 3 0 3 3 3 2 2 2 3 2 2 2 3 0 2 2 2 2
3 0
 3 2 0 3 3 3 2 2 2 2 3 3 3 3 3 0 3 3 3 3 2 2 2 3 3 2 2 3 3 3 2 3 3 3 2
3 2
 3 2 2 3 2 2 3 3 3 3 2 3 2 3 3 2 2 2 3 2 3 2 3 3 2 3 2 2 2 2 3 2 3 0 3
0 3
 2 3 3 2 3 3 2 3 2 0 3 3 3 2 3 2 3 3 2 2 2 3 0 2 3 3 3 2 2 2 3 3 2 3 3
0 3
 2 2 3 3 0 3 2 2 2 2 3 3 2 2 2 3 2 3 2 3 2 3 3 3 3 2 0 2 2 3 3 2 0 2 3
3 3
 3 3 3 3 2 0 2 3 3 3 3 2 0 3 3 3 3 2 3 2 2 0 3 2 3 2 3 3 3 3 2 2 2 3 3
2 3
 2 3 2 2 3 3 2 2 3 3 2 2 3 3 3 3 2 3 3 2 3 2 2 2 2 3 0 2 2 2 3 2 2 2 3
2 3
 3 2 2 3 2 3 2 2 3 0 3 3 3 3 2 2 2 0 2 3 2 3 3 2 2 2 3 2 3 3 3 2 3 3 3
2 2
 2 3 2 2 3 2 2 2 3 2 3 2 3 3 3 2 2 3 2 3 3 2 3 0 3 2 3 3 3 2 3 2 2]
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TEST, testing_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

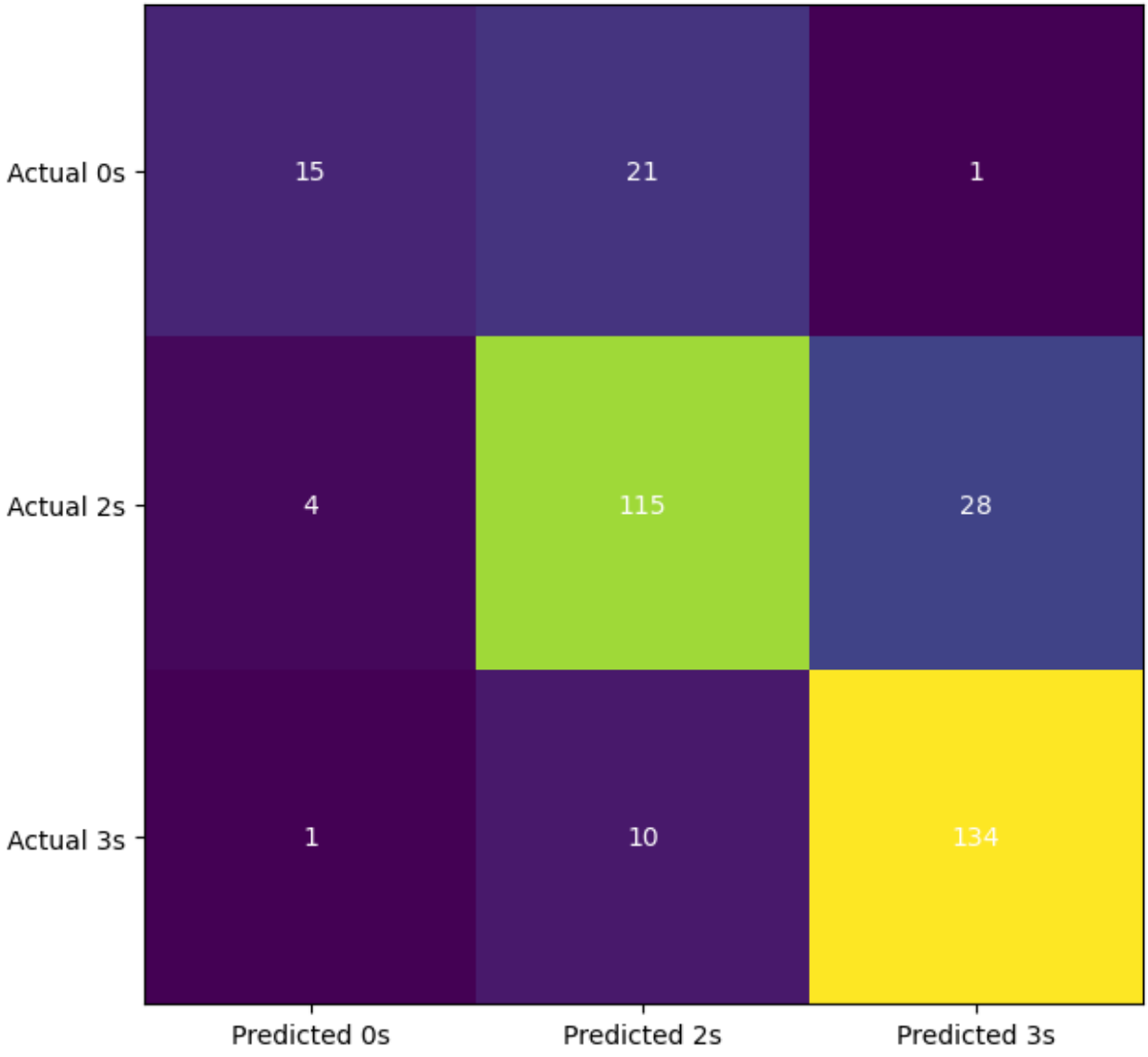
```
The Accuracy of Prediction is  0.8024316109422492
```

```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(Y_TEST, testing_data_prediction)
```

```
array([[ 15,  21,   1],
       [  4, 115,  28],
       [  1,  10, 134]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TEST, testing_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
```

```
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TEST, testing_data_prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.75      0.41      0.53        37
           2       0.79      0.78      0.78       147
           3       0.82      0.92      0.87       145

    accuracy                           0.80       329
   macro avg       0.79      0.70      0.73       329
```

```
weighted avg       0.80      0.80      0.79       329
```

SVM CLASSIFIER

```
#Import svm model
from sklearn import svm
#Create a svm Classifier
shoaib = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
shoaib.fit(X_TRAIN, Y_TRAIN)
```

```
SVC(kernel='linear')
```

Prediction on Train Data

```
# accuracy for prediction on training data
training_data_prediction = shoaib.predict(X_TRAIN)
print(training_data_prediction)
```

```
[3 0 2 2 2 2 3 3 3 2 3 2 2 3 2 2 3 2 0 3 2 3 0 3 3 2 2 3 3 3 3 2 3 3 3
3 3
 3 3 0 3 2 3 0 2 2 2 2 3 3 0 3 2 3 2 2 3 2 3 0 2 3 3 2 2 3 3 3 3 2 3 3
2 2
 2 2 2 3 3 2 2 3 3 0 2 3 3 3 3 2 2 2 2 3 2 3 3 3 3 3 3 3 2 3 3 2 2 3 3
2 3
 3 2 3 2 3 3 3 3 3 2 3 2 2 0 3 2 3 2 2 2 2 3 3 2 0 3 3 2 3 3 2 3 3 2 3
3 3
 3 2 3 3 3 3 3 2 2 3 2 2 2 2 3 3 3 3 2 0 3 2 2 3 3 3 2 2 3 2 2 0 2 2 3
2 3
 3 2 2 3 2 3 3 2 2 3 2 3 3 3 2 0 3 2 2 2 3 3 3 3 3 2 3 2 3 2 3 2 2 3 3
3 2
 2 3 3 3 0 3 2 0 2 3 2 2 3 2 2 2 2 3 2 0 3 2 3 0 3 3 2 2 2 3 3 3 2 2 2
3 2
 0 3 2 3 2 0 3 2 3 2 3 2 2 3 3 0 3 2 2 3 2 2 2 2 3 3 3 3 2 3 2 3 2 0 3
3 3
 0 2 2 2 3 2 2 2 2 2 2 0 2 3 2 3 3 3 3 3 2 2 2 2 3 2 3 3 2 3 2 2 2 3 3
3 2
 2 2 2 2 2 3 3 3 3 2 2 2 2 3 3 3 3 2 3 0 0 2 2 2 2 3 2 0 2 2 2 2 3 2
3 3
 2 3 0 2 2 2 3 2 3 0 3 2 2 3 3 2 3 2 3 2 3 2 2 3 3 3 2 2 2 2 3 2 2 2 2
3 2
 2 2 2 0 0 2 3 3 2 2 2 3 2 3 0 3 2 2 2 3 2 3 3 2 2 2 3 0 0 2 3 3 2 3 2
2 3
 2 3 0 2 2 3 3 3 2 0 2 2 3 3 2 2 2 3 2 0 3 3 2 2 3 2 2 3 2 2 2 2 3 2 2
3 2
```

```
 3 2 2 2 2 2 0 3 2 2 2 3 2 2 3 2 2 2 2 3 0 3 2 2 3 3 3 3 2 3 3 2 2 3 3
3 3
 0 3 3 3 2 2 3 3 3 3 2 3 3 2 2 3 2 0 2 3 0 2 2 3 0 2 3 3 2 2 3 2 0 3 2
2 2
 2 0 2 3 3 2 2 3 2 3 2 2 3 2 2 3 2 0 3 2 3 0 2 3 3 3 3 2 3 3 0 3 3 3 3
0 3
 2 2 3 2 3 3 2 2 2 3 0 3 0 3 2 3 3 3 3 3 2 0 3 2 3 3 3 3 3 2 3 3 2 3 2
3 2
 3 2 2 2 3 3 3 2 2 2 0 2 2 3 2 3 3 3 3 2 3 3 2 2 2 0 3 2 3 3 2 3 2 0 3
3 2
 3 3 2 2 2 3 3 3 2 3 0 3 3 2 3 0 2 3 3 2 2 3 2 2 3 2 3 3 2 2 3 2 2 3 2
2 3
 3 3 2 3 3 3 2 3 3 2 2 3 2 3 3 3 3 2 2 2 2 3 2 3 3 2 2 2 3 3 3 2 2 3 2
2 3
 3 2 2 3 3 3 2 3 2 3 3 2 3 0 2 3 2 3 0 0 3 0 3 3 3 3 3 3 2 3 3 3 0 3 2
3 3
 3 2 3 2 2 2 0 3 3 3 3 3 3 2 2 2 2 2 3 2 3 3 3 3 3 2 3 3 3 3 2 2 3 2 3
3 3
 3 2 2 2 2 2 3 0 3 2 3 2 2 2 2 3 3 3 2 2 3 2 2 3 2 2 2 3 3 2 3 3 2 2 2
3 2
 3 3 3 2 3 2 2 3 2 3 3 2 2 3 3 3 2 3 3 3 2 2 2 0 2 3 3 3 2 3 0 2 3 3 3
3 3
 2 2 3 2 2 2 2 3 3 2 3 3 2 2 2 3 3 0 2 2 3 3 2 3 2 2 2 2 2 2 3 2 3 2 3
2 2
 2 3 3 3 2 2 3 2 2 2 2 3 3 2 3 2 2 2 2 3 0 2 3 3 2 2 3 2 2 3 3 2 2 3 2
2 3
 2 3 3 3 2 2 2 2 3 3 2 0 2 3 3 0 2 3 2 3 3 3 2]
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TRAIN, training_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

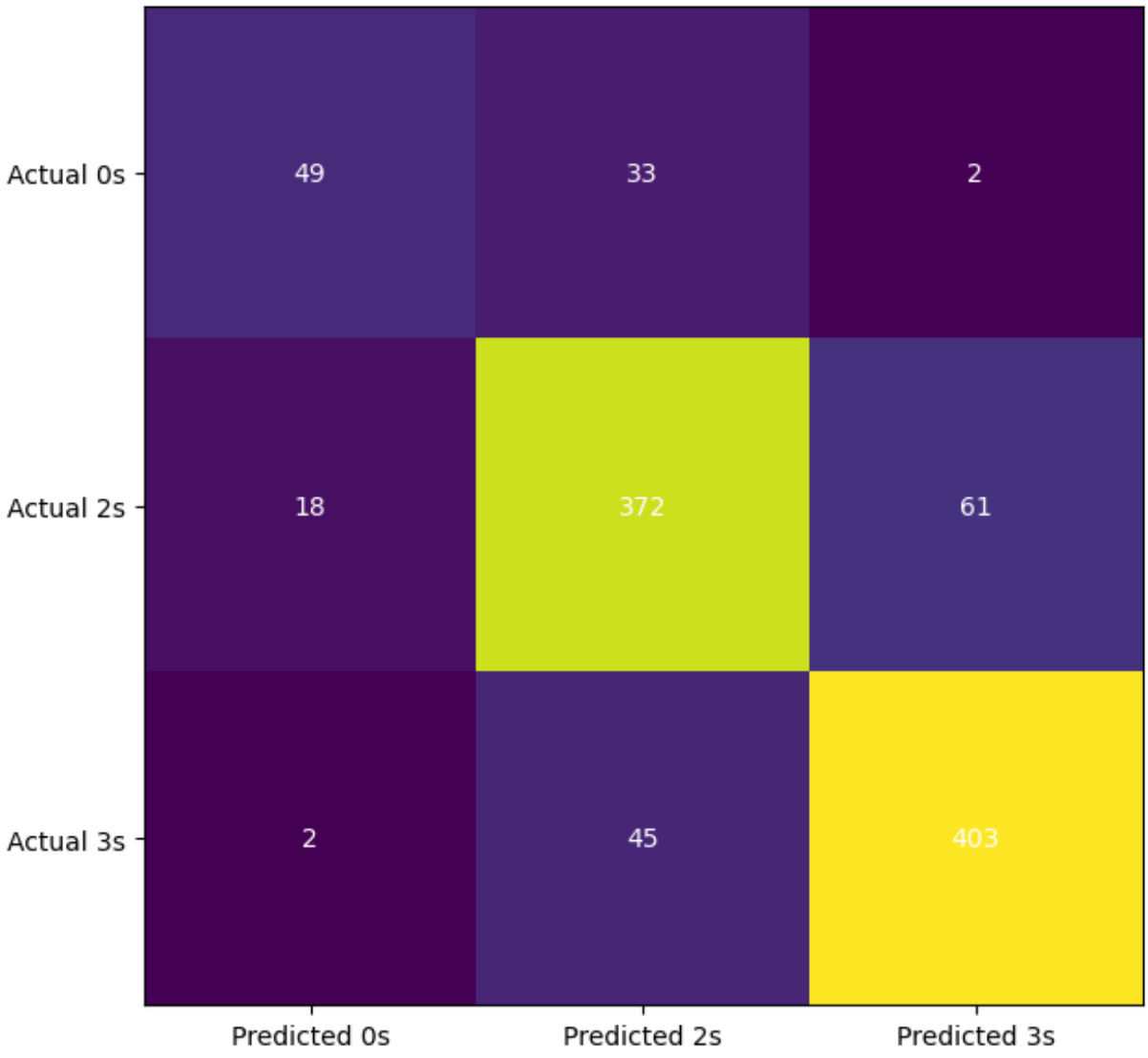
```
The Accuracy of Prediction is  0.8365482233502538
```

```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(Y_TRAIN, training_data_prediction)
```

```
array([[ 49,  33,   2],
       [ 18, 372,  61],
       [  2,  45, 403]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TRAIN, training_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
```

```
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TRAIN, training_data_prediction)
print(report)
```

	precision	recall	f1-score	support
0	0.71	0.58	0.64	84
2	0.83	0.82	0.83	451
3	0.86	0.90	0.88	450
accuracy			0.84	985

```
   macro avg       0.80      0.77      0.78       985
weighted avg       0.83      0.84      0.83       985
```

Testing on Test Data

```
# accuracy for prediction on training data
testing_data_prediction = shoaib.predict(X_TEST)
print(testing_data_prediction)
```

```
[3 2 2 3 2 3 2 3 2 3 2 2 3 2 2 3 2 3 0 3 3 2 3 0 2 3 2 2 0 3 0 2 2 2 2
2 0
 3 0 0 3 3 3 2 0 2 2 3 3 3 3 3 0 2 3 3 3 2 2 2 3 3 2 2 3 3 3 2 3 3 3 3
3 2
 3 2 2 3 2 2 3 3 3 3 2 3 2 2 2 2 2 2 3 2 3 3 3 3 2 3 2 2 2 2 3 2 3 2 3
0 3
 2 3 0 0 3 3 2 3 2 0 3 3 3 2 3 2 3 3 2 2 2 3 0 2 3 3 3 2 2 2 3 3 2 3 3
0 2
 2 2 2 3 0 2 3 2 2 2 2 2 2 2 2 3 2 3 2 3 2 3 3 3 3 3 0 2 2 3 3 2 0 2 3
3 2
 3 3 2 3 2 0 2 3 3 3 2 2 0 3 3 3 3 2 3 2 2 0 3 2 3 2 2 3 3 3 2 2 0 3 3
2 3
 2 3 2 2 3 3 0 2 3 2 2 2 3 3 3 3 3 3 2 3 2 2 2 2 2 0 2 0 2 3 2 0 2 2
2 3
 3 0 2 3 2 3 2 3 3 0 2 3 2 2 2 2 2 2 2 3 2 3 3 2 2 2 3 2 3 3 3 0 3 2 3
2 2
 2 3 2 2 3 2 2 2 2 2 3 2 3 2 3 2 2 3 2 2 3 2 3 0 3 2 3 3 3 2 3 3 2]
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TEST, testing_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

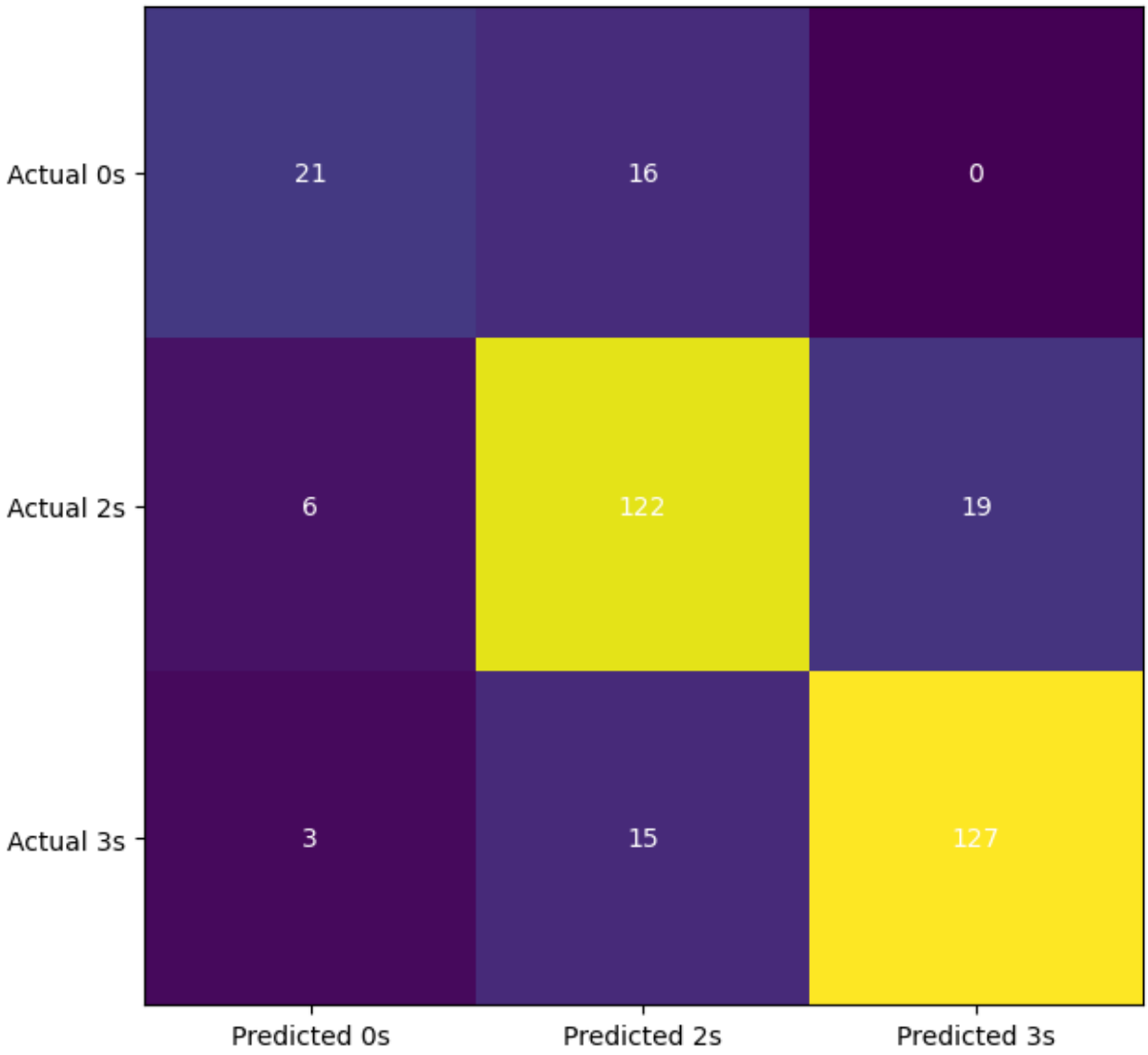
```
The Accuracy of Prediction is  0.8206686930091185
```

```
from sklearn.metrics import classification_report, confusion_matrix
confusion_matrix(Y_TEST, testing_data_prediction)
```

```
array([[ 21,  16,   0],
       [  6, 122,  19],
       [  3,  15, 127]])
```

```
import matplotlib.pyplot as plt
cm =confusion_matrix(Y_TEST, testing_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
```

```
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TEST, testing_data_prediction)
print(report)
```

	precision	recall	f1-score	support
0	0.70	0.57	0.63	37
2	0.80	0.83	0.81	147
3	0.87	0.88	0.87	145

```
    accuracy                           0.82       329
   macro avg       0.79      0.76      0.77       329
weighted avg       0.82      0.82      0.82       329
```

DECISION TREE CLASSIFIER

```
# from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# initialize decision tree classifier
dtree = DecisionTreeClassifier(random_state=30)

# train the classifier
dtree.fit(X_TRAIN, Y_TRAIN)
```

```
DecisionTreeClassifier(random_state=30)
```

Prediction on Train Data

```
# accuracy for prediction on training data
training_data_prediction = dtree.predict(X_TRAIN)
print(training_data_prediction)
```

```
[3 0 2 2 2 2 3 3 3 2 3 2 2 3 0 2 3 2 0 3 2 3 0 3 3 2 2 3 3 3 2 2 3 3 3
3 3
 3 3 0 3 2 2 0 2 2 3 2 3 3 0 3 2 3 0 2 3 2 3 2 2 3 3 2 2 3 2 2 3 2 2 3
2 2
 3 2 2 3 3 2 2 3 3 0 2 3 3 2 3 2 2 2 2 3 2 3 3 3 3 3 3 3 2 3 3 2 2 3 3
2 3
 3 2 3 2 3 3 3 3 2 2 3 2 2 0 3 2 3 2 2 2 2 3 3 0 0 3 2 2 3 3 2 2 3 3 3
2 3
 3 2 3 3 3 3 3 2 2 2 2 2 2 0 3 3 3 3 2 0 3 3 2 3 2 3 3 2 3 2 2 2 0 2 3
2 3
 3 0 2 3 2 3 3 2 3 3 2 3 3 3 2 2 3 2 2 2 2 3 3 3 3 2 3 2 3 2 3 2 2 3 3
3 2
 0 3 3 3 0 3 2 0 3 3 2 0 3 2 2 0 2 3 2 0 3 2 3 2 3 3 2 2 2 2 3 3 2 2 2
2 2
 0 3 2 2 2 2 3 2 3 2 3 3 2 3 3 0 3 2 2 2 2 2 2 0 3 3 3 3 2 3 0 3 2 0 3
3 3
 0 2 2 2 3 2 2 0 3 0 2 0 2 3 2 3 3 3 3 3 0 2 2 2 2 2 3 3 2 3 2 3 2 3 3
3 2
 0 2 2 2 2 3 3 3 3 3 2 2 2 3 3 3 3 3 2 3 0 0 2 2 3 2 3 2 0 0 2 2 2 3 2
3 3
 2 2 0 2 2 2 2 2 3 0 3 2 3 3 3 3 2 2 2 2 3 2 2 3 3 2 2 3 2 2 2 2 2 3 2
3 2
 2 0 2 0 3 2 3 3 2 2 2 3 0 3 2 2 2 2 2 3 3 3 3 2 2 3 3 0 2 2 3 3 2 3 2
```

```
2 3
 2 3 0 2 2 3 3 3 2 2 2 2 3 3 2 2 2 3 2 0 3 3 2 3 3 2 2 3 2 2 2 2 3 2 2
2 2
 3 3 2 2 2 2 0 3 2 2 2 3 2 2 3 3 3 2 0 3 0 2 2 3 3 3 3 3 2 2 3 2 2 3 3
3 3
 0 3 3 3 2 2 3 3 3 3 2 2 2 2 0 3 2 0 2 3 2 2 2 3 0 2 3 2 2 2 2 3 0 3 2
2 2
 2 0 2 3 3 2 2 3 2 3 2 2 3 2 0 3 2 2 3 2 3 0 2 2 3 3 3 2 3 3 2 3 3 3 3
2 2
 2 0 3 2 3 3 2 2 2 3 0 2 2 3 2 2 3 3 3 3 2 3 3 3 3 3 3 3 3 2 3 3 2 3 2
2 2
 2 2 2 2 3 3 3 0 3 2 0 2 2 3 2 3 3 3 2 2 3 3 2 2 2 0 3 2 3 3 0 3 2 0 3
3 2
 0 3 2 2 2 3 3 3 3 3 0 2 3 2 2 2 2 3 3 3 2 3 3 2 3 3 3 3 2 2 3 2 2 3 2
2 2
 3 3 2 3 3 2 2 0 2 2 2 3 0 3 3 3 3 2 2 2 2 3 2 3 3 2 2 2 3 3 3 2 2 3 2
2 3
 3 3 3 3 3 3 2 3 2 3 3 2 3 0 2 3 2 3 2 2 3 0 3 3 3 3 3 3 2 3 3 3 0 3 2
3 3
 3 3 3 2 2 2 0 3 3 3 2 3 3 2 2 2 0 2 2 2 2 2 3 3 2 3 3 3 3 3 2 2 3 2 3
2 3
 3 2 2 2 0 2 3 0 2 3 3 2 3 2 0 2 3 2 2 2 3 0 2 3 2 0 2 3 2 3 3 2 2 2 3
3 2
 2 3 3 2 3 2 2 2 3 3 3 2 2 2 3 3 2 3 3 3 2 2 2 0 2 3 3 3 2 3 2 2 3 3 3
3 3
 2 2 3 2 2 2 2 3 3 2 3 3 2 2 3 3 3 0 2 2 3 3 2 3 2 2 2 2 2 3 3 2 3 2 3
2 3
 0 3 3 3 2 2 3 2 2 3 2 3 3 2 3 2 2 2 2 3 0 3 3 3 2 2 3 2 0 2 3 2 2 3 2
2 3
 2 3 3 3 0 2 3 2 3 3 2 2 2 3 3 0 2 3 2 2 2 3 2]
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TRAIN, training_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

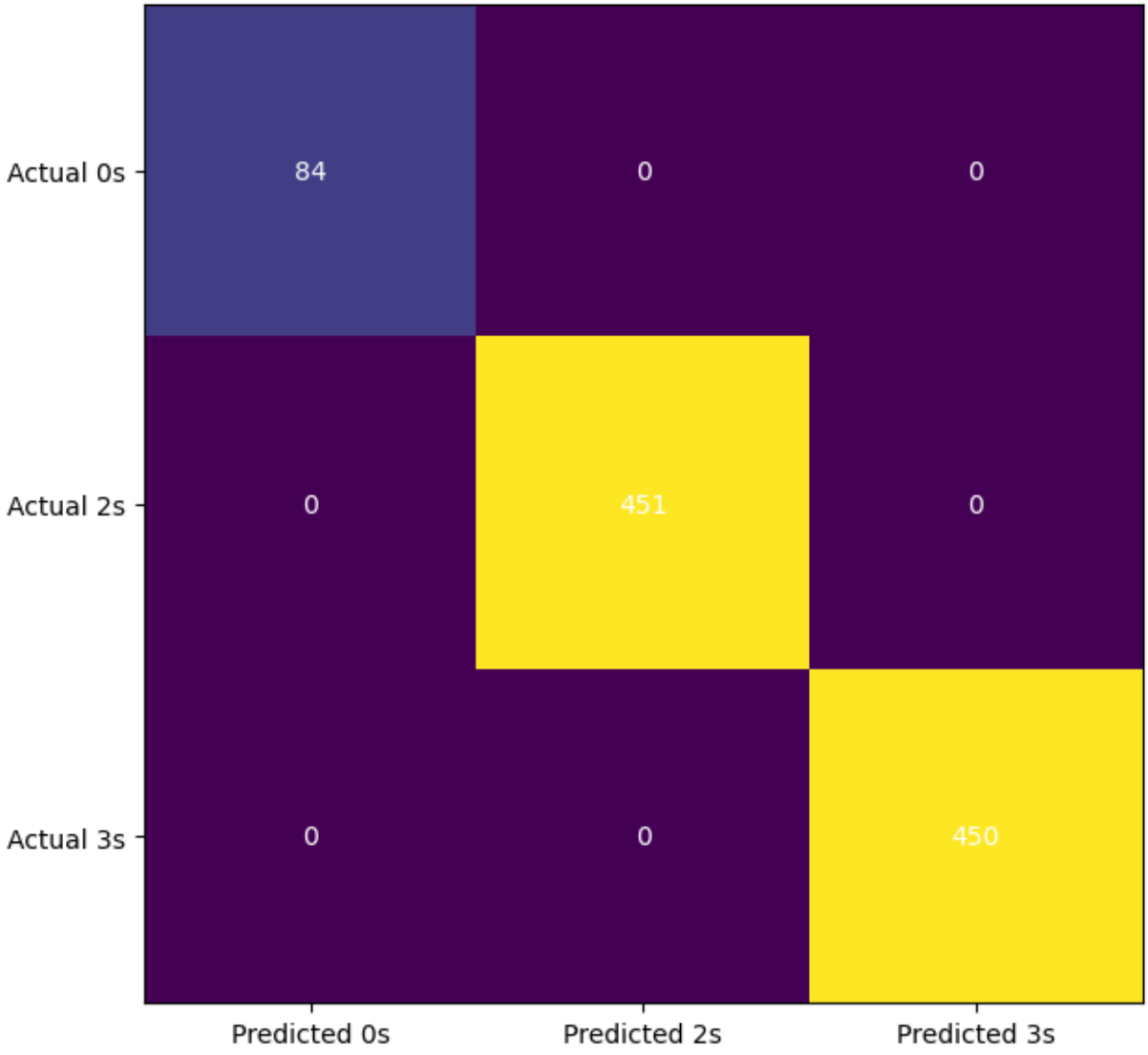
```
The Accuracy of Prediction is  1.0
```

```
confusion_matrix(Y_TRAIN, training_data_prediction)
```

```
array([[ 84,   0,   0],
       [  0, 451,   0],
       [  0,   0, 450]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TRAIN, training_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
```

```
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TRAIN, training_data_prediction)
print(report)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	84
2	1.00	1.00	1.00	451
3	1.00	1.00	1.00	450

```
    accuracy                           1.00       985
   macro avg       1.00      1.00      1.00       985
weighted avg       1.00      1.00      1.00       985
```

Prediction on Test Data

```
# accuracy for prediction on training data
testing_data_prediction = dtree.predict(X_TEST)
print(testing_data_prediction)
```

```
[3 2 2 3 2 3 3 3 2 3 2 3 3 2 2 3 3 3 0 3 3 2 2 0 2 3 0 2 2 3 0 2 2 2 2
3 0
 3 0 0 3 3 2 2 3 2 2 2 3 3 3 3 0 2 3 3 3 0 2 2 3 3 2 0 3 3 2 2 3 3 3 2
3 2
 2 2 0 3 2 2 3 3 3 3 2 3 3 2 2 2 2 2 3 2 3 3 2 3 2 3 2 2 2 2 0 2 3 3 3
0 3
 2 3 3 0 3 3 2 3 2 0 2 3 2 0 3 2 3 3 0 3 2 3 0 2 2 3 3 2 2 3 2 3 3 3 2
0 3
 2 2 2 2 0 2 3 2 3 2 2 3 2 2 2 3 0 3 2 3 2 3 3 3 3 3 0 2 0 3 3 2 0 0 3
3 2
 3 3 2 2 2 0 2 3 3 2 3 2 0 3 3 3 3 2 3 2 2 0 3 3 3 3 3 3 3 3 2 2 0 3 3
2 2
 2 2 2 2 3 3 0 0 3 3 2 2 3 3 2 3 3 3 3 2 2 2 2 2 2 3 2 2 2 2 3 2 0 2 3
2 2
 3 2 2 3 0 3 2 2 3 0 3 3 0 2 2 2 2 0 0 3 2 3 3 2 2 2 3 2 3 3 3 3 2 3 3
2 2
 2 3 2 2 3 2 2 2 3 2 3 2 2 3 3 2 2 3 2 2 0 2 3 2 3 2 2 3 3 2 3 2 2]
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TEST, testing_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

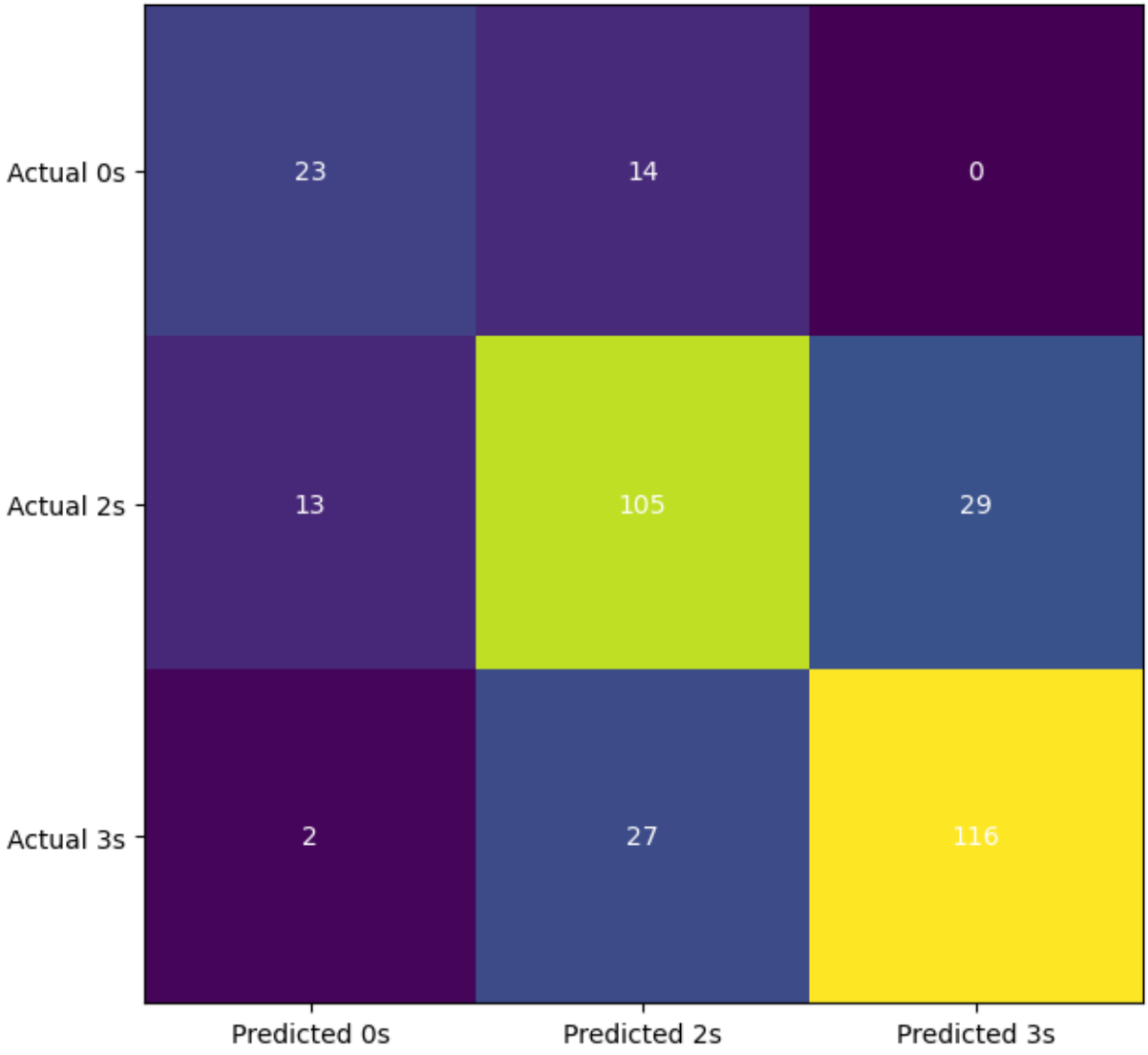
```
The Accuracy of Prediction is  0.7416413373860182
```

```
confusion_matrix(Y_TEST, testing_data_prediction)
```

```
array([[ 23,  14,   0],
       [ 13, 105,  29],
       [  2,  27, 116]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TEST, testing_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
```

```
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TEST, testing_data_prediction)
print(report)
```

	precision	recall	f1-score	support
0	0.61	0.62	0.61	37
2	0.72	0.71	0.72	147
3	0.80	0.80	0.80	145

```
    accuracy                           0.74       329
   macro avg       0.71      0.71      0.71       329
weighted avg       0.74      0.74      0.74       329
```

Over fitting existed in Decession Tree

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=7, random_state=15)
# train the classifier
rf_model.fit(X_TRAIN, Y_TRAIN)
```

```
RandomForestClassifier(n_estimators=7, random_state=15)
```

Prediction on Training Data

```
# accuracy for prediction on training data
training_data_prediction = rf_model.predict(X_TRAIN)
training_data_prediction
```

```
array([3, 0, 2, 2, 2, 2, 3, 3, 3, 2, 3, 2, 2, 3, 0, 2, 3, 2, 0, 3, 2,
3,
       0, 3, 3, 2, 2, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 0, 3, 2, 2,
0,
       2, 2, 3, 2, 3, 3, 0, 3, 2, 3, 2, 2, 3, 2, 3, 2, 2, 3, 3, 2, 2,
3,
       2, 2, 3, 2, 2, 3, 2, 2, 3, 2, 2, 3, 3, 2, 2, 3, 3, 0, 2, 3, 3,
2,
       3, 2, 2, 2, 2, 3, 2, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 2, 3, 3,
2,
       3, 3, 2, 3, 2, 3, 3, 3, 3, 2, 2, 3, 2, 2, 0, 3, 2, 3, 2, 2, 2,
2,
       3, 3, 0, 0, 3, 2, 2, 3, 3, 2, 2, 3, 3, 3, 2, 3, 3, 2, 3, 2, 3,
3,
       3, 2, 2, 2, 2, 2, 2, 0, 3, 3, 3, 3, 2, 0, 3, 3, 2, 3, 2, 3, 3,
2,
       3, 2, 2, 2, 0, 2, 3, 0, 3, 3, 0, 2, 3, 2, 3, 3, 2, 3, 3, 2, 3,
3,
       3, 2, 2, 3, 2, 2, 2, 2, 3, 3, 3, 3, 2, 3, 2, 3, 2, 3, 2, 2, 3,
3,
       3, 2, 0, 3, 3, 3, 0, 3, 2, 0, 3, 3, 2, 0, 3, 2, 2, 0, 2, 3, 2,
0,
       3, 2, 3, 2, 3, 3, 2, 2, 2, 2, 3, 3, 2, 2, 2, 2, 2, 0, 3, 2, 3,
```

```
2,
      2, 3, 2, 3, 2, 3, 3, 2, 3, 3, 0, 3, 2, 2, 2, 2, 2, 2, 0, 3, 3,
3,
      3, 2, 3, 2, 3, 2, 0, 3, 3, 3, 0, 2, 2, 2, 3, 2, 2, 0, 3, 0, 2,
0,
      2, 3, 2, 3, 3, 3, 3, 3, 0, 2, 2, 2, 2, 2, 3, 3, 2, 3, 2, 3, 2,
3,
      3, 3, 2, 0, 2, 2, 2, 2, 3, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3,
2,
      3, 0, 0, 2, 2, 3, 2, 3, 2, 0, 0, 2, 2, 2, 3, 2, 3, 3, 2, 2, 0,
2,
      2, 2, 3, 2, 3, 0, 3, 2, 3, 3, 3, 3, 2, 2, 2, 2, 3, 2, 2, 3, 3,
2,
      2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 0, 2, 0, 3, 2, 3, 3, 2, 2,
2,
      3, 0, 3, 2, 2, 2, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 0, 2, 2, 3, 3,
2,
      3, 2, 2, 3, 2, 3, 0, 2, 2, 3, 3, 3, 2, 2, 2, 2, 3, 3, 2, 2, 2,
3,
      2, 0, 3, 3, 2, 3, 3, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 3,
2,
      2, 2, 2, 0, 3, 2, 2, 2, 3, 2, 2, 3, 3, 3, 2, 0, 3, 0, 2, 2, 3,
3,
      3, 3, 3, 2, 2, 3, 2, 2, 3, 3, 3, 3, 0, 3, 3, 3, 2, 2, 3, 3, 3,
3,
      2, 2, 2, 2, 0, 3, 2, 0, 2, 3, 2, 2, 2, 3, 0, 2, 3, 2, 2, 2, 3,
3,
      0, 3, 2, 2, 2, 2, 0, 2, 3, 3, 2, 2, 3, 2, 3, 2, 2, 3, 2, 0, 3,
2,
      2, 3, 2, 3, 0, 2, 2, 3, 3, 3, 2, 3, 3, 2, 3, 3, 3, 3, 2, 2, 2,
0,
      3, 2, 3, 3, 2, 2, 2, 3, 0, 2, 2, 3, 2, 2, 3, 3, 3, 3, 2, 3, 3,
3,
      3, 3, 3, 3, 3, 2, 3, 3, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 0,
3,
      2, 0, 2, 2, 3, 2, 3, 3, 3, 2, 2, 3, 3, 2, 2, 2, 0, 3, 2, 3, 3,
0,
      3, 2, 0, 3, 3, 2, 0, 3, 2, 2, 2, 3, 3, 3, 3, 3, 0, 2, 3, 2, 2,
2,
      2, 3, 3, 3, 2, 3, 3, 2, 3, 3, 3, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2,
3,
      3, 2, 3, 3, 3, 2, 0, 2, 2, 2, 3, 0, 3, 3, 3, 3, 2, 2, 2, 2, 3,
2,
      3, 3, 2, 2, 2, 3, 3, 3, 2, 2, 3, 2, 2, 3, 3, 3, 2, 3, 3, 3, 2,
3,
      2, 3, 3, 2, 3, 0, 2, 3, 2, 3, 2, 2, 3, 0, 3, 3, 3, 3, 3, 3, 2,
3,
      3, 3, 0, 3, 2, 3, 3, 3, 3, 3, 2, 2, 2, 0, 3, 3, 3, 2, 3, 3, 2,
2,
```

```
       2, 0, 2, 2, 2, 2, 2, 3, 3, 2, 3, 3, 3, 3, 3, 2, 2, 3, 2, 3, 2,
3,
       3, 2, 2, 2, 0, 2, 3, 0, 2, 3, 3, 2, 3, 2, 0, 2, 3, 2, 2, 2, 3,
0,
       2, 3, 2, 0, 2, 3, 2, 3, 3, 2, 2, 2, 3, 3, 2, 2, 3, 3, 2, 3, 2,
2,
       2, 2, 3, 3, 2, 2, 2, 3, 3, 2, 3, 3, 3, 2, 2, 2, 0, 2, 3, 3, 3,
2,
       3, 2, 2, 3, 3, 3, 3, 3, 2, 2, 3, 2, 2, 2, 2, 3, 3, 2, 3, 3, 2,
2,
       3, 3, 3, 0, 2, 2, 3, 3, 2, 3, 2, 2, 2, 2, 2, 3, 3, 2, 3, 2, 3,
2,
       3, 2, 3, 3, 3, 2, 2, 3, 2, 2, 3, 2, 3, 3, 2, 3, 2, 2, 2, 2, 3,
0,
       3, 3, 3, 2, 2, 3, 2, 0, 2, 3, 2, 2, 3, 2, 2, 3, 2, 3, 3, 3, 0,
2,
       3, 2, 3, 3, 2, 2, 2, 3, 3, 0, 2, 3, 2, 2, 2, 3, 2])
```

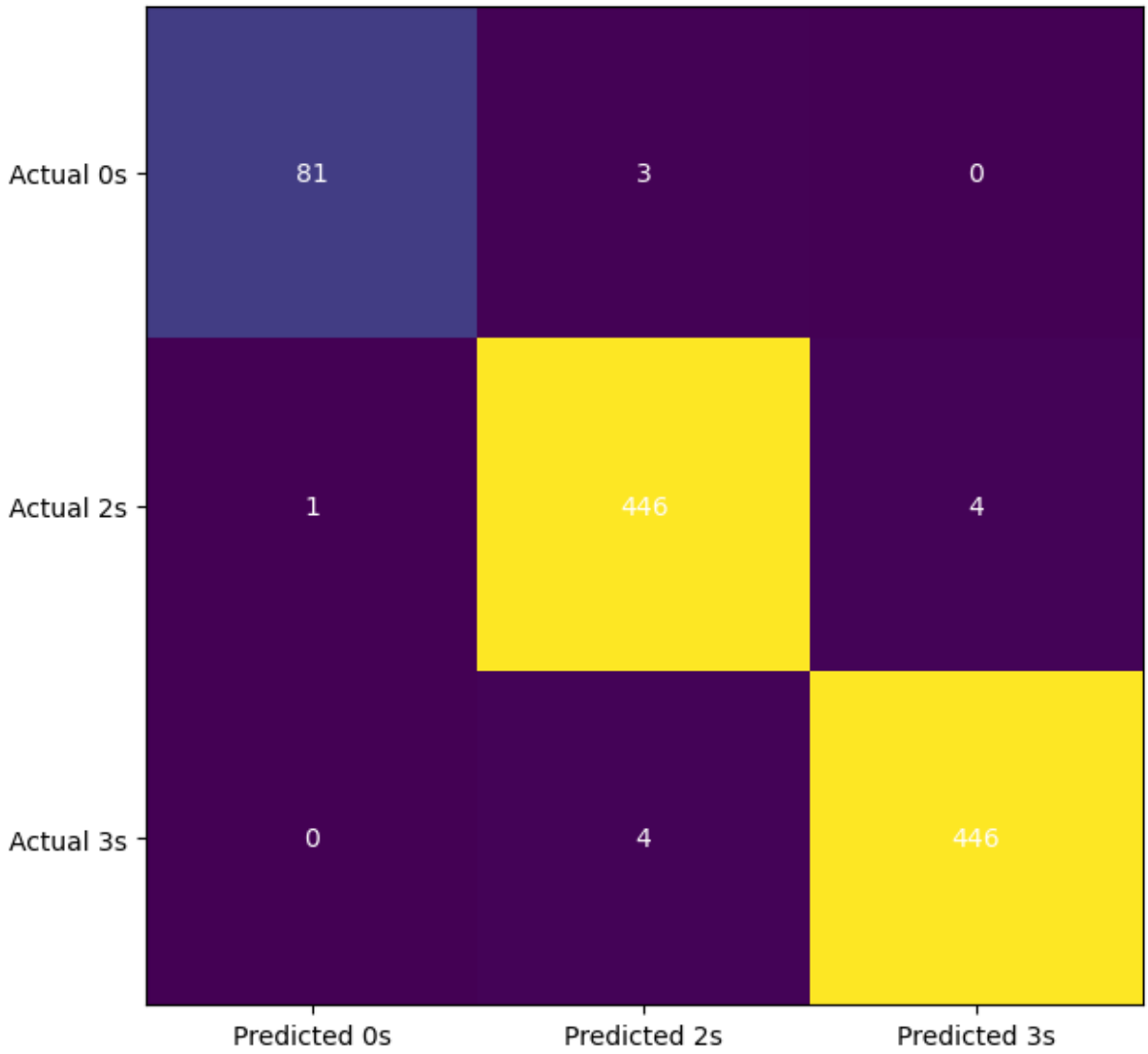
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TRAIN, training_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

```
The Accuracy of Prediction is  0.9878172588832488
```

```
confusion_matrix(Y_TRAIN, training_data_prediction)
```

```
array([[ 81,   3,   0],
       [  1, 446,   4],
       [  0,   4, 446]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TRAIN, training_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TRAIN, training_data_prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.99      0.96      0.98        84
           2       0.98      0.99      0.99       451
           3       0.99      0.99      0.99       450

    accuracy                           0.99       985
   macro avg       0.99      0.98      0.98       985
weighted avg       0.99      0.99      0.99       985
```

Prediction on Test Data

```
# accuracy for prediction on training data
testing_data_prediction = rf_model.predict(X_TEST)
print(testing_data_prediction)
```

```
[3 2 2 3 3 3 2 3 2 3 2 2 2 2 2 3 2 3 2 3 3 2 3 0 2 3 2 2 2 3 0 2 2 2 2
2 0
 3 2 0 3 3 2 2 3 2 2 3 3 3 3 3 0 2 3 3 2 2 2 2 3 3 2 0 3 3 3 2 3 3 3 3
3 2
 3 2 2 3 2 2 3 3 3 3 2 3 2 2 2 2 2 2 3 2 3 3 3 3 2 3 2 2 3 2 3 2 3 2 3
0 3
 2 3 3 0 3 3 0 3 2 0 3 3 3 2 3 2 3 3 2 2 2 3 0 2 3 3 3 2 2 2 3 3 0 3 3
0 3
 2 2 2 3 0 2 2 2 2 2 3 2 2 2 2 3 2 3 2 3 2 3 3 3 3 3 2 2 2 3 3 2 0 2 3
3 3
 3 3 2 3 2 0 2 3 3 2 3 2 0 3 3 3 3 2 3 2 2 0 3 3 3 3 3 3 3 3 2 2 0 3 3
3 3
 2 3 2 2 3 3 0 0 3 3 2 2 2 3 2 3 3 3 3 2 3 2 2 2 2 2 0 2 2 0 3 2 2 2 2
2 3
 3 2 2 3 2 3 2 2 3 0 3 3 0 2 2 2 2 2 2 3 2 3 3 2 2 2 3 2 3 3 3 3 3 3 3
2 2
 2 3 2 2 3 2 2 2 3 2 3 2 2 3 3 2 2 3 2 2 3 2 3 3 3 2 3 3 3 2 3 2 2]
```

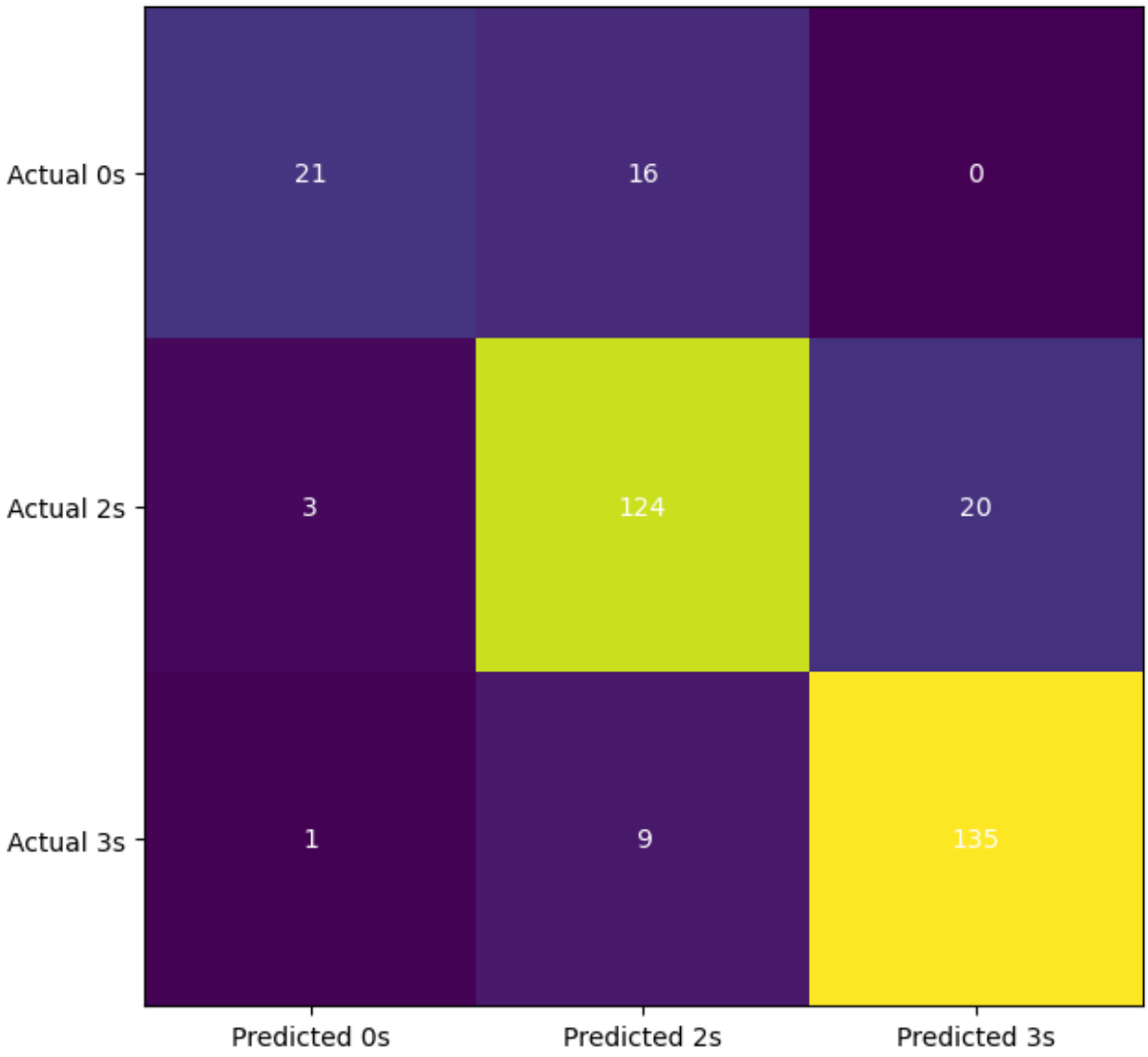
```
accuracy = accuracy_score(Y_TEST, testing_data_prediction)
print("The Accuracy of Prediction is ", accuracy)
```

```
The Accuracy of Prediction is  0.851063829787234
```

```
confusion_matrix(Y_TEST, testing_data_prediction)
```

```
array([[ 21,  16,   0],
       [  3, 124,  20],
       [  1,   9, 135]])
```

```
import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TEST, testing_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()
```




```
report = classification_report(Y_TEST, testing_data_prediction)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.84      0.57      0.68        37
           2       0.83      0.84      0.84       147
           3       0.87      0.93      0.90       145

    accuracy                           0.85       329
   macro avg       0.85      0.78      0.81       329
weighted avg       0.85      0.85      0.85       329
```

Over fitting existed

Hyperparmater tuning using GridSearchCV

GridSearchCV is a scikit-learn function that automates the hyperparameter tuning process¶

and helps to find the best hyperparameters for a given machine learning model.

```
from sklearn.model_selection import GridSearchCV
# Hyperparameter to fine tune
param_grid = {
    'max_depth': range(1, 10, 1),
    'min_samples_leaf': range(1, 20, 2),
    'min_samples_split': range(3, 20, 2),
    'criterion': ["entropy", "gini"]
}

# Decision tree classifier
tree = DecisionTreeClassifier(random_state=5)

# GridSearchCV
# CV=Cross Validation
grid_search = GridSearchCV(estimator=tree, param_grid=param_grid,
cv=5, verbose=True)

grid_search.fit(X_TRAIN, Y_TRAIN)
```

```
Fitting 5 folds for each of 1620 candidates, totalling 8100 fits
```

```
# Best score and estimator
print("best accuracy", grid_search.best_score_)
print(grid_search.best_estimator_)

# accuracy for prediction on training data
training_data_prediction = grid_search.predict(X_TRAIN)
training_data_prediction

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_TRAIN, training_data_prediction)
print("The Accuracy of Prediction is ", accuracy)

confusion_matrix(Y_TRAIN, training_data_prediction)

import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TRAIN, training_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
```

```
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()

report = classification_report(Y_TRAIN, training_data_prediction)
print(report)
```

Testing

```
# accuracy for prediction on training data
testing_data_prediction = grid_search.predict(X_TEST)
print(testing_data_prediction)


accuracy = accuracy_score(Y_TEST, testing_data_prediction)
print("The Accuracy of Prediction is ", accuracy)

confusion_matrix(Y_TEST, testing_data_prediction)

import matplotlib.pyplot as plt
cm = confusion_matrix(Y_TEST, testing_data_prediction)
fig, ax = plt.subplots(figsize=(7,7))
ax.imshow(cm)
ax.xaxis.set(ticks=(0,1,2), ticklabels=('Predicted 0s','Predicted 2s',
'Predicted 3s'))
ax.yaxis.set(ticks=(0,1,2), ticklabels=('Actual 0s','Actual 2s',
'Actual 3s'))
for i in range(3):
    for j in range(3):
        ax.text(j, i, cm[i, j], ha='center', va='center',
color='white')
plt.show()

report = classification_report(Y_TEST, testing_data_prediction)
print(report)
```