2023AIML573_FE_Project

July 21, 2024

FEATURE ENGINEERING End-to End PROJECT (30M)

AIML Certification Programme

0.1 Student Name and ID:

Mention your name and ID if done individually If done as a group,clearly mention the contribution from each group member qualitatively and as a precentage. 1. Preethi Carmel Bosco ID 2023AIML573

0.2 Business Understanding (1M)

Students are expected to identify a regression problem of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?What are the different sources of data?

1. What is the business problem that you are trying to solve?
   The BUsiness problem we are trying to solve is ”Bringing predictability to housing prices”. Currently with out any housing price predictor it is difficult for both buyer and seller to rightly price the house. The rationale behind a particular price set fo rthe house is also not clear to understand. Since price of the house is a big investment for most buyer they will like to get a probable price based on current and past market trend. Also this prediction will help buyers understand
   How much the house can be later sold at,
   What is the Year on year appreciation in price,
   Which localities appreciate most
   What is the mortage value of a property
   List of options that can raise the property price

2. What data do you need to answer the above problem?What are the different sources of data?
   we will need housing price data with various feature and dimensions like SalePrice,Location,Utilities,Neighborhood,Condition,number of bed rooms and bath rooms The dataset is a housing dataset presented by De Cock (2011). The data came to him directly from the Ames City Assessor’s Office in the form of a data dump from their records system. The original Excel file contained 113 variables describing 3970 property sales that had occurred in Ames, Iowa between 2006 and 2010. However, so that the dataset could

be used as a “layman’s” data set that could be easily understood by users at all levels he removed any variables that required special knowledge or previous calculations for their use. Most of these deleted variables were related to weighting and adjustment factors used in the city’s current modelling system.

0.3 Data Requirements and Data Collection (3+1M)

In the initial data collection stage, data scientists identify and gather the available data resources. These can be in the form of structured, unstructured, and even semi-structured data relevant to the problem domain.

Identify the required data that fulfills the data requirements stage of the data science methodology Mention the source of the data.(Give the link if you have sourced it from any public data set) Briefly explain the data set identified .

Based on the business use case and requirement we can us ethe public kaggle data set of ames housing prices https://www.kaggle.com/c/house-prices-advanced-regression-techniques/

The dataset contains 2930 records (rows) and 82 features (columns) which will be used to predict our target column which is Sales Price i.e the amount the apartment or house sell for considering different conditions.

Import the above data and read it into a data frame

[266]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[267]:
```
# load the csv file in a pandas data frame using function read_csv
Housing_df = pd.read_csv('train.csv')
Housing_df1 = Housing_df # copy of df to work with
```

Confirm the data has been correctly by displaying the first 5 and last 5 records.

[268]:
```
# print first 5 records of dataframe Housing_df1
print(Housing_df1.head(5))
# print last 5 records of dataframe Housing_df1
print(Housing_df1.tail(5))
```

```
   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

  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
2          Lvl     AllPub  ...        0     NaN   NaN         NaN       0      9
3          Lvl     AllPub  ...        0     NaN   NaN         NaN       0      2
4          Lvl     AllPub  ...        0     NaN   NaN         NaN       0     12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]
        Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
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
1460  1461          20       RL         68.0     9717   Pave   NaN      Reg
1461  1462          20       RL         75.0     9937   Pave   NaN      Reg

     LandContour Utilities  ... PoolArea PoolQC  Fence MiscFeature MiscVal  \
1457         Lvl    AllPub  ...        0    NaN  GdPrv        Shed    2500
1458         Lvl    AllPub  ...        0    NaN    NaN         NaN       0
1459         Lvl    AllPub  ...        0    NaN    NaN         NaN       0
1460         Lvl    AllPub  ...        0    NaN    NaN         NaN       0
1461         Lvl    AllPub  ...        0    NaN    NaN         NaN       0

     MoSold YrSold  SaleType  SaleCondition  SalePrice
1457      5   2010        WD         Normal     266500
1458      4   2010        WD         Normal     142125
1459      6   2008        WD         Normal     147500
1460      4   2010        WD         Normal     142125
1461      6   2008        WD         Normal     147500

[5 rows x 81 columns]
```

Get the dimensions of the dataframe.

[269]:
```
print(Housing_df1.shape)
```

```
(1462, 81)
```

[270]:
```
# print only column heads
Col_list = Housing_df1.columns
print(Col_list)
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
```

```
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

There are 1460 records with 81 columns.

Display the description and statistical summary of the data.

[271]:
```python
for col in Housing_df1.columns:
    print(f"\n{col} \n")
    print(Housing_df1[col].describe())
```

```
Id

count    1462.00000
mean      731.50000
std       422.18736
min         1.00000
25%       366.25000
50%       731.50000
75%      1096.75000
max      1462.00000
Name: Id, dtype: float64

MSSubClass

count    1462.000000
mean       56.846785
std        42.293616
min        20.000000
25%        20.000000
50%        50.000000
75%        70.000000
max       190.000000
Name: MSSubClass, dtype: float64
```

MSZoning

```
count     1462
unique       5
top         RL
freq      1153
Name: MSZoning, dtype: object
```

LotFrontage

```
count    1203.000000
mean       70.052369
std        24.265032
min        21.000000
25%        59.000000
50%        69.000000
75%        80.000000
max       313.000000
Name: LotFrontage, dtype: float64
```

LotArea

```
count      1462.000000
mean      10515.884405
std        9974.464229
min        1300.000000
25%        7558.500000
50%        9485.000000
75%       11600.000000
max      215245.000000
Name: LotArea, dtype: float64
```

Street

```
count     1462
unique       2
top       Pave
freq      1456
Name: Street, dtype: object
```

Alley

```
count       91
unique       2
top       Grvl
freq        50
Name: Alley, dtype: object
```

```
LotShape

count     1462
unique       4
top        Reg
freq       927
Name: LotShape, dtype: object

LandContour

count     1462
unique       4
top        Lvl
freq      1313
Name: LandContour, dtype: object

Utilities

count       1462
unique         2
top       AllPub
freq        1461
Name: Utilities, dtype: object

LotConfig

count       1462
unique         5
top       Inside
freq        1054
Name: LotConfig, dtype: object

LandSlope

count     1462
unique       3
top        Gtl
freq      1384
Name: LandSlope, dtype: object

Neighborhood

count      1462
unique       25
top       NAmes
freq        226
Name: Neighborhood, dtype: object
```

```
Condition1

count     1462
unique       9
top       Norm
freq      1262
Name: Condition1, dtype: object

Condition2

count     1462
unique       8
top       Norm
freq      1447
Name: Condition2, dtype: object

BldgType

count     1462
unique       5
top       1Fam
freq      1222
Name: BldgType, dtype: object

HouseStyle

count       1462
unique         8
top       1Story
freq         728
Name: HouseStyle, dtype: object

OverallQual

count    1462.000000
mean        6.097811
std         1.382647
min         1.000000
25%         5.000000
50%         6.000000
75%         7.000000
max        10.000000
Name: OverallQual, dtype: float64

OverallCond

count    1462.000000
```

```
mean        5.575923
std         1.112148
min         1.000000
25%         5.000000
50%         5.000000
75%         6.000000
max         9.000000
Name: OverallCond, dtype: float64

YearBuilt

count    1462.000000
mean     1971.248974
std        30.187792
min      1872.000000
25%      1954.000000
50%      1972.500000
75%      2000.000000
max      2010.000000
Name: YearBuilt, dtype: float64

YearRemodAdd

count    1462.000000
mean     1984.859781
std        20.639871
min      1950.000000
25%      1967.000000
50%      1994.000000
75%      2004.000000
max      2010.000000
Name: YearRemodAdd, dtype: float64

RoofStyle

count      1462
unique        6
top       Gable
freq       1142
Name: RoofStyle, dtype: object

RoofMatl

count        1462
unique          8
top       CompShg
freq         1436
Name: RoofMatl, dtype: object
```

Exterior1st

```
count        1462
unique         15
top       VinylSd
freq          515
Name: Exterior1st, dtype: object
```

Exterior2nd

```
count        1462
unique         16
top       VinylSd
freq          504
Name: Exterior2nd, dtype: object
```

MasVnrType

```
count         588
unique          3
top       BrkFace
freq          445
Name: MasVnrType, dtype: object
```

MasVnrArea

```
count    1454.000000
mean      103.542641
std       180.982379
min         0.000000
25%         0.000000
50%         0.000000
75%       165.750000
max      1600.000000
Name: MasVnrArea, dtype: float64
```

ExterQual

```
count     1462
unique       4
top         TA
freq       907
Name: ExterQual, dtype: object
```

ExterCond

```
count     1462
```

```
unique       5
top         TA
freq      1284
Name: ExterCond, dtype: object

Foundation

count      1462
unique        6
top       PConc
freq        647
Name: Foundation, dtype: object

BsmtQual

count     1425
unique       4
top         TA
freq       651
Name: BsmtQual, dtype: object

BsmtCond

count     1425
unique       4
top         TA
freq      1313
Name: BsmtCond, dtype: object

BsmtExposure

count     1424
unique       4
top         No
freq       954
Name: BsmtExposure, dtype: object

BsmtFinType1

count     1425
unique       6
top        Unf
freq       430
Name: BsmtFinType1, dtype: object

BsmtFinSF1

count    1462.000000
```

```
mean      443.634063
std       456.014768
min         0.000000
25%         0.000000
50%       383.500000
75%       712.750000
max      5644.000000
Name: BsmtFinSF1, dtype: float64

BsmtFinType2

count     1424
unique       6
top        Unf
freq      1256
Name: BsmtFinType2, dtype: object

BsmtFinSF2

count    1462.000000
mean       47.387825
std       163.367059
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max      1474.000000
Name: BsmtFinSF2, dtype: float64

BsmtUnfSF

count    1462.000000
mean      566.557456
std       441.957220
min         0.000000
25%       221.500000
50%       475.000000
75%       808.000000
max      2336.000000
Name: BsmtUnfSF, dtype: float64

TotalBsmtSF

count    1462.000000
mean     1057.579343
std       438.436028
min         0.000000
25%       796.000000
```

```
50%       992.000000
75%      1297.750000
max      6110.000000
Name: TotalBsmtSF, dtype: float64
```

Heating

```
count     1462
unique       6
top       GasA
freq      1430
Name: Heating, dtype: object
```

HeatingQC

```
count     1462
unique       5
top         Ex
freq       741
Name: HeatingQC, dtype: object
```

CentralAir

```
count     1462
unique       2
top          Y
freq      1367
Name: CentralAir, dtype: object
```

Electrical

```
count      1461
unique        5
top       SBrkr
freq       1335
Name: Electrical, dtype: object
```

1stFlrSF

```
count    1462.000000
mean     1162.632695
std       386.337110
min       334.000000
25%       882.000000
50%      1087.000000
75%      1391.000000
max      4692.000000
Name: 1stFlrSF, dtype: float64
```

```
2ndFlrSF

count    1462.000000
mean      346.517784
std       436.418166
min         0.000000
25%         0.000000
50%         0.000000
75%       728.000000
max      2065.000000
Name: 2ndFlrSF, dtype: float64

LowQualFinSF

count    1462.000000
mean        5.836525
std        48.590270
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       572.000000
Name: LowQualFinSF, dtype: float64

GrLivArea

count    1462.000000
mean     1514.987004
std       525.288942
min       334.000000
25%      1128.500000
50%      1461.500000
75%      1776.000000
max      5642.000000
Name: GrLivArea, dtype: float64

BsmtFullBath

count    1462.000000
mean        0.426129
std         0.518990
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         3.000000
Name: BsmtFullBath, dtype: float64
```

```
BsmtHalfBath

count    1462.000000
mean        0.057456
std         0.238599
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         2.000000
Name: BsmtHalfBath, dtype: float64

FullBath

count    1462.000000
mean        1.564295
std         0.550935
min         0.000000
25%         1.000000
50%         2.000000
75%         2.000000
max         3.000000
Name: FullBath, dtype: float64

HalfBath

count    1462.000000
mean        0.383037
std         0.502900
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         2.000000
Name: HalfBath, dtype: float64

BedroomAbvGr

count    1462.000000
mean        2.865937
std         0.815542
min         0.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         8.000000
Name: BedroomAbvGr, dtype: float64
```

```
KitchenAbvGr

count    1462.000000
mean        1.046512
std         0.220194
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         3.000000
Name: KitchenAbvGr, dtype: float64

KitchenQual

count     1462
unique       4
top         TA
freq       736
Name: KitchenQual, dtype: object

TotRmsAbvGrd

count    1462.000000
mean        6.516416
std         1.624822
min         2.000000
25%         5.000000
50%         6.000000
75%         7.000000
max        14.000000
Name: TotRmsAbvGrd, dtype: float64

Functional

count     1462
unique       7
top        Typ
freq      1362
Name: Functional, dtype: object

Fireplaces

count    1462.000000
mean        0.612175
std         0.644624
min         0.000000
25%         0.000000
```

```
50%         1.000000
75%         1.000000
max         3.000000
Name: Fireplaces, dtype: float64
```

FireplaceQu

```
count     770
unique      5
top        Gd
freq      380
Name: FireplaceQu, dtype: object
```

GarageType

```
count       1381
unique         6
top       Attchd
freq         872
Name: GarageType, dtype: object
```

GarageYrBlt

```
count    1381.000000
mean     1978.475742
std        24.686416
min      1900.000000
25%      1961.000000
50%      1980.000000
75%      2002.000000
max      2010.000000
Name: GarageYrBlt, dtype: float64
```

GarageFinish

```
count     1381
unique       3
top        Unf
freq       606
Name: GarageFinish, dtype: object
```

GarageCars

```
count    1462.000000
mean        1.766074
std         0.747342
min         0.000000
25%         1.000000
```

```
50%          2.000000
75%          2.000000
max          4.000000
Name: GarageCars, dtype: float64

GarageArea

count    1462.000000
mean      472.686047
std       213.807290
min         0.000000
25%       328.500000
50%       479.500000
75%       576.000000
max      1418.000000
Name: GarageArea, dtype: float64

GarageQual

count     1381
unique       5
top         TA
freq      1313
Name: GarageQual, dtype: object

GarageCond

count     1381
unique       5
top         TA
freq      1328
Name: GarageCond, dtype: object

PavedDrive

count     1462
unique       3
top          Y
freq      1342
Name: PavedDrive, dtype: object

WoodDeckSF

count    1462.000000
mean       94.869357
std       126.571566
min         0.000000
25%         0.000000
```

```
50%        0.000000
75%      168.000000
max      857.000000
Name: WoodDeckSF, dtype: float64
```

OpenPorchSF

```
count    1462.000000
mean       46.642955
std        66.224266
min         0.000000
25%         0.000000
50%        25.000000
75%        68.000000
max       547.000000
Name: OpenPorchSF, dtype: float64
```

EnclosedPorch

```
count    1462.000000
mean       22.000684
std        61.125397
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       552.000000
Name: EnclosedPorch, dtype: float64
```

3SsnPorch

```
count    1462.000000
mean        3.404925
std        29.297528
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       508.000000
Name: 3SsnPorch, dtype: float64
```

ScreenPorch

```
count    1462.000000
mean       15.040356
std        55.722021
min         0.000000
25%         0.000000
```

```
50%          0.000000
75%          0.000000
max        480.000000
Name: ScreenPorch, dtype: float64

PoolArea

count    1462.000000
mean        2.755130
std        40.149927
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max       738.000000
Name: PoolArea, dtype: float64

PoolQC

count      7
unique     3
top       Gd
freq       3
Name: PoolQC, dtype: object

Fence

count       281
unique        4
top       MnPrv
freq        157
Name: Fence, dtype: object

MiscFeature

count       54
unique       4
top       Shed
freq        49
Name: MiscFeature, dtype: object

MiscVal

count     1462.000000
mean        43.429549
std        495.785938
min          0.000000
25%          0.000000
```

```
50%           0.000000
75%           0.000000
max       15500.000000
Name: MiscVal, dtype: float64
```

```
MoSold
```

```
count    1462.000000
mean        6.320109
std         2.702470
min         1.000000
25%         5.000000
50%         6.000000
75%         8.000000
max        12.000000
Name: MoSold, dtype: float64
```

```
YrSold
```

```
count    1462.000000
mean     2007.817373
std         1.328423
min      2006.000000
25%      2007.000000
50%      2008.000000
75%      2009.000000
max      2010.000000
Name: YrSold, dtype: float64
```

```
SaleType
```

```
count     1462
unique       9
top         WD
freq      1269
Name: SaleType, dtype: object
```

```
SaleCondition
```

```
count       1462
unique         6
top       Normal
freq        1200
Name: SaleCondition, dtype: object
```

```
SalePrice
```

```
count      1462.000000
```

```
mean     180871.799590
std       79399.396259
min       34900.000000
25%      130000.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

Dropping the column ID

[272]:
```python
Housing_df1 = Housing_df1.drop('Id', axis=1)
```

Display the columns and their respective data types.

[273]:
```python
import types
#for col in Housing_df1.columns:
    #print(f"{col}        {type(Housing_df1[col][1])} \n") # index 1 to get␣
 ↪sample data and its type
categorial_columns=[]
Numerical_columns=[]
for i in Housing_df1.iloc[1].index:
    if isinstance(Housing_df1[i].values[0],(np.floating,np.integer)):
        Numerical_columns.append(i)
    else:
        categorial_columns.append(i)
print (len(categorial_columns))
print (len(Numerical_columns))
```

```
43
37
```

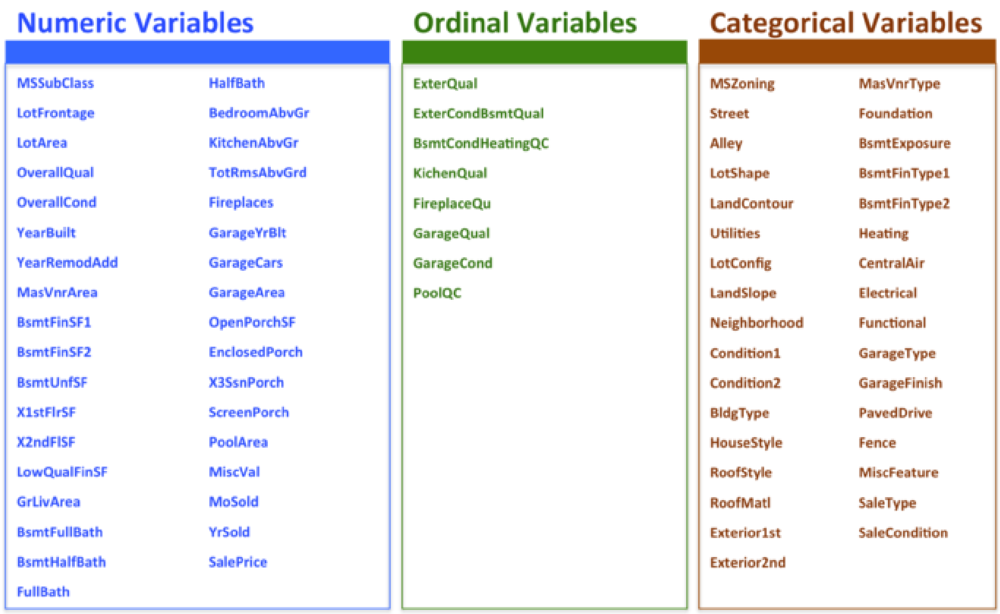
[274]:
```python
#discrete or ordinal?
discrete_columns=[]
continous_columns=[]
date_columns=[]

#for i in Numerical_columns:# discrete and continous
    ##if len(Housing_df1[i].unique())<15: # we can analyse them for nominal or␣
 ↪ordinal
        #print(f"\n {i}\n")
        #print(Housing_df1[i].unique())
ordinal_columns =['PoolQC','GarageQual', 'GarageCond',␣
 ↪'KitchenQual','ExterQual','ExterCond',␣
 ↪'BsmtQual','BsmtCond','HeatingQC','OverallQual', 'OverallCond','FireplaceQu']

print(len(Numerical_columns))
print(len(categorial_columns))
```

```
## figuring ordinal columns
for i in ordinal_columns:
    if i in Numerical_columns:
        Numerical_columns.remove(i)
    if i in categorial_columns:
        categorial_columns.remove(i)



##figuring out continous and discrete columns

for i in Numerical_columns:
    #print(type(Housing_df1[i].iloc[1]))
    if isinstance(Housing_df1[i].iloc[1], np.floating):
        continous_columns.append(i)
    elif isinstance(Housing_df1[i].iloc[1], np.integer):
        discrete_columns.append(i)
    else:
        print("wronglg classified as numeric")
        print(i)
        print(type(Housing_df1[i].iloc[1]))

## figuring date columns
for i in Housing_df1.columns:

    if ("mo".upper() in i.upper()) or  ("yr".upper() in i.upper()) or  ("year".
 ↪upper() in i.upper()):
        date_columns.append(i)
        if i in Numerical_columns:
            Numerical_columns.remove(i)
        if i in discrete_columns:
            discrete_columns.remove(i)
        if i in categorial_columns:
            categorial_columns.remove(i)
        if i in continous_columns:
            continous_columns.remove(i)
        if i in ordinal_columns:
            ordinal_columns.remove(i)
print("categorial_columns ")
print(len(categorial_columns))
print(categorial_columns)
print("Numerical_columns")
print(len(Numerical_columns))
print((Numerical_columns))
print("discrete_columns")
print(len(discrete_columns))
print((discrete_columns))
```

```python
print("continous_columns")
print(len(continous_columns))
print(continous_columns)
print("date_columns")
print(len(date_columns))
print(date_columns)
print("ordinal_columns")
print(len(ordinal_columns))
print((ordinal_columns))
```

```
37
43
categorial_columns
33
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType', 'Foundation', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
'Heating', 'CentralAir', 'Electrical', 'Functional', 'GarageType',
'GarageFinish', 'PavedDrive', 'Fence', 'MiscFeature', 'SaleType',
'SaleCondition']
Numerical_columns
30
['MSSubClass', 'LotFrontage', 'LotArea', 'MasVnrArea', 'BsmtFinSF1',
'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch',
'3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'SalePrice']
discrete_columns
28
['MSSubClass', 'LotArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea',
'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch',
'PoolArea', 'MiscVal', 'SalePrice']
continous_columns
2
['LotFrontage', 'MasVnrArea']
date_columns
5
['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'MoSold', 'YrSold']
ordinal_columns
12
['PoolQC', 'GarageQual', 'GarageCond', 'KitchenQual', 'ExterQual', 'ExterCond',
'BsmtQual', 'BsmtCond', 'HeatingQC', 'OverallQual', 'OverallCond',
```

```
'FireplaceQu']
```



Other than dates,OverallQual,OverallCond other attributes are numerical

[ ]:
```

```

from analysis we see that other than OverallCond,OverallQual and date attributes, all other attribute sare discrete nummerical.

[275]:
```
Housing_df1.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1462 entries, 0 to 1461
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1462 non-null   int64
 1   MSZoning       1462 non-null   object
 2   LotFrontage    1203 non-null   float64
 3   LotArea        1462 non-null   int64
 4   Street         1462 non-null   object
 5   Alley          91 non-null     object
 6   LotShape       1462 non-null   object
 7   LandContour    1462 non-null   object
 8   Utilities      1462 non-null   object
 9   LotConfig      1462 non-null   object
 10  LandSlope      1462 non-null   object
 11  Neighborhood   1462 non-null   object
 12  Condition1     1462 non-null   object
```

```
13  Condition2     1462 non-null   object
14  BldgType       1462 non-null   object
15  HouseStyle     1462 non-null   object
16  OverallQual    1462 non-null   int64
17  OverallCond    1462 non-null   int64
18  YearBuilt      1462 non-null   int64
19  YearRemodAdd   1462 non-null   int64
20  RoofStyle      1462 non-null   object
21  RoofMatl       1462 non-null   object
22  Exterior1st    1462 non-null   object
23  Exterior2nd    1462 non-null   object
24  MasVnrType     588 non-null    object
25  MasVnrArea     1454 non-null   float64
26  ExterQual      1462 non-null   object
27  ExterCond      1462 non-null   object
28  Foundation     1462 non-null   object
29  BsmtQual       1425 non-null   object
30  BsmtCond       1425 non-null   object
31  BsmtExposure   1424 non-null   object
32  BsmtFinType1   1425 non-null   object
33  BsmtFinSF1     1462 non-null   int64
34  BsmtFinType2   1424 non-null   object
35  BsmtFinSF2     1462 non-null   int64
36  BsmtUnfSF      1462 non-null   int64
37  TotalBsmtSF    1462 non-null   int64
38  Heating        1462 non-null   object
39  HeatingQC      1462 non-null   object
40  CentralAir     1462 non-null   object
41  Electrical     1461 non-null   object
42  1stFlrSF       1462 non-null   int64
43  2ndFlrSF       1462 non-null   int64
44  LowQualFinSF   1462 non-null   int64
45  GrLivArea      1462 non-null   int64
46  BsmtFullBath   1462 non-null   int64
47  BsmtHalfBath   1462 non-null   int64
48  FullBath       1462 non-null   int64
49  HalfBath       1462 non-null   int64
50  BedroomAbvGr   1462 non-null   int64
51  KitchenAbvGr   1462 non-null   int64
52  KitchenQual    1462 non-null   object
53  TotRmsAbvGrd   1462 non-null   int64
54  Functional     1462 non-null   object
55  Fireplaces     1462 non-null   int64
56  FireplaceQu    770 non-null    object
57  GarageType     1381 non-null   object
58  GarageYrBlt    1381 non-null   float64
59  GarageFinish   1381 non-null   object
60  GarageCars     1462 non-null   int64
```

```
 61  GarageArea     1462 non-null   int64
 62  GarageQual     1381 non-null   object
 63  GarageCond     1381 non-null   object
 64  PavedDrive     1462 non-null   object
 65  WoodDeckSF     1462 non-null   int64
 66  OpenPorchSF    1462 non-null   int64
 67  EnclosedPorch  1462 non-null   int64
 68  3SsnPorch      1462 non-null   int64
 69  ScreenPorch    1462 non-null   int64
 70  PoolArea       1462 non-null   int64
 71  PoolQC         7 non-null      object
 72  Fence          281 non-null    object
 73  MiscFeature    54 non-null     object
 74  MiscVal        1462 non-null   int64
 75  MoSold         1462 non-null   int64
 76  YrSold         1462 non-null   int64
 77  SaleType       1462 non-null   object
 78  SaleCondition  1462 non-null   object
 79  SalePrice      1462 non-null   int64
dtypes: float64(3), int64(34), object(43)
memory usage: 3.9 MB
```

Convert the columns to appropriate data types

The dataset has appropiate data types set already, Encoding and column derivation willbe done after clean up

Write your observations from the above. From the above we can find the data type of the features We have multiple values missing values which has to handled after analysis. The data further needs to be visualized to find the distribution and get more initution

0.3.1 Check for Data Quality Issues (1.5M)

- duplicate data
- missing data
- data inconsistencies

[276]:
```
# Missing data count
missing_data = Housing_df1.isnull().sum()
missing_percentage = (missing_data / len(Housing_df1)) * 100
missing_info = pd.DataFrame({'Missing Values': missing_data, 'Percentage':␣
↪missing_percentage})
print(missing_info)
print(missing_info[missing_info['Missing Values'] > 0])
```

```
               Missing Values  Percentage
MSSubClass                  0    0.000000
MSZoning                    0    0.000000
```

```
LotFrontage              259   17.715458
LotArea                    0    0.000000
Street                     0    0.000000
…                        …           …
MoSold                     0    0.000000
YrSold                     0    0.000000
SaleType                   0    0.000000
SaleCondition              0    0.000000
SalePrice                  0    0.000000

[80 rows x 2 columns]
              Missing Values  Percentage
LotFrontage              259   17.715458
Alley                   1371   93.775650
MasVnrType               874   59.781122
MasVnrArea                 8    0.547196
BsmtQual                  37    2.530780
BsmtCond                  37    2.530780
BsmtExposure              38    2.599179
BsmtFinType1              37    2.530780
BsmtFinType2              38    2.599179
Electrical                 1    0.068399
FireplaceQu              692   47.332421
GarageType                81    5.540356
GarageYrBlt               81    5.540356
GarageFinish              81    5.540356
GarageQual                81    5.540356
GarageCond                81    5.540356
PoolQC                  1455   99.521204
Fence                   1181   80.779754
MiscFeature             1408   96.306430
```

[277]:

```python
# checking for duplicate records with same value for YrSold, LotArea, SalePrice
# Creating a DataFrame object
df_temp = pd.DataFrame(Housing_df1[['YrSold', 'LotArea',␣
 ↪'SalePrice','YearBuilt','GrLivArea']])

# Selecting duplicate rows except first
# occurrence based on all columns
duplicate = df_temp[df_temp.duplicated()]

print("Duplicate Rows :")
duplicate
```

```
Duplicate Rows :
```

[277]:

	YrSold	LotArea	SalePrice	YearBuilt	GrLivArea
193	2006	2522	130000	2004	1709
1460	2010	9717	142125	1950	1078
1461	2008	9937	147500	1965	1256

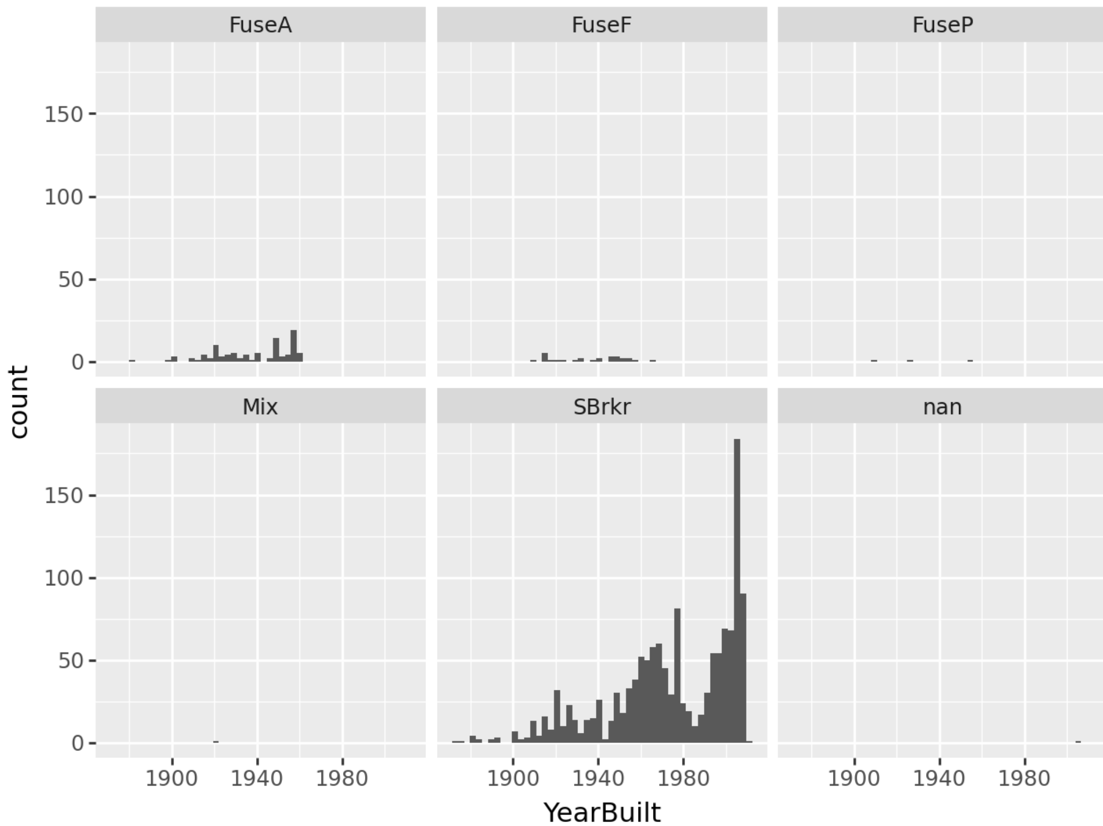
Data inconsistency

Fromdocumentation we understand that following inconsistency are expected null values Alley means no alley Bsmt* means no basement FireplaceQu means no fireplace Garage* means no garage expect missing values for GarageYrBlt as its same as house year built PoolQC meansno pool Fence means no fence MiscFeature means no such item as an elevator, tennis court, second garage, etc...

[278]:
```
print (missing_info[missing_info['Missing Values'] > 0].index)
```

```
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Electrical', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence',
       'MiscFeature'],
      dtype='object')
```

[279]:
```
from plotnine import ggplot, aes, geom_line,geom_histogram,facet_wrap
( ggplot(Housing_df1, aes("YearBuilt"))
   + geom_histogram(bins = 50)
   + facet_wrap("Electrical")
)
```



the above plot it seems safe to impute null values in Electrical as ”SBrkr” seeing as that type is dominant over the time period

0.3.2 Handling the data quality issues(1.5M)

Apply techniques * to remove duplicate data * to impute or remove missing data * to remove data inconsistencies Give detailed explanation for each column how you handle the data quality issues.

dropping duplicate data

[280]:
```
i = [193,1460,1461]
Housing_df1.drop(i,inplace=True)
print(Housing_df1.shape)
```

```
(1459, 80)
```

Remove data inconsistency and impute data

[281]:
```
import warnings
warnings.filterwarnings('ignore')
```

MODE AND MEAN IMPUTATION

[282]:
```python
# imputing the categorial variable values with mode .this willcause teh current␣
↪skew to be emphasised.
#imputing numericals with mean.
Alley_impute= Housing_df1['Alley'].mode()
BsmtQual_impute= Housing_df1['BsmtQual'].mode()
BsmtCond_impute= Housing_df1['BsmtCond'].mode()
BsmtExposure_impute= Housing_df1['BsmtExposure'].mode()
BsmtFinType1_impute= Housing_df1['BsmtFinType1'].mode()
BsmtFinType2_impute= Housing_df1['BsmtFinType1'].mode()
FireplaceQu_impute= Housing_df1['FireplaceQu'].mode()
GarageType_impute= Housing_df1['GarageType'].mode()
GarageFinish_impute= Housing_df1['GarageFinish'].mode()
GarageQual_impute= Housing_df1['GarageQual'].mode()
GarageCond_impute= Housing_df1['GarageCond'].mode()
PoolQC_impute= Housing_df1['PoolQC'].mode()
Fence_impute= Housing_df1['Fence'].mode()
MiscFeature_impute= Housing_df1['MiscFeature'].mode()
Electrical_impute= Housing_df1['Electrical'].mode()
MasVnrType_impute= Housing_df1['MasVnrType'].mode()
MasVnrArea_impute= Housing_df1['MasVnrArea'].mean()
LotFrontage_impute= Housing_df1['LotFrontage'].mean()
#for garage buit fill with year buit
Housing_df1['GarageYrBlt'].fillna(Housing_df1['YearBuilt'],inplace=True)
```

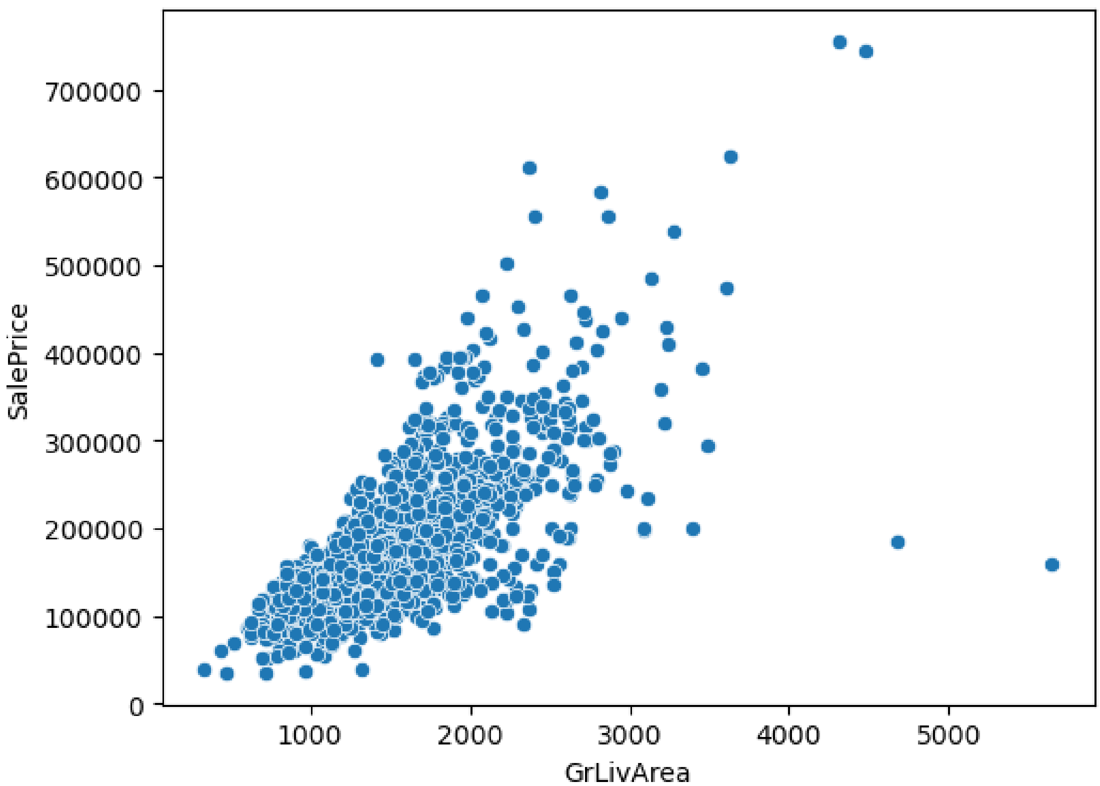
[283]:
```python
# setting with nullalternative values
Housing_df1.Alley.fillna(Alley_impute.values[0],inplace=True) #
Housing_df1.BsmtQual.fillna(BsmtQual_impute.values[0],inplace=True)
Housing_df1.BsmtCond.fillna(BsmtCond_impute.values[0],inplace=True)
Housing_df1.BsmtExposure.fillna(BsmtExposure_impute.values[0],inplace=True)
Housing_df1.BsmtFinType1.fillna(BsmtFinType1_impute.values[0],inplace=True)
Housing_df1.BsmtFinType2.fillna(BsmtFinType2_impute.values[0],inplace=True)
Housing_df1.FireplaceQu.fillna(FireplaceQu_impute.values[0],inplace=True)
Housing_df1.GarageType.fillna(GarageType_impute.values[0],inplace=True)
Housing_df1.GarageFinish.fillna(GarageFinish_impute.values[0],inplace=True)
Housing_df1.GarageQual.fillna(GarageQual_impute.values[0],inplace=True)
Housing_df1.GarageCond.fillna(GarageCond_impute.values[0],inplace=True)
Housing_df1.PoolQC.fillna(PoolQC_impute.values[0],inplace=True)
Housing_df1.Fence.fillna(Fence_impute.values[0],inplace=True)
Housing_df1.MiscFeature.fillna(MiscFeature_impute.values[0],inplace=True)
Housing_df1.Electrical.fillna(Electrical_impute.values[0],inplace=True)
Housing_df1.MasVnrType.fillna(MasVnrType_impute.values[0],inplace=True)
Housing_df1.MasVnrArea.fillna(MasVnrArea_impute,inplace=True)
Housing_df1.LotFrontage.fillna(LotFrontage_impute,inplace=True)
#Housing_df1.GarageYrBlt.fillna(GarageYrBlt_impute,inplace=True)
```

Chceking for presence of null values

[284]:
```
#Housing_df1 = fillwithnull(Housing_df1)
# Missing data count
missing_data = Housing_df1.isnull().sum()
missing_percentage = (missing_data / len(Housing_df1)) * 100
missing_info = pd.DataFrame({'Missing Values': missing_data, 'Percentage':␣
↪missing_percentage})
print(missing_info[missing_info['Missing Values'] > 0])
#print(Housing_df1.isnull().sum().sort_values(ascending=False))
```

```
Empty DataFrame
Columns: [Missing Values, Percentage]
Index: []
```

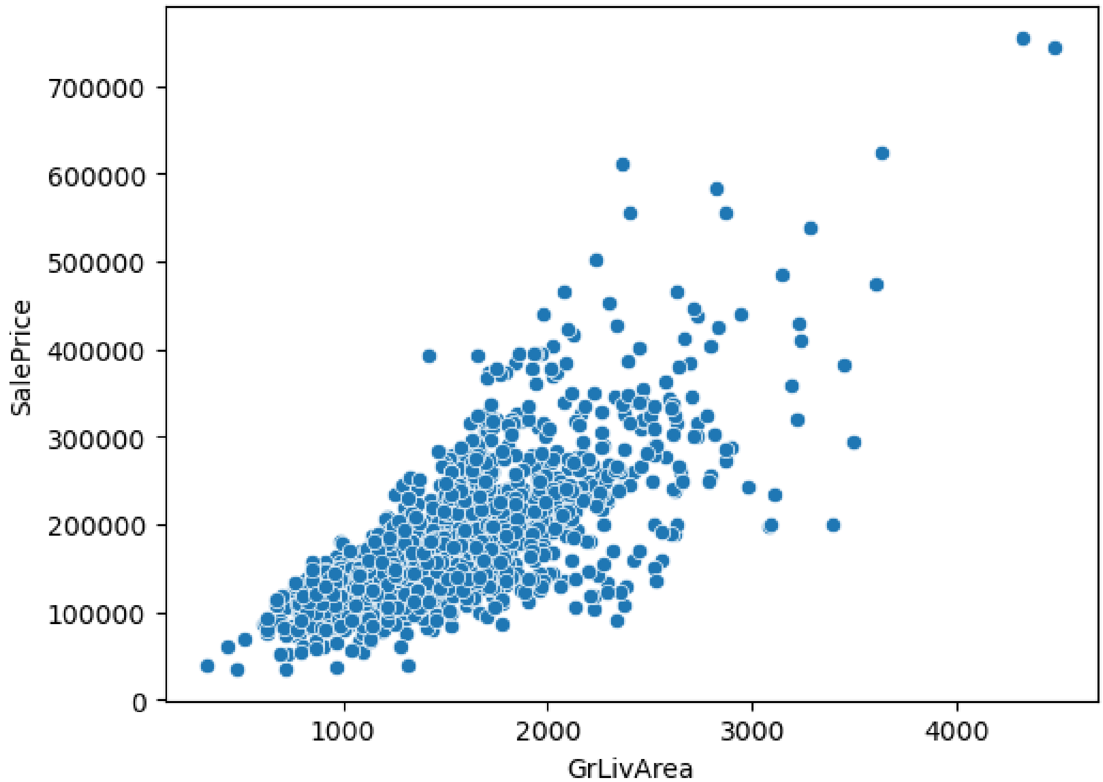
[285]:
```
sns.scatterplot(data=Housing_df1,x="GrLivArea",y="SalePrice")
plt.show()
```



deleting outliers records from data set

[286]:
```
outliers = ((Housing_df1.GrLivArea > 4000) & (Housing_df1.SalePrice < 5E5))
Housing_df1 =  Housing_df1[~(outliers)]
Housing_dfTest =Housing_df1
sns.scatterplot(data=Housing_df1,x="GrLivArea",y="SalePrice")
```

[286]: <Axes: xlabel='GrLivArea', ylabel='SalePrice'>



[ ]:

0.3.3 Normalise the data wherever necessary(1M)

Normalization refers to the process of transforming features in a dataset to a specific range. This range can be different depending on the chosen normalization technique. The two most common normalization techniques are Min-Max Scaling and Z-Score Normalization, which is also called Standardization

MIN MAX scaling normalisation of numerical attributes

[287]:
```
print(discrete_columns)
print(Housing_df1[discrete_columns])
```

```
['MSSubClass', 'LotArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea',
'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch',
'PoolArea', 'MiscVal', 'SalePrice']
```

	MSSubClass	LotArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	\
0	60	8450	706	0	150	856	
1	20	9600	978	0	284	1262	
2	60	11250	486	0	434	920	
3	70	9550	216	0	540	756	
4	60	14260	655	0	490	1145	
…	…	…	…	…	…	…	
1455	60	7917	0	0	953	953	
1456	20	13175	790	163	589	1542	
1457	70	9042	275	0	877	1152	
1458	20	9717	49	1029	0	1078	
1459	20	9937	830	290	136	1256	

	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	…	GarageCars	\
0	856	854	0	1710	…	2	
1	1262	0	0	1262	…	2	
2	920	866	0	1786	…	2	
3	961	756	0	1717	…	3	
4	1145	1053	0	2198	…	3	
…	…	…	…	…	…	…	
1455	953	694	0	1647	…	2	
1456	2073	0	0	2073	…	2	
1457	1188	1152	0	2340	…	1	
1458	1078	0	0	1078	…	1	
1459	1256	0	0	1256	…	1	

	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	\
0	548	0	61	0	0	
1	460	298	0	0	0	
2	608	0	42	0	0	
3	642	0	35	272	0	
4	836	192	84	0	0	
…	…	…	…	…	…	
1455	460	0	40	0	0	
1456	500	349	0	0	0	
1457	252	0	60	0	0	
1458	240	366	0	112	0	
1459	276	736	68	0	0	

	ScreenPorch	PoolArea	MiscVal	SalePrice
0	0	0	0	208500
1	0	0	0	181500
2	0	0	0	223500
3	0	0	0	140000
4	0	0	0	250000
…	…	…	…	…
1455	0	0	0	175000
1456	0	0	0	210000

```
1457            0          0      2500      266500
1458            0          0         0      142125
1459            0          0         0      147500

[1457 rows x 28 columns]
```

[288]:

```python
print(Housing_df1[discrete_columns])
for i in discrete_columns:
    max_i = Housing_df1[i].max()
    min_i = Housing_df1[i].min()
    Range_i = max_i -min_i
    scaled_i = (Housing_df1[i] - min_i)/Range_i
    Housing_df1[i] = scaled_i
print(Housing_df1[discrete_columns])
```

```
      MSSubClass  LotArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  \
0             60     8450         706           0        150          856
1             20     9600         978           0        284         1262
2             60    11250         486           0        434          920
3             70     9550         216           0        540          756
4             60    14260         655           0        490         1145
…          …       …           …           …         …          …
1455          60     7917           0           0        953          953
1456          20    13175         790         163        589         1542
1457          70     9042         275           0        877         1152
1458          20     9717          49        1029          0         1078
1459          20     9937         830         290        136         1256

      1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  …  GarageCars  \
0          856       854             0       1710  …           2
1         1262         0             0       1262  …           2
2          920       866             0       1786  …           2
3          961       756             0       1717  …           3
4         1145      1053             0       2198  …           3
…        …       …           …          …  …        …
1455       953       694             0       1647  …           2
1456      2073         0             0       2073  …           2
1457      1188      1152             0       2340  …           1
1458      1078         0             0       1078  …           1
1459      1256         0             0       1256  …           1

      GarageArea  WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  \
0            548           0           61              0          0
1            460         298            0              0          0
2            608           0           42              0          0
3            642           0           35            272          0
4            836         192           84              0          0
…          …         …          …            …          …
```

```
1455        460         0          40             0           0
1456        500       349           0             0           0
1457        252         0          60             0           0
1458        240       366           0           112           0
1459        276       736          68             0           0

      ScreenPorch  PoolArea  MiscVal  SalePrice
0               0         0        0     208500
1               0         0        0     181500
2               0         0        0     223500
3               0         0        0     140000
4               0         0        0     250000
…             …       …      …        …
1455            0         0        0     175000
1456            0         0        0     210000
1457            0         0     2500     266500
1458            0         0        0     142125
1459            0         0        0     147500

[1457 rows x 28 columns]
      MSSubClass   LotArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  \
0       0.235294  0.033420    0.322669    0.000000   0.064212     0.266999
1       0.000000  0.038795    0.446984    0.000000   0.121575     0.393637
2       0.235294  0.046507    0.222121    0.000000   0.185788     0.286962
3       0.294118  0.038561    0.098720    0.000000   0.231164     0.235808
4       0.235294  0.060576    0.299360    0.000000   0.209760     0.357143
…            …        …         …          …         …          …
1455    0.235294  0.030929    0.000000    0.000000   0.407962     0.297255
1456    0.000000  0.055505    0.361060    0.110583   0.252140     0.480973
1457    0.294118  0.036187    0.125686    0.000000   0.375428     0.359326
1458    0.000000  0.039342    0.022395    0.698100   0.000000     0.336245
1459    0.000000  0.040370    0.379342    0.196744   0.058219     0.391765

      1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  …  GarageCars  \
0     0.180373  0.413559           0.0   0.332207  …        0.50
1     0.320663  0.000000           0.0   0.224046  …        0.50
2     0.202488  0.419370           0.0   0.350555  …        0.50
3     0.216655  0.366102           0.0   0.333897  …        0.75
4     0.280235  0.509927           0.0   0.450024  …        0.75
…        …        …            …         …  …         …
1455  0.213891  0.336077           0.0   0.316997  …        0.50
1456  0.600898  0.000000           0.0   0.419845  …        0.50
1457  0.295093  0.557869           0.0   0.484307  …        0.25
1458  0.257084  0.000000           0.0   0.179623  …        0.25
1459  0.318590  0.000000           0.0   0.222598  …        0.25

      GarageArea  WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  \
0       0.394245    0.000000     0.111517       0.000000        0.0
```

```
1       0.330935   0.347725   0.000000     0.000000      0.0
2       0.437410   0.000000   0.076782     0.000000      0.0
3       0.461871   0.000000   0.063985     0.492754      0.0
4       0.601439   0.224037   0.153565     0.000000      0.0
…            …        …          …            …         …
1455    0.330935   0.000000   0.073126     0.000000      0.0
1456    0.359712   0.407235   0.000000     0.000000      0.0
1457    0.181295   0.000000   0.109689     0.000000      0.0
1458    0.172662   0.427071   0.000000     0.202899      0.0
1459    0.198561   0.858810   0.124314     0.000000      0.0

      ScreenPorch  PoolArea  MiscVal  SalePrice
0             0.0       0.0  0.00000   0.241078
1             0.0       0.0  0.00000   0.203583
2             0.0       0.0  0.00000   0.261908
3             0.0       0.0  0.00000   0.145952
4             0.0       0.0  0.00000   0.298709
…             …        …        …          …
1455          0.0       0.0  0.00000   0.194556
1456          0.0       0.0  0.00000   0.243161
1457          0.0       0.0  0.16129   0.321622
1458          0.0       0.0  0.00000   0.148903
1459          0.0       0.0  0.00000   0.156367

[1457 rows x 28 columns]
```

0.3.4 Standardise the data (1M)

Standardization is the process of transforming data into a common format which you to make the meaningful comparison.

Standardization is a preprocessing step that's commonly applied to numerical features in machine learning. The goal of standardization is to transform the feature values so that they have a mean of 0 and a standard deviation of 1.

[289]:
```python
print(Housing_df1[discrete_columns])
from sklearn.preprocessing import StandardScaler
Housing_df1[discrete_columns] = StandardScaler().
 ↪fit_transform(Housing_df1[discrete_columns])
print(Housing_df1[discrete_columns])
```

```
      MSSubClass   LotArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  \
0       0.235294  0.033420    0.322669    0.000000   0.064212     0.266999
1       0.000000  0.038795    0.446984    0.000000   0.121575     0.393637
2       0.235294  0.046507    0.222121    0.000000   0.185788     0.286962
3       0.294118  0.038561    0.098720    0.000000   0.231164     0.235808
4       0.235294  0.060576    0.299360    0.000000   0.209760     0.357143
…            …        …          …           …          …           …
```

```
1455    0.235294  0.030929    0.000000    0.000000   0.407962     0.297255
1456    0.000000  0.055505    0.361060    0.110583   0.252140     0.480973
1457    0.294118  0.036187    0.125686    0.000000   0.375428     0.359326
1458    0.000000  0.039342    0.022395    0.698100   0.000000     0.336245
1459    0.000000  0.040370    0.379342    0.196744   0.058219     0.391765

      1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  …  GarageCars  \
0     0.180373  0.413559           0.0   0.332207  …        0.50
1     0.320663  0.000000           0.0   0.224046  …        0.50
2     0.202488  0.419370           0.0   0.350555  …        0.50
3     0.216655  0.366102           0.0   0.333897  …        0.75
4     0.280235  0.509927           0.0   0.450024  …        0.75
…          …         …           …          …  …         …
1455  0.213891  0.336077           0.0   0.316997  …        0.50
1456  0.600898  0.000000           0.0   0.419845  …        0.50
1457  0.295093  0.557869           0.0   0.484307  …        0.25
1458  0.257084  0.000000           0.0   0.179623  …        0.25
1459  0.318590  0.000000           0.0   0.222598  …        0.25

      GarageArea  WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  \
0       0.394245    0.000000     0.111517       0.000000        0.0
1       0.330935    0.347725     0.000000       0.000000        0.0
2       0.437410    0.000000     0.076782       0.000000        0.0
3       0.461871    0.000000     0.063985       0.492754        0.0
4       0.601439    0.224037     0.153565       0.000000        0.0
…            …           …            …              …          …
1455    0.330935    0.000000     0.073126       0.000000        0.0
1456    0.359712    0.407235     0.000000       0.000000        0.0
1457    0.181295    0.000000     0.109689       0.000000        0.0
1458    0.172662    0.427071     0.000000       0.202899        0.0
1459    0.198561    0.858810     0.124314       0.000000        0.0

      ScreenPorch  PoolArea  MiscVal  SalePrice
0             0.0       0.0  0.00000   0.241078
1             0.0       0.0  0.00000   0.203583
2             0.0       0.0  0.00000   0.261908
3             0.0       0.0  0.00000   0.145952
4             0.0       0.0  0.00000   0.298709
…             …         …        …          …
1455          0.0       0.0  0.00000   0.194556
1456          0.0       0.0  0.00000   0.243161
1457          0.0       0.0  0.16129   0.321622
1458          0.0       0.0  0.00000   0.148903
1459          0.0       0.0  0.00000   0.156367

[1457 rows x 28 columns]
      MSSubClass   LotArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF  TotalBsmtSF  \
0       0.075226 -0.204462    0.616593   -0.288975  -0.943115    -0.473746
```

1	-0.871675	-0.087794	1.245035	-0.288975	-0.639921	0.504622
2	0.075226	0.079600	0.108295	-0.288975	-0.300525	-0.319521
3	0.311951	-0.092866	-0.515527	-0.288975	-0.060684	-0.714724
4	0.075226	0.384966	0.498760	-0.288975	-0.173817	0.222678
…	…	…	…	…	…	…
1455	0.075226	-0.258535	-1.014584	-0.288975	0.873787	-0.239998
1456	-0.871675	0.274892	0.810671	0.720838	0.050185	1.179359
1457	0.311951	-0.144403	-0.379210	-0.288975	0.701826	0.239547
1458	-0.871675	-0.075924	-0.901372	6.085854	-1.282512	0.061223
1459	-0.871675	-0.053605	0.903089	1.507624	-0.974792	0.490163

	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	…	GarageCars	\
0	-0.814450	1.168173	-0.120367	0.393167	…	0.313277	
1	0.276928	-0.793684	-0.120367	-0.488980	…	0.313277	
2	-0.642410	1.195740	-0.120367	0.542817	…	0.313277	
3	-0.532197	0.943042	-0.120367	0.406950	…	1.651823	
4	-0.037582	1.625327	-0.120367	1.354077	…	1.651823	
…	…	…	…	…	…	…	
1455	-0.553702	0.800612	-0.120367	0.269115	…	0.313277	
1456	2.456997	-0.793684	-0.120367	1.107942	…	0.313277	
1457	0.078007	1.852755	-0.120367	1.633686	…	-1.025269	
1458	-0.217686	-0.793684	-0.120367	-0.851290	…	-1.025269	
1459	0.260800	-0.793684	-0.120367	-0.500795	…	-1.025269	

	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	\
0	0.357576	-0.751234	0.225839	-0.359742	-0.116461	
1	-0.057077	1.626569	-0.708129	-0.359742	-0.116461	
2	0.640294	-0.751234	-0.065069	-0.359742	-0.116461	
3	0.800501	-0.751234	-0.172246	4.088121	-0.116461	
4	1.714623	0.780774	0.577991	-0.359742	-0.116461	
…	…	…	…	…	…	
1455	-0.057077	-0.751234	-0.095691	-0.359742	-0.116461	
1456	0.131401	2.033509	-0.708129	-0.359742	-0.116461	
1457	-1.037167	-0.751234	0.210528	-0.359742	-0.116461	
1458	-1.093710	2.169155	-0.708129	1.471731	-0.116461	
1459	-0.924079	5.121461	0.333016	-0.359742	-0.116461	

	ScreenPorch	PoolArea	MiscVal	SalePrice
0	-0.270507	-0.063731	-0.087779	0.346386
1	-0.270507	-0.063731	-0.087779	0.006695
2	-0.270507	-0.063731	-0.087779	0.535104
3	-0.270507	-0.063731	-0.087779	-0.515424
4	-0.270507	-0.063731	-0.087779	0.868505
…	…	…	…	…
1455	-0.270507	-0.063731	-0.087779	-0.075083
1456	-0.270507	-0.063731	-0.087779	0.365258
1457	-0.270507	-0.063731	4.947879	1.076094
1458	-0.270507	-0.063731	-0.087779	-0.488689

```
1459    -0.270507 -0.063731 -0.087779  -0.421065

[1457 rows x 28 columns]
```

0.3.5 Perform Binning (1M)

Binning is a process of transforming continuous numerical variables into discrete categorical 'bins', for grouped analysis.

[290]:
```
#print(Housing_df1[continous_columns])
```

[291]:
```
# using unsepervised binning of equal width
min =Housing_df1['LotFrontage'].min()
max =Housing_df1['LotFrontage'].max()
range_temp=( max - min)
bin =[min,min+(range_temp*0.25),min+(range_temp*0.5),min+(0.75*range_temp),max]
print(bin)
labels = ['Small', 'Medium', 'Large', 'Verylarge']
Housing_df1['LotFrontage_bin'] = pd.cut(Housing_df1['LotFrontage'],␣
  ↪bin,labels=labels)
print(Housing_df1['LotFrontage_bin'].unique())
print(Housing_df1['LotFrontage_bin'].isna().sum())
```

```
[21.0, 94.0, 167.0, 240.0, 313.0]
['Small', 'Medium', NaN, 'Large', 'Verylarge']
Categories (4, object): ['Small' < 'Medium' < 'Large' < 'Verylarge']
23
```

0.3.6 Perform Data Discretization(2M)

[292]:
```
# using unsepervised binning of equal width
min =Housing_df1['MasVnrArea'].min()
max =Housing_df1['MasVnrArea'].max()
range_temp=( max - min)
bin =[min,min+(range_temp*0.25),min+(range_temp*0.5),min+(0.75*range_temp),max]
print(bin)
labels = ['Small', 'Medium', 'Large', 'Verylarge']
Housing_df1['MasVnrArea_bin'] = pd.cut(Housing_df1['MasVnrArea'],␣
  ↪bin,labels=labels)
print(Housing_df1['MasVnrArea_bin'].isna().sum())
nan_rows = Housing_df1['MasVnrArea_bin'].isna()
#print(nan_rows)
#for i, (nan_rows_i) in enumerate(nan_rows):
#    if(nan_rows_i):
#        print(Housing_df1[['MasVnrArea','MasVnrArea_bin']].iloc[i])
```

```
[0.0, 400.0, 800.0, 1200.0, 1600.0]
861
```

We get NAN bin for 0 value , Sincebinnning is exclusive for lower range. so imputing it to small bin

[293]:
```python
Housing_df1['MasVnrArea_bin'].fillna("Small",inplace=True)
Housing_df1['LotFrontage_bin'].fillna("Small",inplace=True)
print(Housing_df1['LotFrontage_bin'].unique())
print(Housing_df1['MasVnrArea_bin'].unique())
# adding these columns to our list of categorical columns
categorial_columns.append('MasVnrArea_bin');
categorial_columns.append('LotFrontage_bin');
```

```
['Small', 'Medium', 'Large', 'Verylarge']
Categories (4, object): ['Small' < 'Medium' < 'Large' < 'Verylarge']
['Small', 'Medium', 'Large', 'Verylarge']
Categories (4, object): ['Small' < 'Medium' < 'Large' < 'Verylarge']
```

0.3.7 Perform encoding (1M)

[294]:
```python
from sklearn.preprocessing import OneHotEncoder

print("Categorical variables:")
print(categorial_columns)
print('No. of. categorical features: ',
      len(categorial_columns))

ohe = OneHotEncoder(handle_unknown='ignore',sparse_output=False).
 ↪set_output(transform='pandas')
for i in categorial_columns:
      ohe_done = ohe.fit_transform(Housing_df1[[i]])
      Housing_df1 = pd.concat([Housing_df1,ohe_done],axis =1)
Housing_df1=Housing_df1.drop(categorial_columns,axis=1)
print(Housing_df1.columns)
```

```
Categorical variables:
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType', 'Foundation', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
'Heating', 'CentralAir', 'Electrical', 'Functional', 'GarageType',
'GarageFinish', 'PavedDrive', 'Fence', 'MiscFeature', 'SaleType',
'SaleCondition', 'MasVnrArea_bin', 'LotFrontage_bin']
No. of. categorical features:  35
Index(['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond',
       'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'ExterQual', 'ExterCond',
       …
```

```
        'SaleCondition_Normal', 'SaleCondition_Partial', 'MasVnrArea_bin_Large',
        'MasVnrArea_bin_Medium', 'MasVnrArea_bin_Small',
        'MasVnrArea_bin_Verylarge', 'LotFrontage_bin_Large',
        'LotFrontage_bin_Medium', 'LotFrontage_bin_Small',
        'LotFrontage_bin_Verylarge'],
       dtype='object', length=261)
```

[295]:
```
print(Housing_df1.head(2))
```

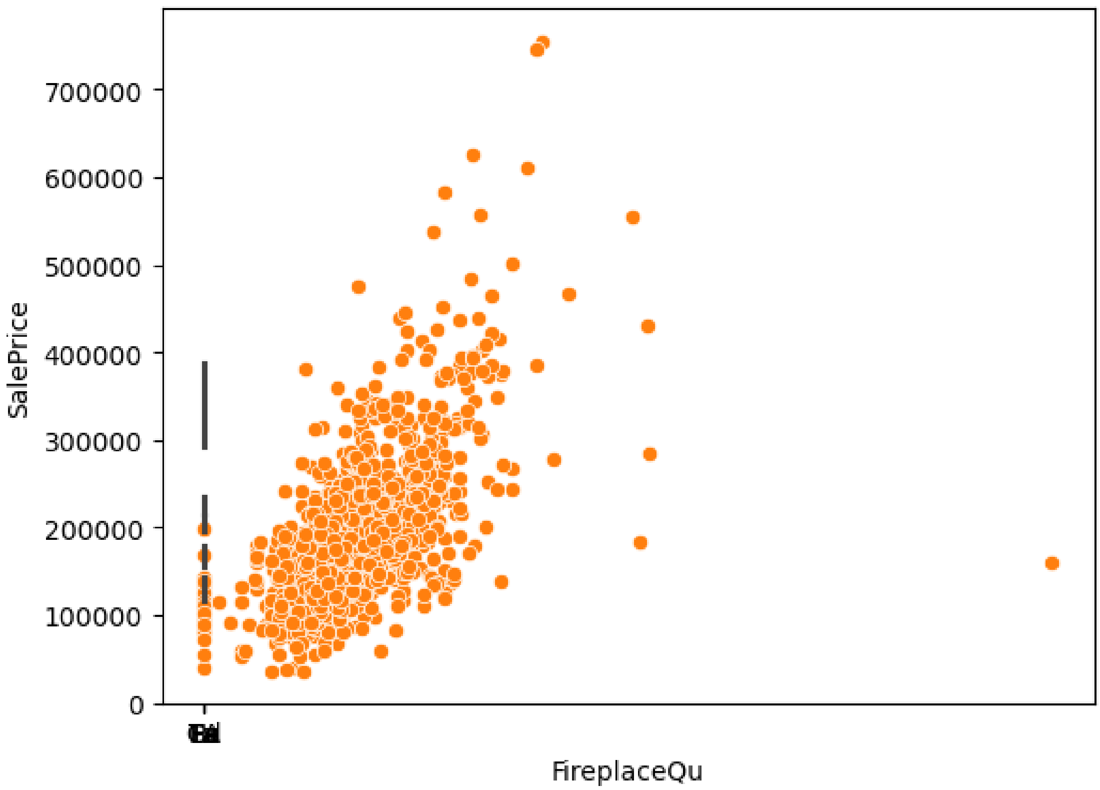
```
   MSSubClass  LotFrontage   LotArea  OverallQual  OverallCond  YearBuilt  \
0    0.075226         65.0 -0.204462            7            5       2003
1   -0.871675         80.0 -0.087794            6            8       1976

   YearRemodAdd  MasVnrArea ExterQual ExterCond  … SaleCondition_Normal  \
0          2003       196.0        Gd        TA  …                  1.0
1          1976         0.0        TA        TA  …                  1.0

  SaleCondition_Partial  MasVnrArea_bin_Large  MasVnrArea_bin_Medium  \
0                   0.0                   0.0                    0.0
1                   0.0                   0.0                    0.0

   MasVnrArea_bin_Small  MasVnrArea_bin_Verylarge LotFrontage_bin_Large  \
0                   1.0                       0.0                   0.0
1                   1.0                       0.0                   0.0

   LotFrontage_bin_Medium  LotFrontage_bin_Small  LotFrontage_bin_Verylarge
0                     0.0                    1.0                        0.0
1                     0.0                    1.0                        0.0

[2 rows x 261 columns]
```

0.3.8 EDA using Visuals(3M)

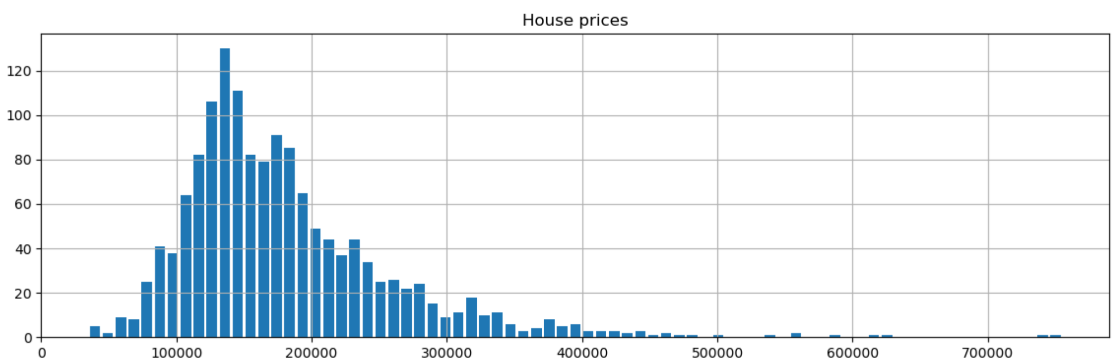
Use any 3 or more visualisation methods (Boxplot,Scatterplot,histogram,....etc) to perform Exploratory data analysis and briefly give interpretations from each visual.

[296]:
```
sns.barplot(x='FireplaceQu', y="SalePrice", data=Housing_df)
sns.scatterplot(x='TotalBsmtSF', y="SalePrice", data=Housing_df)
```

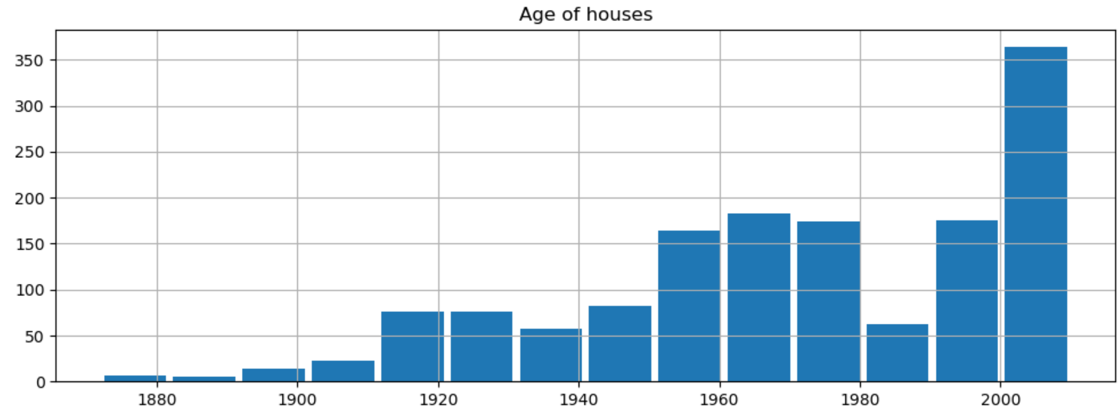
[296]: <Axes: xlabel='FireplaceQu', ylabel='SalePrice'>



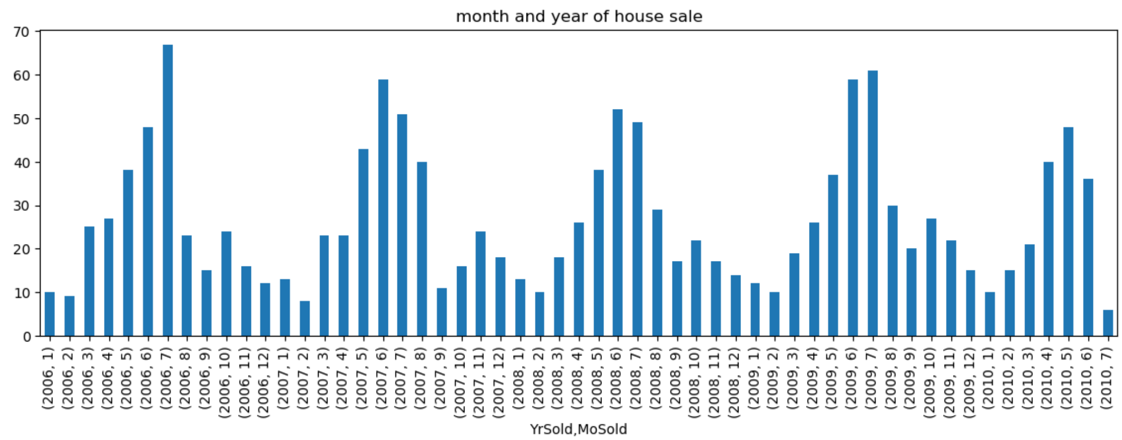
[297]:
```
Housing_df.SalePrice.hist(bins=75, rwidth=.8, figsize=(14,4))
plt.title('House prices')
plt.show()
```



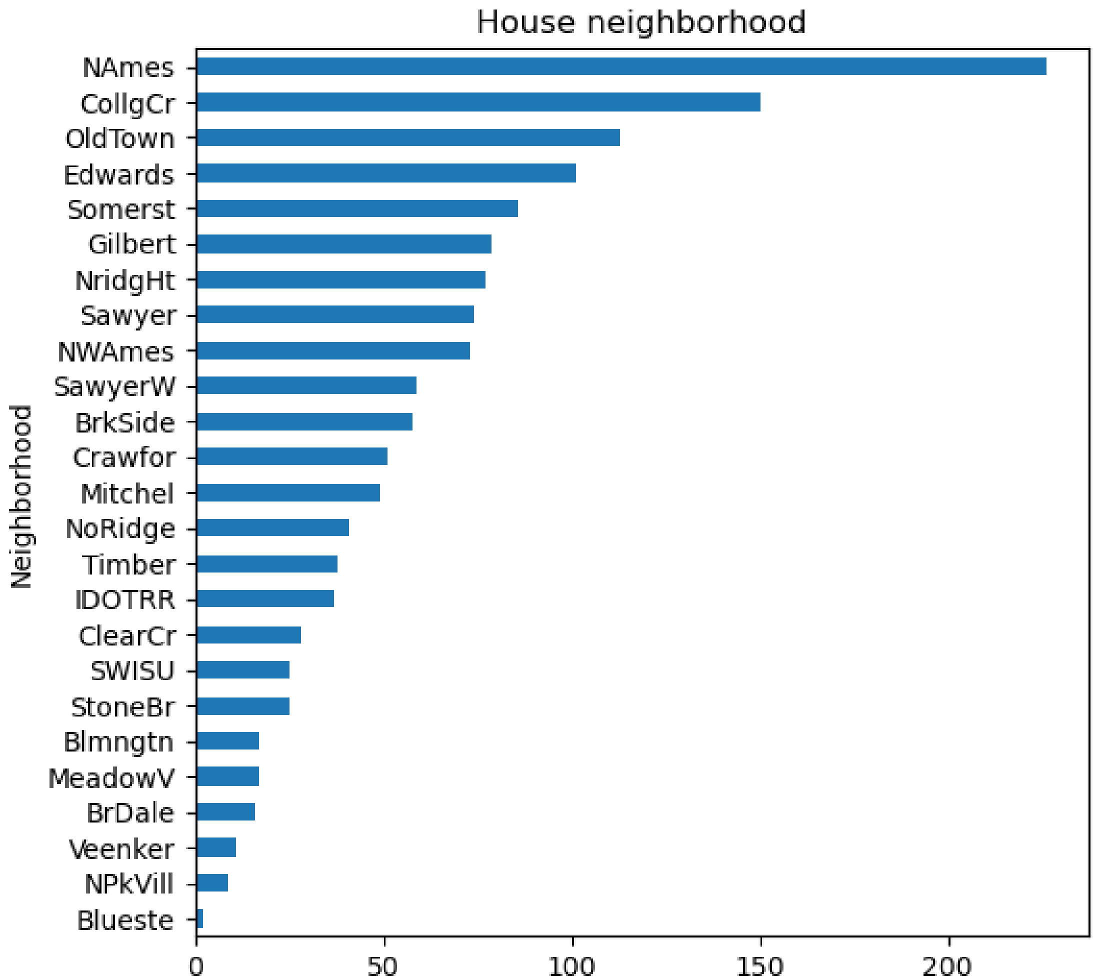
[298]:
```
Housing_df.YearBuilt.hist(bins=14, rwidth=.9, figsize=(12,4))
plt.title('Age of houses')
plt.show()
```



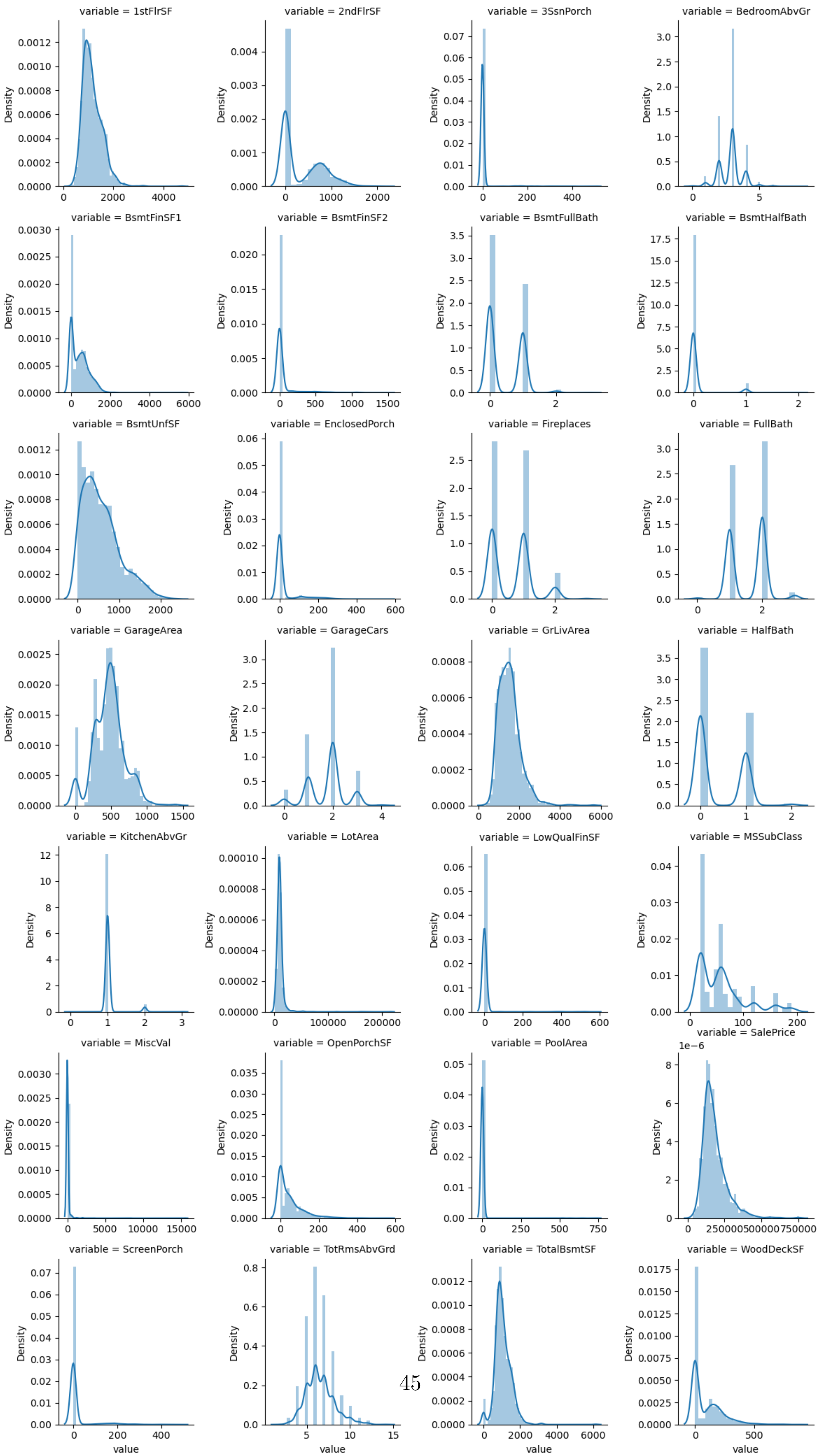
[299]:
```
Housing_df.groupby(['YrSold','MoSold']).Id.count().plot(kind='bar', figsize=(14,4))
plt.title('month and year of house sale')
plt.show()
```



[300]:
```
Housing_df.groupby('Neighborhood').Id.count().\
    sort_values().\
    plot(kind='barh', figsize=(6,6))
plt.title('House neighborhood')
plt.show()
```



[301]:
```
# Grid of distribution plots of all numerical features
f = pd.melt(Housing_df, value_vars=sorted(discrete_columns))
g = sns.FacetGrid(f, col='variable', col_wrap=4, sharex=False, sharey=False)
g = g.map(sns.distplot, 'value')
```


variable = 1stFlrSF
variable = 2ndFlrSF
variable = 3SsnPorch
variable = BedroomAbvGr
variable = BsmtFinSF1
variable = BsmtFinSF2
variable = BsmtFullBath
variable = BsmtHalfBath
variable = BsmtUnfSF
variable = EnclosedPorch
variable = Fireplaces
variable = FullBath
variable = GarageArea
variable = GarageCars
variable = GrLivArea
variable = HalfBath
variable = KitchenAbvGr
variable = LotArea
variable = LowQualFinSF
variable = MSSubClass
variable = MiscVal
variable = OpenPorchSF
variable = PoolArea
variable = SalePrice
variable = ScreenPorch
variable = TotRmsAbvGrd
variable = TotalBsmtSF
variable = WoodDeckSF
Density
value

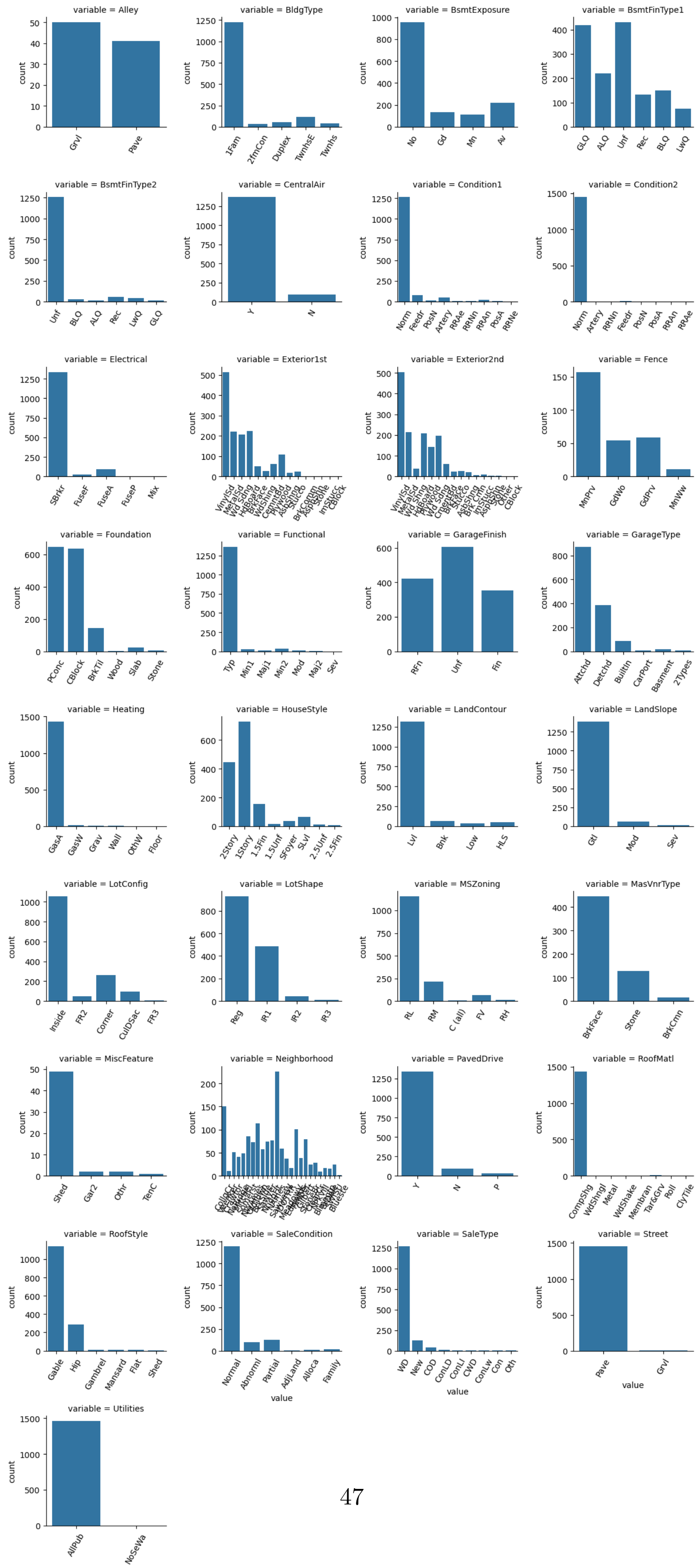
[302]:

```
print(categorial_columns)
```

```
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType', 'Foundation', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
'Heating', 'CentralAir', 'Electrical', 'Functional', 'GarageType',
'GarageFinish', 'PavedDrive', 'Fence', 'MiscFeature', 'SaleType',
'SaleCondition', 'MasVnrArea_bin', 'LotFrontage_bin']
```

[303]:

```
# Count plots of categorical features
temp =categorial_columns
temp.remove("LotFrontage_bin")
temp.remove("MasVnrArea_bin")
f = pd.melt(Housing_df, value_vars=sorted(temp))
g = sns.FacetGrid(f, col='variable', col_wrap=4, sharex=False, sharey=False)
plt.xticks(rotation='vertical')
g = g.map(sns.countplot, 'value')
[plt.setp(ax.get_xticklabels(), rotation=60) for ax in g.axes.flat]
g.fig.tight_layout()
plt.show()
```


variable = Alley
Grvl
Pave
variable = BldgType
1Fam
2fmCon
Duplex
TwnhsE
Twnhs
variable = BsmtExposure
No
Gd
Mn
Av
variable = BsmtFinType1
GLQ
ALQ
Unf
Rec
BLQ
LwQ
variable = BsmtFinType2
Unf
BLQ
ALQ
Rec
LwQ
GLQ
variable = CentralAir
Y
N
variable = Condition1
Norm
Feedr
PosN
Artery
RRAe
RRNn
RRAn
PosA
RRNe
variable = Condition2
Norm
Artery
RRNn
Feedr
PosN
PosA
RRAn
RRAe
variable = Electrical
SBrkr
FuseF
FuseA
FuseP
Mix
variable = Exterior1st
variable = Exterior2nd
variable = Fence
MnPrv
GdWo
GdPrv
MnWw
variable = Foundation
PConc
CBlock
BrkTil
Wood
Slab
Stone
variable = Functional
Typ
Min1
Maj1
Min2
Mod
Maj2
Sev
variable = GarageFinish
RFn
Unf
Fin
variable = GarageType
Attchd
Detchd
BuiltIn
CarPort
Basment
2Types
variable = Heating
GasA
GasW
Grav
Wall
OthW
Floor
variable = HouseStyle
2Story
1Story
1.5Fin
1.5Unf
SFoyer
SLvl
2.5Unf
2.5Fin
variable = LandContour
Lvl
Bnk
Low
HLS
variable = LandSlope
Gtl
Mod
Sev
variable = LotConfig
Inside
FR2
Corner
CulDSac
FR3
variable = LotShape
Reg
IR1
IR2
IR3
variable = MSZoning
RL
RM
C (all)
FV
RH
variable = MasVnrType
BrkFace
Stone
BrkCmn
variable = MiscFeature
Shed
Gar2
Othr
TenC
variable = Neighborhood
variable = PavedDrive
Y
N
P
variable = RoofMatl
CompShg
WdShngl
Metal
WdShake
Membran
Tar&Grv
Roll
ClyTile
variable = RoofStyle
Gable
Hip
Gambrel
Mansard
Flat
Shed
variable = SaleCondition
Normal
Abnorml
Partial
AdjLand
Alloca
Family
variable = SaleType
WD
New
COD
ConLD
ConLI
CWD
ConLw
Con
Oth
variable = Street
Pave
Grvl
variable = Utilities
AllPub
NoSeWa
count
value

0.3.9 Feature Selection(2M)

Apply Univariate filters identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring 1. Mutual Information (Information Gain) 2. Gini index 3. Gain Ratio 4. Chi-Squared test 5. Fisher Score (From the above 5 you are required to use any two)

0.3.10 1. Chi-Sqaured

Used for categorical features.

[304]:
```
colums= np.array(Housing_df1.columns)
New_cat =[x for x in colums if x not in discrete_columns]
New_cat =[x for x in New_cat if x not in ordinal_columns]
New_cat =[x for x in New_cat if x not in date_columns]
New_cat =[x for x in New_cat if x not in continous_columns]
print(New_cat)
print(discrete_columns)
```

```
['MSZoning_C (all)', 'MSZoning_FV', 'MSZoning_RH', 'MSZoning_RL', 'MSZoning_RM',
'Street_Grvl', 'Street_Pave', 'Alley_Grvl', 'Alley_Pave', 'LotShape_IR1',
'LotShape_IR2', 'LotShape_IR3', 'LotShape_Reg', 'LandContour_Bnk',
'LandContour_HLS', 'LandContour_Low', 'LandContour_Lvl', 'Utilities_AllPub',
'Utilities_NoSeWa', 'LotConfig_Corner', 'LotConfig_CulDSac', 'LotConfig_FR2',
'LotConfig_FR3', 'LotConfig_Inside', 'LandSlope_Gtl', 'LandSlope_Mod',
'LandSlope_Sev', 'Neighborhood_Blmngtn', 'Neighborhood_Blueste',
'Neighborhood_BrDale', 'Neighborhood_BrkSide', 'Neighborhood_ClearCr',
'Neighborhood_CollgCr', 'Neighborhood_Crawfor', 'Neighborhood_Edwards',
'Neighborhood_Gilbert', 'Neighborhood_IDOTRR', 'Neighborhood_MeadowV',
'Neighborhood_Mitchel', 'Neighborhood_NAmes', 'Neighborhood_NPkVill',
'Neighborhood_NWAmes', 'Neighborhood_NoRidge', 'Neighborhood_NridgHt',
'Neighborhood_OldTown', 'Neighborhood_SWISU', 'Neighborhood_Sawyer',
'Neighborhood_SawyerW', 'Neighborhood_Somerst', 'Neighborhood_StoneBr',
'Neighborhood_Timber', 'Neighborhood_Veenker', 'Condition1_Artery',
'Condition1_Feedr', 'Condition1_Norm', 'Condition1_PosA', 'Condition1_PosN',
'Condition1_RRAe', 'Condition1_RRAn', 'Condition1_RRNe', 'Condition1_RRNn',
'Condition2_Artery', 'Condition2_Feedr', 'Condition2_Norm', 'Condition2_PosA',
'Condition2_PosN', 'Condition2_RRAe', 'Condition2_RRAn', 'Condition2_RRNn',
'BldgType_1Fam', 'BldgType_2fmCon', 'BldgType_Duplex', 'BldgType_Twnhs',
'BldgType_TwnhsE', 'HouseStyle_1.5Fin', 'HouseStyle_1.5Unf',
'HouseStyle_1Story', 'HouseStyle_2.5Fin', 'HouseStyle_2.5Unf',
'HouseStyle_2Story', 'HouseStyle_SFoyer', 'HouseStyle_SLvl', 'RoofStyle_Flat',
'RoofStyle_Gable', 'RoofStyle_Gambrel', 'RoofStyle_Hip', 'RoofStyle_Mansard',
'RoofStyle_Shed', 'RoofMatl_CompShg', 'RoofMatl_Membran', 'RoofMatl_Metal',
```

```
'RoofMatl_Roll', 'RoofMatl_Tar&Grv', 'RoofMatl_WdShake', 'RoofMatl_WdShngl',
'Exterior1st_AsbShng', 'Exterior1st_AsphShn', 'Exterior1st_BrkComm',
'Exterior1st_BrkFace', 'Exterior1st_CBlock', 'Exterior1st_CemntBd',
'Exterior1st_HdBoard', 'Exterior1st_ImStucc', 'Exterior1st_MetalSd',
'Exterior1st_Plywood', 'Exterior1st_Stone', 'Exterior1st_Stucco',
'Exterior1st_VinylSd', 'Exterior1st_Wd Sdng', 'Exterior1st_WdShing',
'Exterior2nd_AsbShng', 'Exterior2nd_AsphShn', 'Exterior2nd_Brk Cmn',
'Exterior2nd_BrkFace', 'Exterior2nd_CBlock', 'Exterior2nd_CmentBd',
'Exterior2nd_HdBoard', 'Exterior2nd_ImStucc', 'Exterior2nd_MetalSd',
'Exterior2nd_Other', 'Exterior2nd_Plywood', 'Exterior2nd_Stone',
'Exterior2nd_Stucco', 'Exterior2nd_VinylSd', 'Exterior2nd_Wd Sdng',
'Exterior2nd_Wd Shng', 'MasVnrType_BrkCmn', 'MasVnrType_BrkFace',
'MasVnrType_Stone', 'Foundation_BrkTil', 'Foundation_CBlock',
'Foundation_PConc', 'Foundation_Slab', 'Foundation_Stone', 'Foundation_Wood',
'BsmtExposure_Av', 'BsmtExposure_Gd', 'BsmtExposure_Mn', 'BsmtExposure_No',
'BsmtFinType1_ALQ', 'BsmtFinType1_BLQ', 'BsmtFinType1_GLQ', 'BsmtFinType1_LwQ',
'BsmtFinType1_Rec', 'BsmtFinType1_Unf', 'BsmtFinType2_ALQ', 'BsmtFinType2_BLQ',
'BsmtFinType2_GLQ', 'BsmtFinType2_LwQ', 'BsmtFinType2_Rec', 'BsmtFinType2_Unf',
'Heating_Floor', 'Heating_GasA', 'Heating_GasW', 'Heating_Grav', 'Heating_OthW',
'Heating_Wall', 'CentralAir_N', 'CentralAir_Y', 'Electrical_FuseA',
'Electrical_FuseF', 'Electrical_FuseP', 'Electrical_Mix', 'Electrical_SBrkr',
'Functional_Maj1', 'Functional_Maj2', 'Functional_Min1', 'Functional_Min2',
'Functional_Mod', 'Functional_Sev', 'Functional_Typ', 'GarageType_2Types',
'GarageType_Attchd', 'GarageType_Basment', 'GarageType_BuiltIn',
'GarageType_CarPort', 'GarageType_Detchd', 'GarageFinish_Fin',
'GarageFinish_RFn', 'GarageFinish_Unf', 'PavedDrive_N', 'PavedDrive_P',
'PavedDrive_Y', 'Fence_GdPrv', 'Fence_GdWo', 'Fence_MnPrv', 'Fence_MnWw',
'MiscFeature_Gar2', 'MiscFeature_Othr', 'MiscFeature_Shed', 'MiscFeature_TenC',
'SaleType_COD', 'SaleType_CWD', 'SaleType_Con', 'SaleType_ConLD',
'SaleType_ConLI', 'SaleType_ConLw', 'SaleType_New', 'SaleType_Oth',
'SaleType_WD', 'SaleCondition_Abnorml', 'SaleCondition_AdjLand',
'SaleCondition_Alloca', 'SaleCondition_Family', 'SaleCondition_Normal',
'SaleCondition_Partial', 'MasVnrArea_bin_Large', 'MasVnrArea_bin_Medium',
'MasVnrArea_bin_Small', 'MasVnrArea_bin_Verylarge', 'LotFrontage_bin_Large',
'LotFrontage_bin_Medium', 'LotFrontage_bin_Small', 'LotFrontage_bin_Verylarge']
['MSSubClass', 'LotArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea',
'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch',
'PoolArea', 'MiscVal', 'SalePrice']
```

[305]:
```
(Housing_df1[discrete_columns] < 0).any().any()
```

[305]:
```
True
```

[306]:
```
import pandas as pd
from sklearn.feature_selection import chi2

X = Housing_df[discrete_columns]
X[X < 1] = 2 # removing zerovalues
y =Housing_df['SalePrice']

print(len(y))
chi2_scores, p_values = chi2(X,y)
print("Chi-2 Scores",chi2_scores)
print("p values",p_values)
```

```
1462
Chi-2 Scores [1.93461846e+04 1.01159718e+07 3.99055098e+05 3.67980133e+05
 2.75686880e+05 1.74605816e+05 1.23822853e+05 4.62969666e+05
 1.81676298e+05 1.97082536e+05 1.04272830e+02 1.54988014e+01
 1.85701062e+02 1.12654447e+02 1.64221595e+02 2.89155557e+01
 3.60471656e+02 1.31435497e+02 1.99543515e+02 9.62212746e+04
 1.31033491e+05 7.15128327e+04 8.99538667e+04 9.69260695e+04
 1.19789116e+05 2.21597881e+05 5.99278422e+06 5.09229737e+07]
p values [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0.
0.
 0. 0. 0. 0.]
```

Using *chi2* function removes the features that are the most likely to be independent of the target class and therefore irrelevant for classification. * Higher the value of chi2 --> More dependence

2.Information gain

[307]:
```
import pandas as pd
from sklearn.feature_selection import mutual_info_classif

X = Housing_dfTest[discrete_columns]
y=[]
#discretize the continous variablesale price
for i in Housing_dfTest['SalePrice']:
    if i>5000:
        y.append(1)
    else:
        y.append(0)


mutual_info_gain_values = mutual_info_classif(X, y,  discrete_features='auto',␣
 ↪n_neighbors=3, copy=True, random_state=None, n_jobs=None)


print("Information Gain Values",mutual_info_gain_values)
```

```
Information Gain Values [2.22044605e-16 2.22044605e-16 2.22044605e-16
```

```
2.22044605e-16
 2.22044605e-16 2.22044605e-16 2.22044605e-16 2.22044605e-16
 2.22044605e-16 2.22044605e-16 2.22044605e-16 2.22044605e-16
 3.43170899e-04 2.22044605e-16 2.22044605e-16 2.22044605e-16
 2.22044605e-16 3.43170899e-04 2.22044605e-16 2.22044605e-16
 6.86341798e-04 2.22044605e-16 3.43170899e-04 2.22044605e-16
 3.43170899e-04 2.22044605e-16 2.22044605e-16 2.22044605e-16]
```

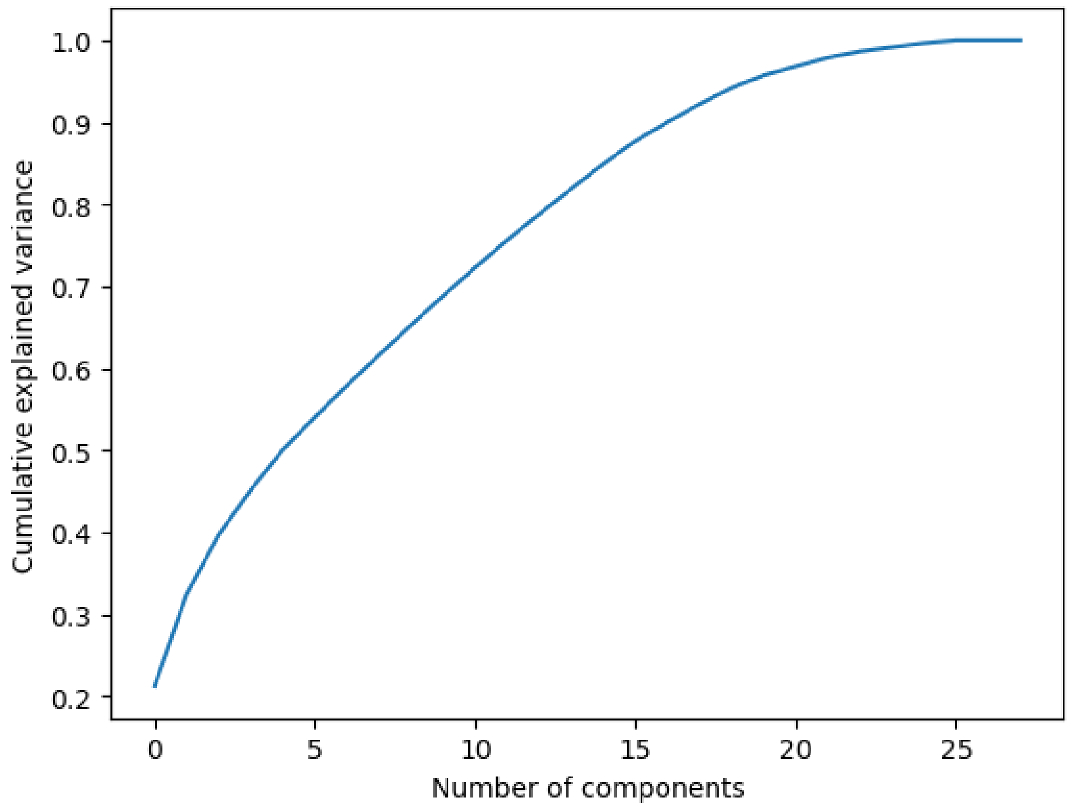
PCA

[308]:
```
print(discrete_columns)
X= Housing_df1[discrete_columns]
print(X.shape)
y=Housing_df1['SalePrice']
```

```
['MSSubClass', 'LotArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea',
'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch',
'PoolArea', 'MiscVal', 'SalePrice']
(1457, 28)
```

[309]:
```
from sklearn.decomposition import PCA
#Checking cumulative variance using PCA by applying PCA using all dimensions to␣
↪observe change in variance with number of components
pca = PCA(28)
pca_full = pca.fit(X)

plt.plot(np.cumsum(pca_full.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
```

[309]:
```
Text(0, 0.5, 'Cumulative explained variance')
```



[310]:
```
#Reducing dimensions to 3 using PCA, as there is an elbow near 3 there is a␣
↪small change in variance afterwards
pca = PCA(n_components=3)
pca.fit(X)
```

[310]: PCA(n_components=3)

[311]:
```
PCA(copy=True, iterated_power='auto', n_components=3, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

[311]: PCA(n_components=3)

[312]:
```
#Checking explained variance
print(pca.explained_variance_)
print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.cumsum())
```

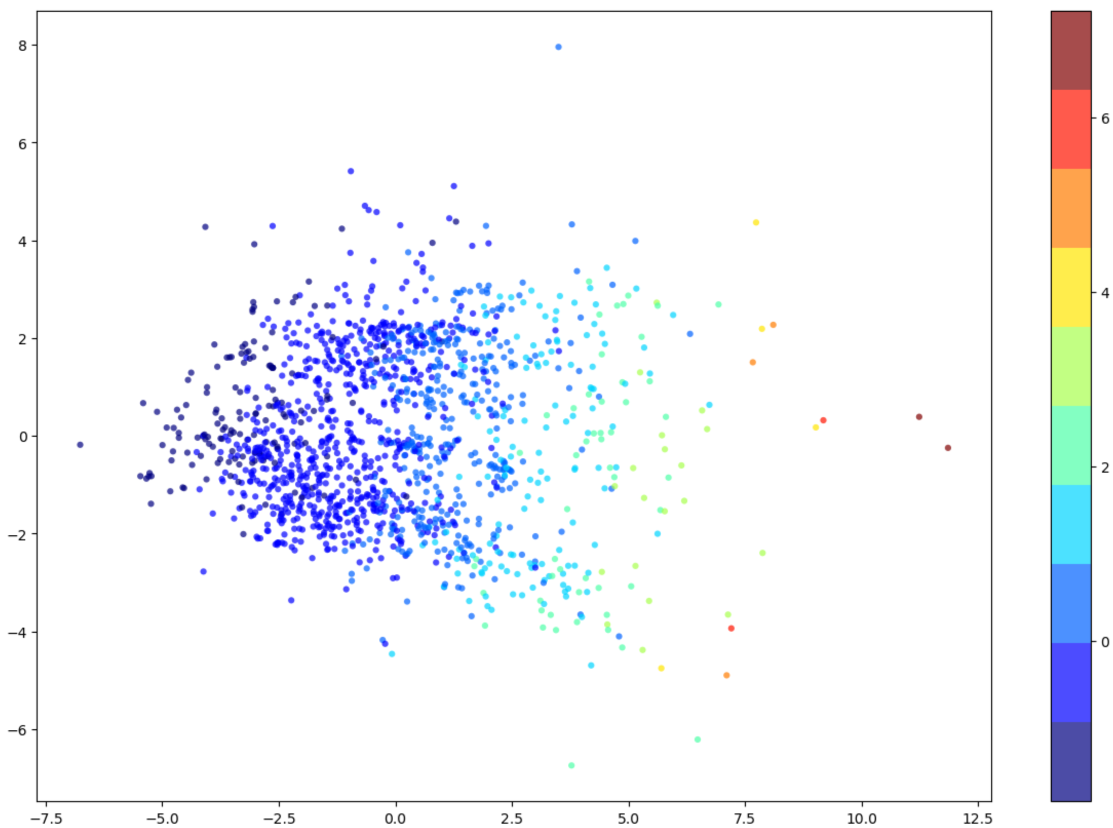
```
[5.94815891 3.13686858 2.03091299]
[0.21228844 0.11195413 0.07248282]
[0.21228844 0.32424257 0.3967254 ]
```

[313]:
```
#Reducing components further to 2 and visualize resulting principal components
pca = PCA(n_components=2)
pca = pca.fit_transform(X)
pca
```

[313]:
```
array([[ 0.73088356,  0.94873398],
       [ 0.30507899, -1.48656875],
       [ 1.00735739,  0.65057888],
       …,
       [ 1.93212298,  2.4629889 ],
       [-2.42801639, -2.18477596],
       [-0.33091317, -1.88320083]])
```

[314]:
```
plt.figure(figsize=(15,10))
plt.scatter(pca[:,0], pca[:,1], c=y, edgecolor='none', alpha=0.7,
            cmap=plt.get_cmap('jet', 10), s=20)
plt.colorbar()
```

[314]:
```
<matplotlib.colorbar.Colorbar at 0x7f35dedce3d0>
```



0.3.11 Report observations (2M)

Write your observations from the results of each of the above method(1M). Clearly justify your choice of the method.(1M)

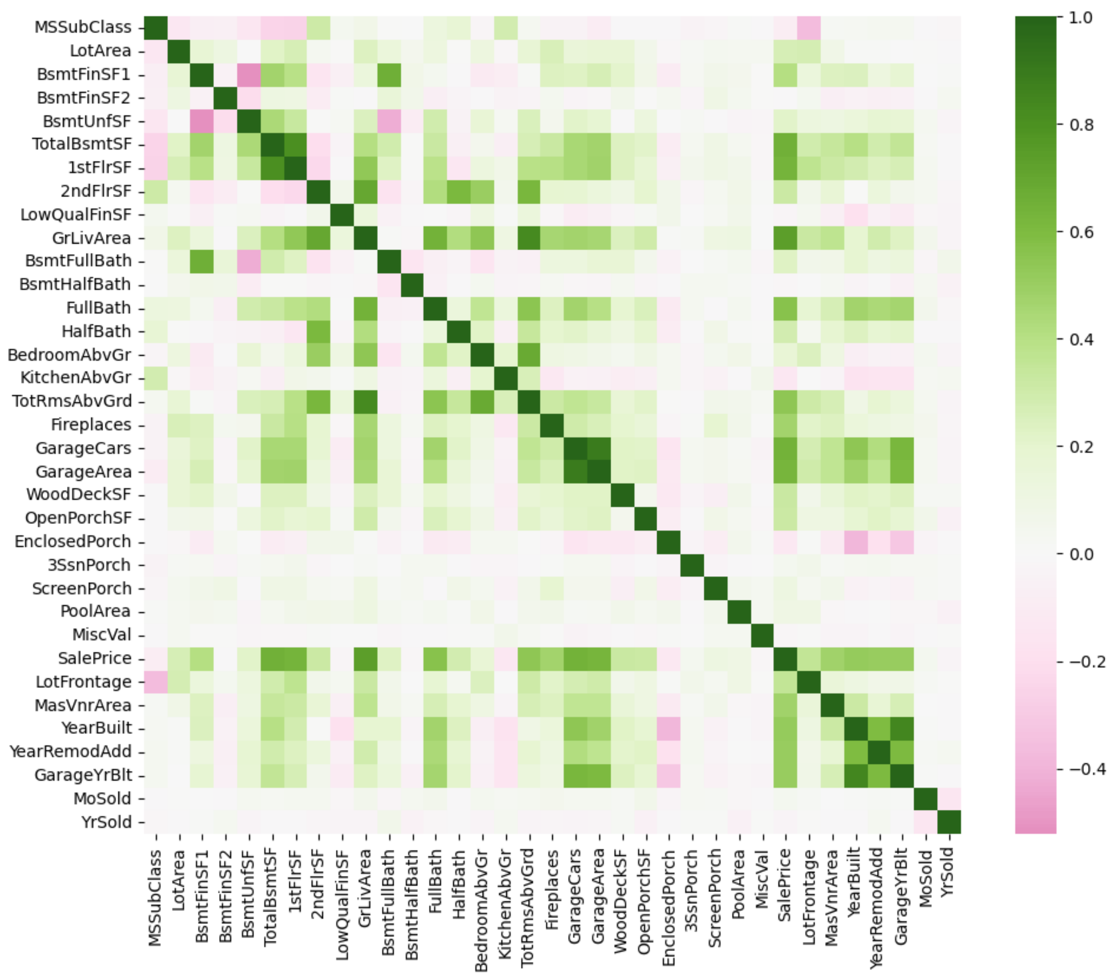
The most important columns with highest amount of information gain and chi 2 vales are ['OverallQual','TotalBsmtSF','1stFlrSF','GrLivArea','GarageCars','GarageArea','KitchenAbvGr','FullBath','YearBuilt']

0.3.12 Correlation Analysis (3 M)

Perform correlation analysis(1M) and plot the visuals(1M).Briefly explain each process,why is it used and interpret the result(1M).

[315]:
```
f, ax = plt.subplots(figsize=(12, 9))
correlation_feature = discrete_columns + continous_columns +date_columns
sns.heatmap(Housing_df1[correlation_feature].
 ↪corr(),square=True,cmap="PiYG",center=0)
```

[315]: `<Axes: >`



Strong Predictors: OverallQual TotalBsmtSF 1stFlrSF GrLivArea GarageCars GarageArea KitchenAbvGr FullBath YearBuilt

Moderate Predictors: Lot Frontage Lot Area MasVnrArea BsmtFinSF1 2ndFlrSF FullBath Fireplaces GarageYrBlt EnclosedPorch Weak Predictors: BsmtUnfSF BsmtFullBath BedroomAbvGr WoodDeckSF OpenPorchSF

[316]:
```
imp_col␣
↪=['OverallQual','TotalBsmtSF','1stFlrSF','GrLivArea','GarageCars','GarageArea','KitchenAbvG
```

0.3.13 Model Building and Prediction (4M)

Fit a linear regression model using the most important features identified(1M).Plot the visuals(1M).Briefly explain the regression model,equation (1M) and perform one prediction using the same(1M).

Regression algorithms are a subset of machine learning algorithms that predict a continuous output variable based on one or more input features. Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. Linear Regression: Linear regression is a statistical regression method which is used for predictive analysis. It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. It is used for solving the regression problem in machine learning. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression. If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression.

[317]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error
from sklearn.model_selection import train_test_split

model_LR = LinearRegression()
traindf=Housing_df1[imp_col]
X =traindf
Y =Housing_df1["SalePrice"]
X_train, X_valid, Y_train, Y_valid = train_test_split(
    X, Y, train_size=0.8, test_size=0.2, random_state=0)
model_LR.fit(X_train, Y_train)

print(f"model co efficients {model_LR.coef_}")
print(f"model intercept_ {model_LR.intercept_}")
Y_pred = model_LR.predict(X_valid)
```

```
print(f"Mean error percentage {mean_absolute_percentage_error(Y_valid,␣
 ↪Y_pred)}")
```

```
model co efficients [ 0.20283251  0.16022718  0.05740673  0.43452748  0.01966939
0.10139186
 -0.07914115 -0.04690223  0.00478868]
model intercept_ -10.683345413372333
Mean error percentage 4.196130727749416
```

[318]:
```
type(X_valid.iloc[0])
```

[318]:
```
pandas.core.series.Series
```

[319]:
```
Y_pred = model_LR.predict([X_valid.iloc[0]])
print(f"{Y_pred} actual is {Y_valid.iloc[0]}")
```

```
[0.53103594] actual is 0.5162322222194581
```

0.3.14 Observations and Conclusions(1M)

We see that 'OverallQual','TotalBsmtSF','1stFlrSF','GrLivArea','GarageCars','GarageArea','KitchenAbvGr','Full are important features that has impact on the house price. we have Fitted a linear regression modelbasedon these faetures to predict saleprice of houses forany new data. Our modeldoesnot have any over fitting and performs welll with new data as well with accuract of 96%

0.3.15 Solution (1M)

What is the solution that is proposed to solve the business problem discussed in the beginning. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

The solution involved the following 1. Complete anaysis of problem space , data source, data schema,metadata, datatype 2. Indentified the type of data and object type to clean the data 3. checked for data quality issues and fixed them with techniques like imputation, duplicates removal, not a valid value identification 4. Exploratory data analysis with various visualistion chart was doen to identify tren and correlation. 5. Nomalisation and standarisation was done to make the data Machine learning algorith ready 6. Binning and discretionsation was done to convert continous features to discrete bins withouwith minimal information loss 7. Pearson correlation and feature selection were used to indentify the most important attributes. 8. LInear regression model fitting , testing, evaluation and prediction for new values were done.