

House Price Prediction

Life Cycle of a Machine Learning Project

1. Understanding the Problem Statement
2. Data Collection
3. EDA (Exploratory Data Analysis)
4. Data Cleaning
5. Data Pre-Processing
6. Model Training
7. Choose Best Model

Problem Statement

1. This dataset contains data for houses to be sold and various features of those houses.
2. Aim is to predict the price of the house based on the features.
3. Before creating a model we need to do EDA.

Data Description

Description of the data is present in 'data_description.txt' file

A) Importing Data and Required Packages

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import seaborn as sns
        4 import matplotlib.pyplot as plt
        5 %matplotlib inline
        6
        7 import warnings
        8 warnings.filterwarnings('ignore')
```

```
C:\Users\Sachin Dev\AppData\Roaming\Python\Python38\site-packages\pandas\core\computation\expressions.py:20: UserWarning: Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

```
In [2]: 1 pd.pandas.set_option('display.max_columns',None)
```

Import csv as Pandas Dataframe

```
In [3]: 1 df = pd.read_csv('HousingData.csv')
```

Show Top 5 Rows

In [4]: 1 df.head()

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPu
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPu
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPu
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPu
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPu

Show Bottom 5 Rows

In [5]: 1 df.tail()

Out[5]:

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

Shape of the data

In [6]: 1 df.shape

Out[6]: (1460, 81)

There are 81 columns in the dataset. So, we will need to do feature selection at one point.

Check Null Values and Data Types

In [7]: 1 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           1452 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

```

Some features have a lot of Null Values

Exploring Data

Check Missing Values

```
In [8]: 1 ###1: create a list of features which has missing values
2 features_with_na = [feature for feature in df.columns if df[feature].isnull(
3
4 ###2: printing the names of features and % of missing values
5 for feature in features_with_na:
6     print(feature, np.round(df[feature].isnull().mean()*100 , 4), '%missing
```

```
LotFrontage 17.7397 %missing values
Alley 93.7671 %missing values
MasVnrType 0.5479 %missing values
MasVnrArea 0.5479 %missing values
BsmtQual 2.5342 %missing values
BsmtCond 2.5342 %missing values
BsmtExposure 2.6027 %missing values
BsmtFinType1 2.5342 %missing values
BsmtFinType2 2.6027 %missing values
FireplaceQu 47.2603 %missing values
GarageType 5.5479 %missing values
GarageYrBlt 5.5479 %missing values
GarageFinish 5.5479 %missing values
GarageQual 5.5479 %missing values
GarageCond 5.5479 %missing values
PoolQC 99.5205 %missing values
Fence 80.7534 %missing values
MiscFeature 96.3014 %missing values
```

```
In [9]: 1 len(features_with_na)
```

Out[9]: 18

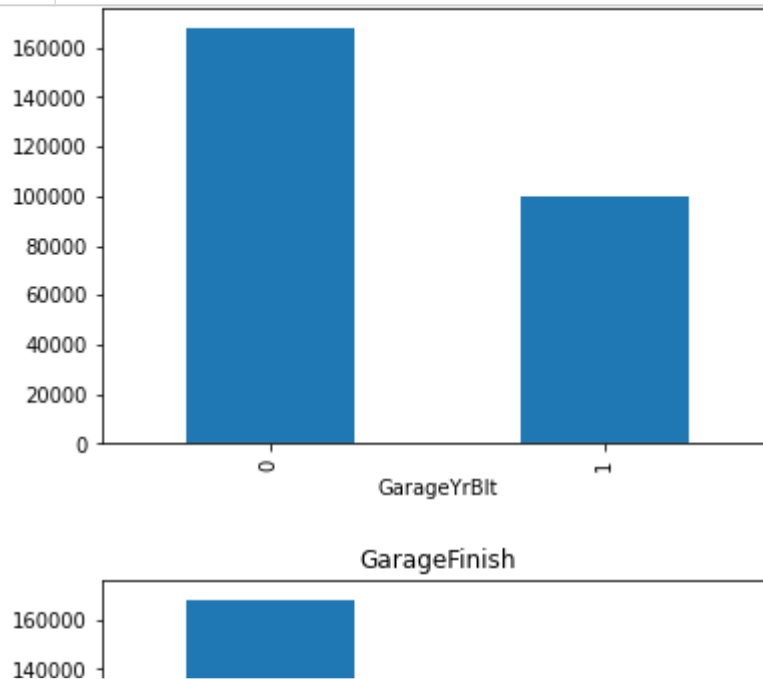
There are 18 features which have missing values and out of these 18 features 4 features have more than 80% Null values

Since there are many missing values, we need to find the relationship between missing values and SalesPrice(Target Column)

```

In [10]: 1 for feature in features_with_na:
          2     data = df.copy()
          3
          4     ### making a variable that indicates 1 if observation is missing and 0 o
          5     data[feature] = np.where(data[feature].isnull(), 1, 0) ### this will cre
          6
          7
          8     ### calculate mean of SalesPrice where information is missing or present
          9     data.groupby(feature)['SalePrice'].median().plot.bar()
         10     plt.title(feature)
         11     plt.show()

```



Here the relation of dependent features is clearly visible with the missing values. So, we will need to replace these Nan values with something meaningful during feature engineering process.

Also, some features like Id etc are not required so we can drop these features

```
In [ ]:
```

```
1
```

Numerical Features

```

In [11]: 1 numerical_features = [feature for feature in df.columns if df[feature].dtype
          2 print("Number of numerical features: ", len(numerical_features))
          3

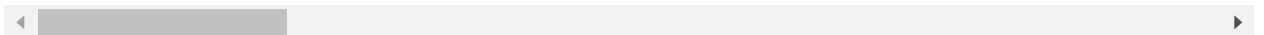
```

Number of numerical features: 38

```
In [12]: 1 ### Top 5 rows of numerical features
        2 df[numerical_features].head()
```

Out[12]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	Ma
0	1	60	65.0	8450	7	5	2003	2003	
1	2	20	80.0	9600	6	8	1976	1976	
2	3	60	68.0	11250	7	5	2001	2002	
3	4	70	60.0	9550	7	5	1915	1970	
4	5	60	84.0	14260	8	5	2000	2000	



```
In [13]: 1 for feature in numerical_features:
          2     print(feature)
```

```
Id
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice
```

Datetime Features (Temporal Variables)

In the dataset, we have few Year features. We can extract information from these features. Eg:
The difference b/w the year house built and the year house sold.

```
In [14]: 1 year_feature = [feature for feature in numerical_features if 'Yr' in feature]
          2 year_feature
```

```
Out[14]: ['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
```

Exploring year_feature

In [15]:

```

1 for feature in year_feature:
2     print(feature, df[feature].unique())
3     print('-----'*3)

```

YearBuilt [2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 1965 2005 1962 2006

1960 1929 1970 1967 1958 1930 2002 1968 2007 1951 1957 1927 1920 1966
 1959 1994 1954 1953 1955 1983 1975 1997 1934 1963 1981 1964 1999 1972
 1921 1945 1982 1998 1956 1948 1910 1995 1991 2009 1950 1961 1977 1985
 1979 1885 1919 1990 1969 1935 1988 1971 1952 1936 1923 1924 1984 1926
 1940 1941 1987 1986 2008 1908 1892 1916 1932 1918 1912 1947 1925 1900
 1980 1989 1992 1949 1880 1928 1978 1922 1996 2010 1946 1913 1937 1942
 1938 1974 1893 1914 1906 1890 1898 1904 1882 1875 1911 1917 1872 1905]

YearRemodAdd [2003 1976 2002 1970 2000 1995 2005 1973 1950 1965 2006 1962 2007
 1960

2001 1967 2004 2008 1997 1959 1990 1955 1983 1980 1966 1963 1987 1964
 1972 1996 1998 1989 1953 1956 1968 1981 1992 2009 1982 1961 1993 1999
 1985 1979 1977 1969 1958 1991 1971 1952 1975 2010 1984 1986 1994 1988
 1954 1957 1951 1978 1974]

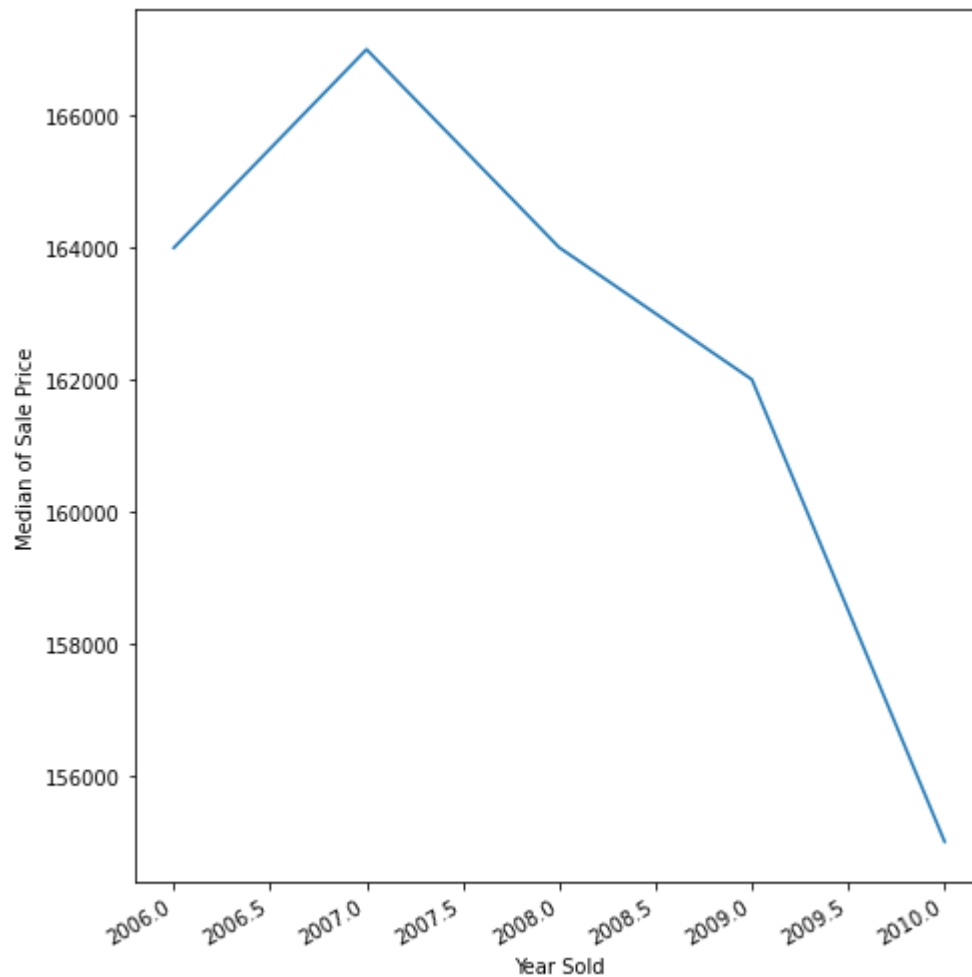
GarageYrBlt [2003. 1976. 2001. 1998. 2000. 1993. 2004. 1973. 1931. 1939. 1965.
 2005.

1962. 2006. 1960. 1991. 1970. 1967. 1958. 1930. 2002. 1968. 2007. 2008.
 1957. 1920. 1966. 1959. 1995. 1954. 1953. nan 1983. 1977. 1997. 1985.
 1963. 1981. 1964. 1999. 1935. 1990. 1945. 1987. 1989. 1915. 1956. 1948.
 1974. 2009. 1950. 1961. 1921. 1900. 1979. 1951. 1969. 1936. 1975. 1971.
 1923. 1984. 1926. 1955. 1986. 1988. 1916. 1932. 1972. 1918. 1980. 1924.
 1996. 1940. 1949. 1994. 1910. 1978. 1982. 1992. 1925. 1941. 2010. 1927.
 1947. 1937. 1942. 1938. 1952. 1928. 1922. 1934. 1906. 1914. 1946. 1908.
 1929. 1933.]

YrSold [2008 2007 2006 2009 2010]

Relation between Year sold and SalePrice

```
In [16]: 1 plt.figure(figsize=(7,7))
2
3 df.groupby('YrSold')['SalePrice'].median().plot()
4
5 plt.gcf().autofmt_xdate()
6 plt.xlabel("Year Sold Vs Sale Price")
7 plt.xlabel('Year Sold')
8 plt.ylabel('Median of Sale Price')
9 plt.tight_layout()
10 plt.show()
```



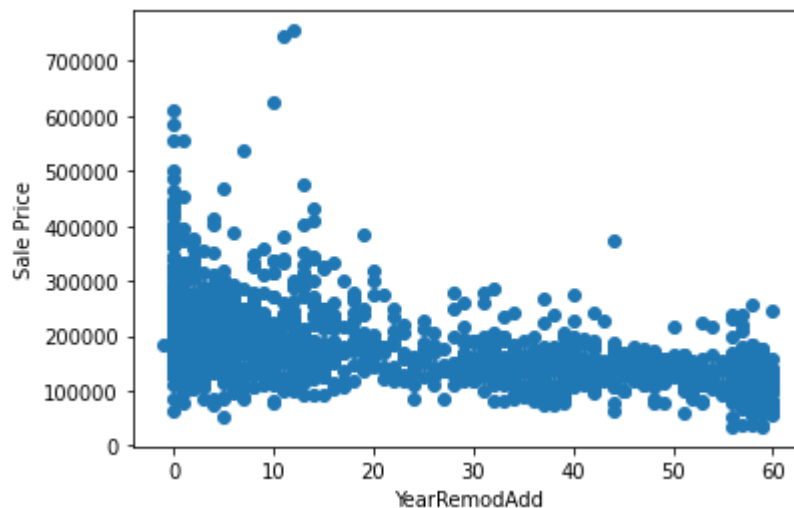
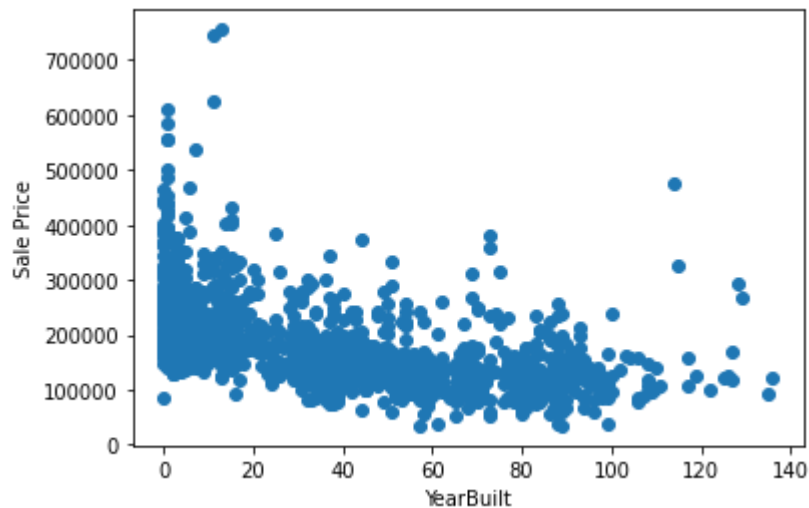
Clearly we can see that from 2007 onwards Price of Houses decreases with the increase in Year

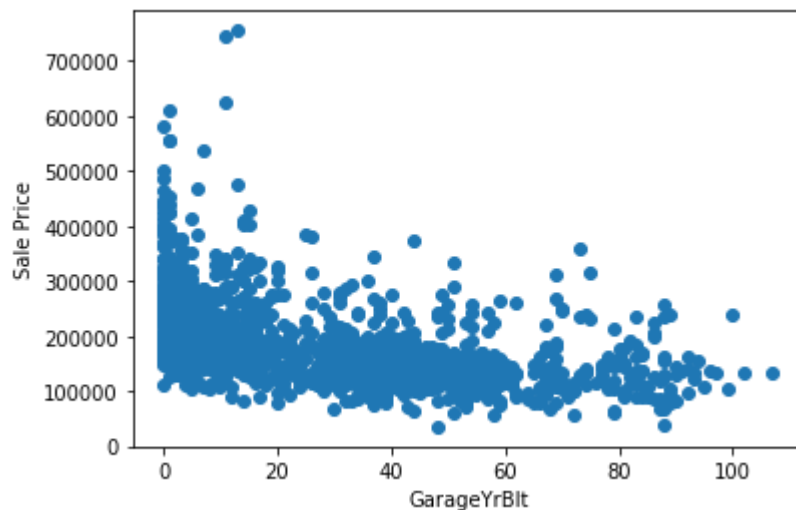
Compare the difference b/w all year features with sale price

In [17]: 1 year_feature

Out[17]: ['YearBuilt', 'YearRemodAdd', 'GarageYrBltn', 'YrSold']

```
In [18]: 1 for feature in year_feature:
2         if feature != 'YrSold':
3             data = df.copy()
4
5             ### eg feature is YearBuilt = 1960 and Yr Sold = 2000
6             ### so, data[feature] = 2000 - 1960 = 40yrs
7
8             data[feature] = data['YrSold'] - data[feature]
9
10            plt.scatter(data[feature], data['SalePrice'])
11            plt.xlabel(feature)
12            plt.ylabel('Sale Price')
13            plt.show()
```





AS expected the newer the house or renovated early or new garage, higher the price.

Discrete Features

```
In [19]: 1  ### variables with less than 25 unique values
          2  ### and variables should not be preset in year features
          3  ### exlude Id cloumn also
          4
          5  discrete_feature = [feature for feature in numerical_features if len(df[feat
          6  print("Number of Discrete Features: {}".format(len(discrete_feature)))
```

Number of Discrete Features: 17

```
In [20]: 1  discrete_feature
```

```
Out[20]: ['MSSubClass',
          'OverallQual',
          'OverallCond',
          'LowQualFinSF',
          'BsmtFullBath',
          'BsmtHalfBath',
          'FullBath',
          'HalfBath',
          'BedroomAbvGr',
          'KitchenAbvGr',
          'TotRmsAbvGrd',
          'Fireplaces',
          'GarageCars',
          '3SsnPorch',
          'PoolArea',
          'MiscVal',
          'MoSold']
```

Visualize Discrete features

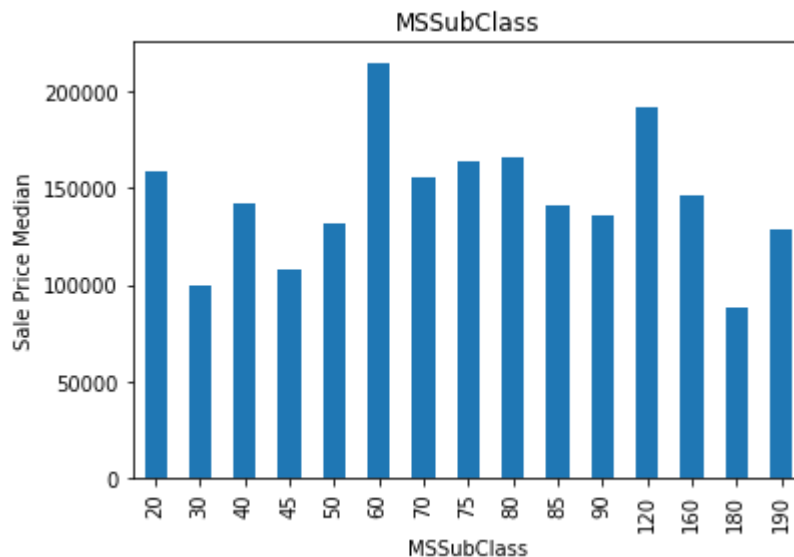
In [21]: 1 df[discrete_feature].head()

Out[21]:

| | MSSubClass | OverallQual | OverallCond | LowQualFinSF | BsmtFullBath | BsmtHalfBath | FullBath | H |
|---|------------|-------------|-------------|--------------|--------------|--------------|----------|---|
| 0 | 60 | 7 | 5 | 0 | 1 | 0 | 2 | |
| 1 | 20 | 6 | 8 | 0 | 0 | 1 | 2 | |
| 2 | 60 | 7 | 5 | 0 | 1 | 0 | 2 | |
| 3 | 70 | 7 | 5 | 0 | 1 | 0 | 1 | |
| 4 | 60 | 8 | 5 | 0 | 1 | 0 | 2 | |

Relationship of Discrete features with Price Column

```
In [22]: 1
2 for feature in discrete_feature:
3     data = df.copy()
4
5     data.groupby(feature)['SalePrice'].median().plot.bar()
6     plt.xlabel(feature)
7     plt.ylabel('Sale Price Median')
8     plt.title(feature)
9     plt.show()
```



Observations

1. Overall Quality has a positive correlation with Sale Price.
2. Full Bath has a positive correlation with Sale Price.
3. Sale Price increases with the increase in TotRmsAbvGrd from 2 to 11, then the sale price decreases.
4. Even a single Fireplace can impact or increase the price of the House.
5. Houses with 3 GarageCars have the highest price.

Continuous Features

```
In [23]: 1  ### variables that are not present in discrete feature and year feature list
        2  ### Also exclude Id column
        3
        4  continuous_feature = [feature for feature in numerical_features if feature n
        5  print(f'Number of Continuous Features: {len(continuous_feature)}')
```

Number of Continuous Features: 16

```
In [24]: 1  continuous_feature
```

```
Out[24]: ['LotFrontage',
          'LotArea',
          'MasVnrArea',
          'BsmtFinSF1',
          'BsmtFinSF2',
          'BsmtUnfSF',
          'TotalBsmtSF',
          '1stFlrSF',
          '2ndFlrSF',
          'GrLivArea',
          'GarageArea',
          'WoodDeckSF',
          'OpenPorchSF',
          'EnclosedPorch',
          'ScreenPorch',
          'SalePrice']
```

Visualize continuous_features

```
In [25]: 1  df[continuous_feature].head()
```

```
Out[25]:
```

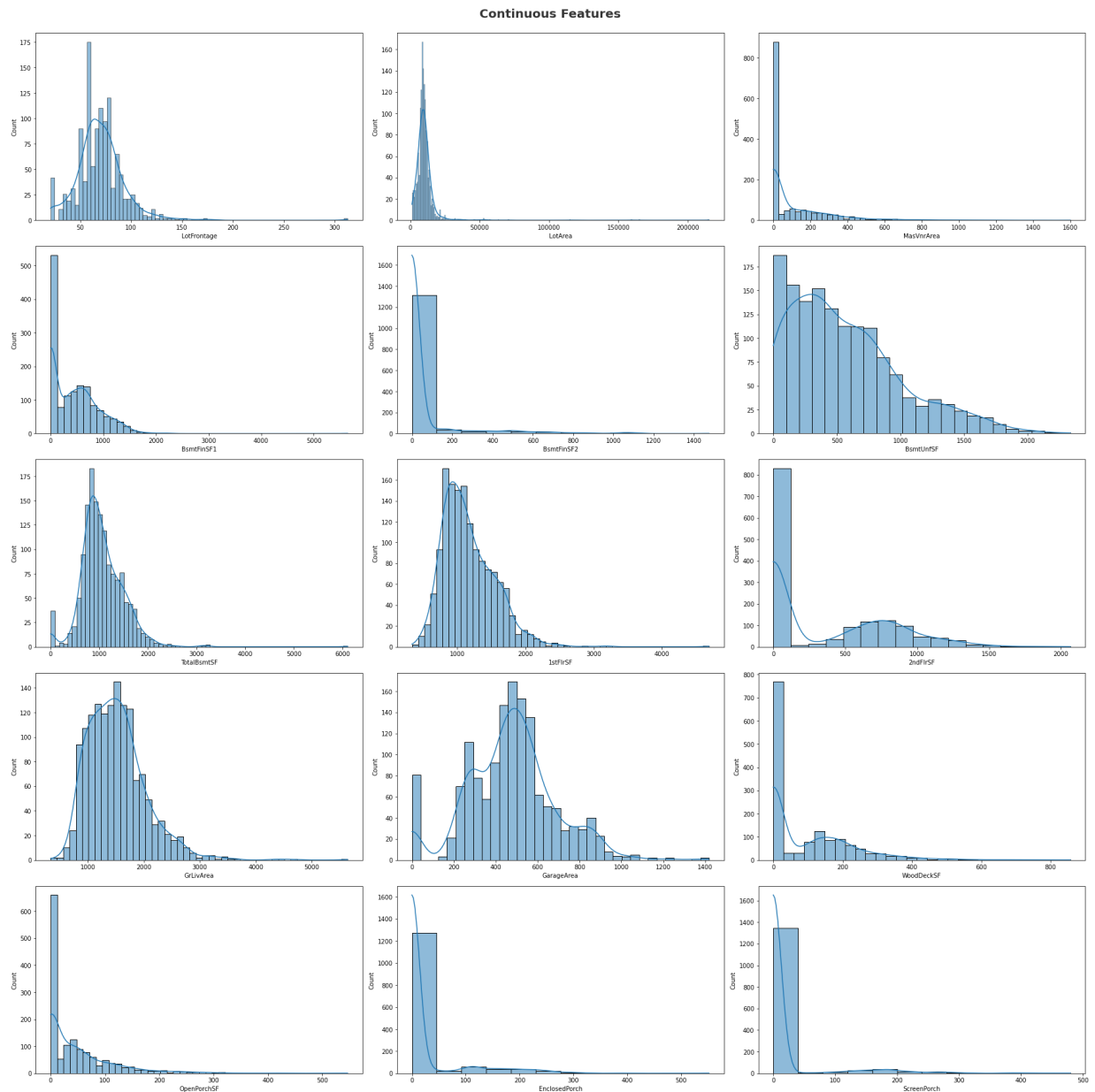
| | LotFrontage | LotArea | MasVnrArea | BsmtFinSF1 | BsmtFinSF2 | BsmtUnfSF | TotalBsmtSF | 1stFlrS |
|---|-------------|---------|------------|------------|------------|-----------|-------------|---------|
| 0 | 65.0 | 8450 | 196.0 | 706 | 0 | 150 | 856 | 85 |
| 1 | 80.0 | 9600 | 0.0 | 978 | 0 | 284 | 1262 | 126 |
| 2 | 68.0 | 11250 | 162.0 | 486 | 0 | 434 | 920 | 92 |
| 3 | 60.0 | 9550 | 0.0 | 216 | 0 | 540 | 756 | 96 |
| 4 | 84.0 | 14260 | 350.0 | 655 | 0 | 490 | 1145 | 114 |

Analyze continuous features with the help of Histograms

```

In [26]: 1 plt.figure(figsize=(25,25))
2 plt.suptitle("Continuous Features",fontsize=20,fontweight='bold',alpha=0.8,y
3
4 for i in range(len(continuous_feature)):
5     if continuous_feature[i] != 'SalePrice':
6         plt.subplot(5,3,i+1)
7         sns.histplot(data=df,x=df[continuous_feature[i]],kde=True)
8         plt.xlabel(continuous_feature[i])
9         plt.ylabel("Count")
10        plt.tight_layout()

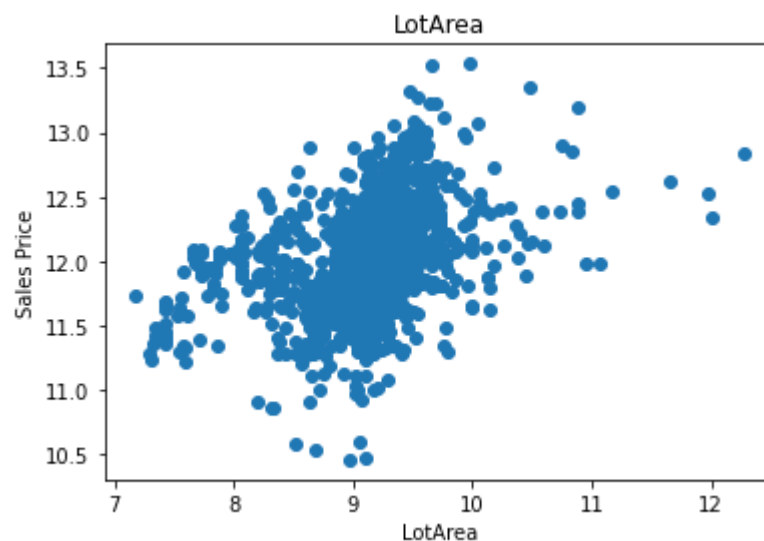
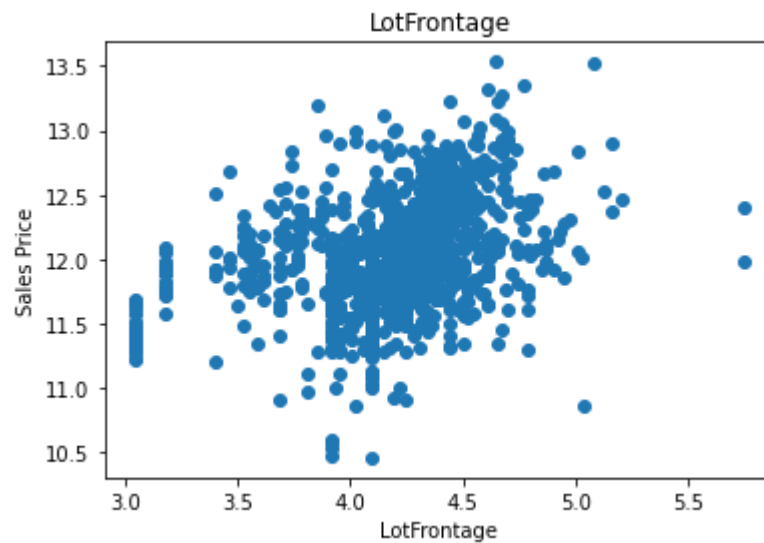
```

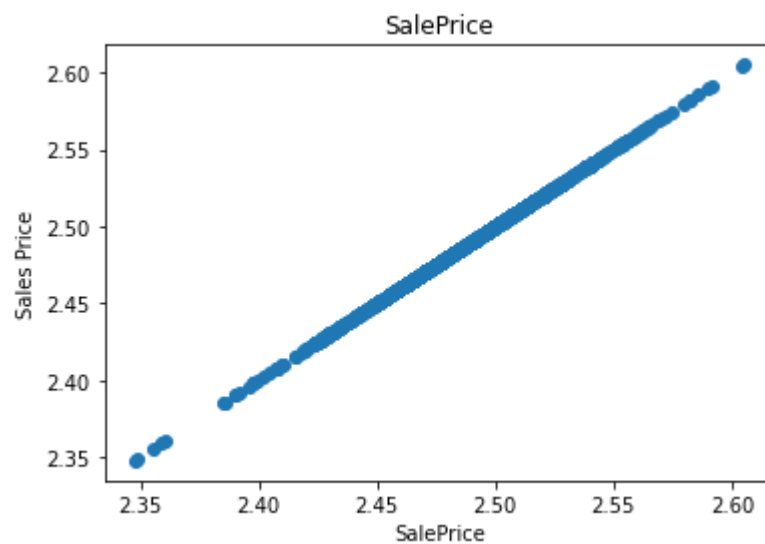
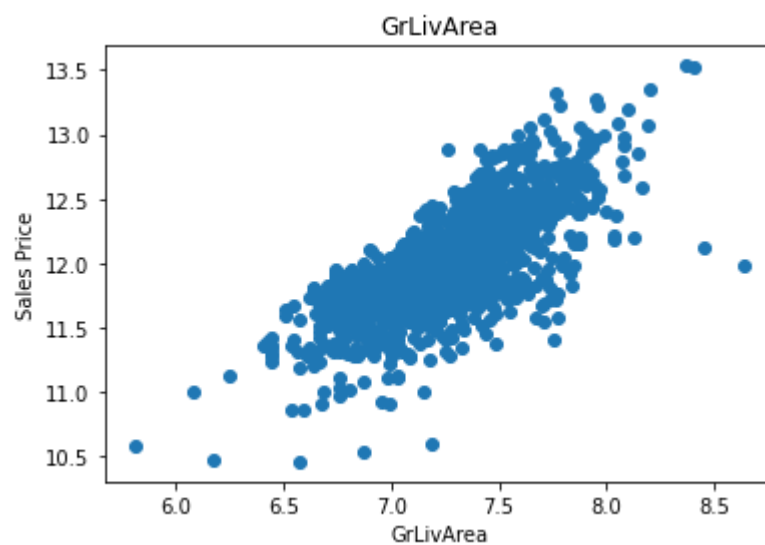
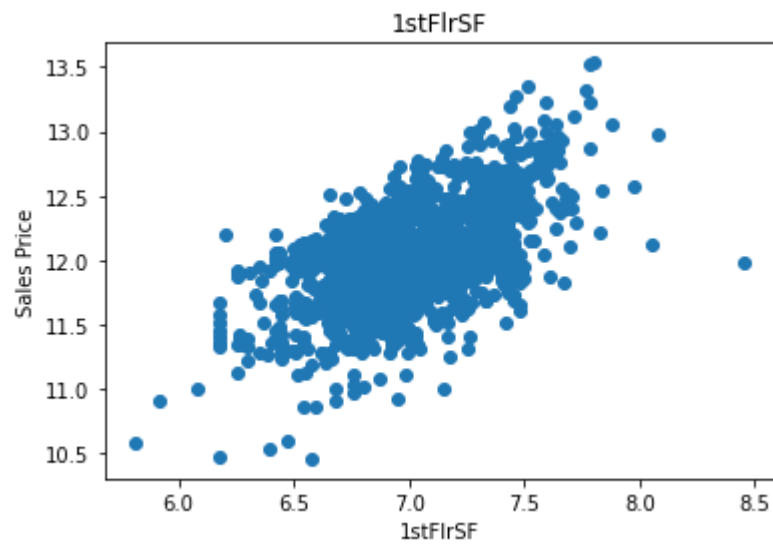


There are a lot of columns which are skewed.

log Transformation to convert skewed data to normal distribution

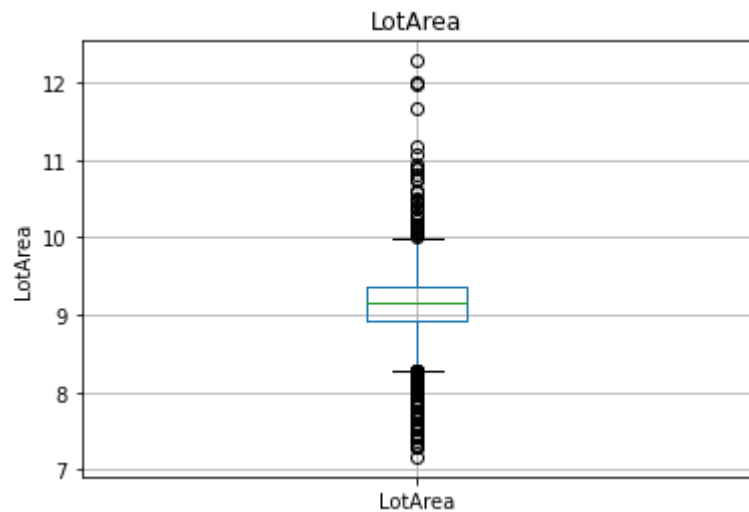
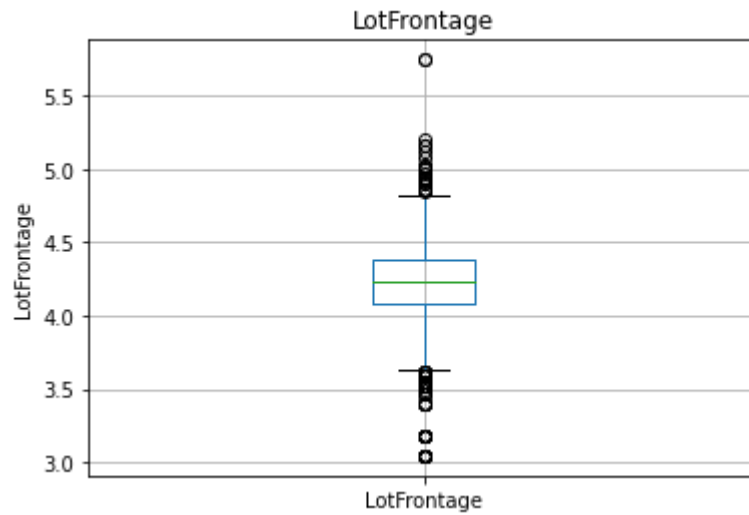
```
In [27]: 1 for feature in continuous_feature:
2         data = df.copy()
3
4         ##skip 0 because log(0) --> infinte
5         if 0 in data[feature].unique():
6             pass
7         else:
8             data[feature] = np.log(data[feature])
9             data['SalePrice'] = np.log(data['SalePrice'])
10            plt.scatter(data[feature],data['SalePrice'])
11            plt.xlabel(feature)
12            plt.title(feature)
13            plt.ylabel('Sales Price')
14            plt.show()
```

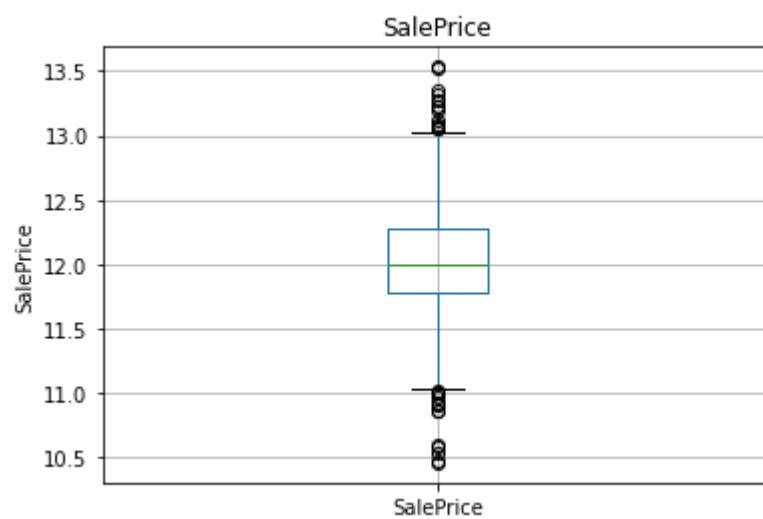
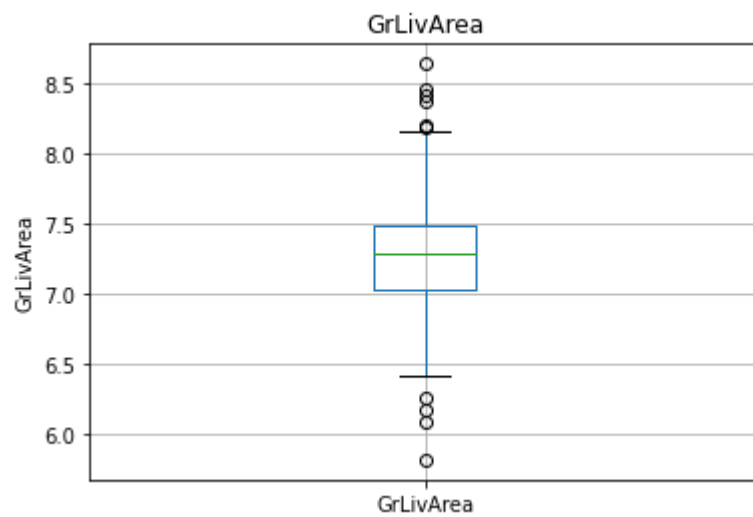
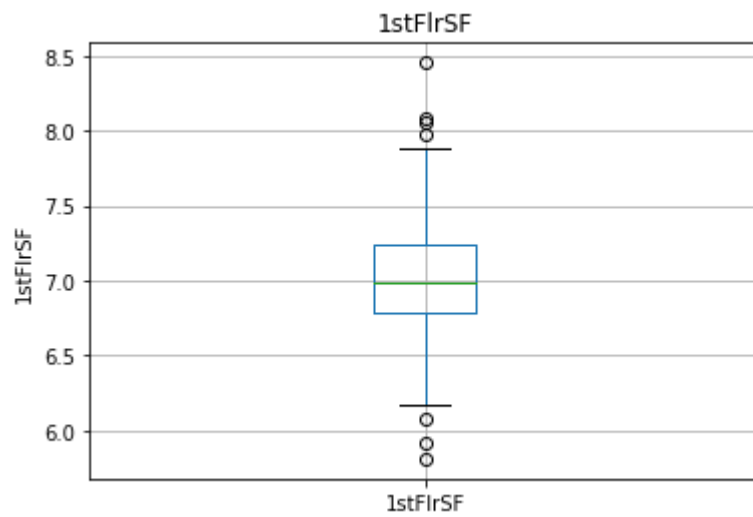




Check Outliers

```
In [28]: 1 for feature in continuous_feature:
2         data = df.copy()
3         if 0 in data[feature].unique():
4             pass
5         else:
6             data[feature] = np.log(data[feature])
7             data.boxplot(column=feature)
8             plt.ylabel(feature)
9             plt.title(feature)
10            plt.show()
```





There are lot of outliers in the dataset

Categorical Variables

```
In [29]: 1 categorical_feature = [feature for feature in df.columns if df[feature].dtype
2         print(f"Number of Categorical Feature: {len(categorical_feature)}")
```

Number of Categorical Feature: 43

```
In [30]: 1 categorical_feature
```

```
Out[30]: ['MSZoning',
'Street',
'Alley',
'LotShape',
'LandContour',
'Utilities',
'LotConfig',
'LandSlope',
'Neighborhood',
'Condition1',
'Condition2',
'BldgType',
'HouseStyle',
'RoofStyle',
'RoofMatl',
'Exterior1st',
'Exterior2nd',
'MasVnrType',
'ExterQual',
'ExterCond',
'Foundation',
'BsmtQual',
'BsmtCond',
'BsmtExposure',
'BsmtFinType1',
'BsmtFinType2',
'Heating',
'HeatingQC',
'CentralAir',
'Electrical',
'KitchenQual',
'Functional',
'FireplaceQu',
'GarageType',
'GarageFinish',
'GarageQual',
'GarageCond',
'PavedDrive',
'PoolQC',
'Fence',
'MiscFeature',
'SaleType',
'SaleCondition']
```

In [31]: 1 df[categorical_feature].head()

Out[31]:

| | MSZoning | Street | Alley | LotShape | LandContour | Utilities | LotConfig | LandSlope | Neighborhood |
|---|----------|--------|-------|----------|-------------|-----------|-----------|-----------|--------------|
| 0 | RL | Pave | NaN | Reg | Lvl | AllPub | Inside | Gtl | CollgCr |
| 1 | RL | Pave | NaN | Reg | Lvl | AllPub | FR2 | Gtl | Veenker |
| 2 | RL | Pave | NaN | IR1 | Lvl | AllPub | Inside | Gtl | CollgCr |
| 3 | RL | Pave | NaN | IR1 | Lvl | AllPub | Corner | Gtl | Crawfor |
| 4 | RL | Pave | NaN | IR1 | Lvl | AllPub | FR2 | Gtl | NoRidge |

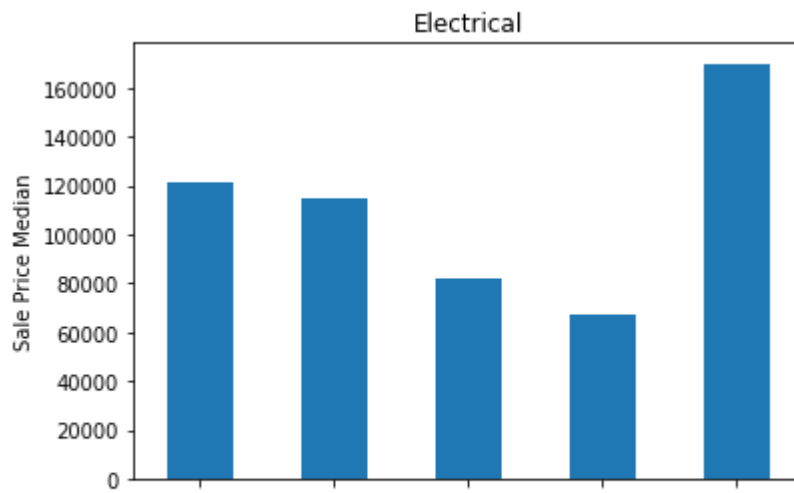
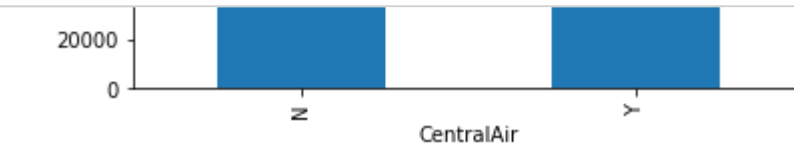
Cardinal Values -- categories inside categorical values

```
In [32]: 1 for feature in categorical_feature:
          2     print('feature is {} and num of categories are {}'.format(feature, len(df[feature])))

feature is MSZoning and num of categories are 5
feature is Street and num of categories are 2
feature is Alley and num of categories are 3
feature is LotShape and num of categories are 4
feature is LandContour and num of categories are 4
feature is Utilities and num of categories are 2
feature is LotConfig and num of categories are 5
feature is LandSlope and num of categories are 3
feature is Neighborhood and num of categories are 25
feature is Condition1 and num of categories are 9
feature is Condition2 and num of categories are 8
feature is BldgType and num of categories are 5
feature is HouseStyle and num of categories are 8
feature is RoofStyle and num of categories are 6
feature is RoofMatl and num of categories are 8
feature is Exterior1st and num of categories are 15
feature is Exterior2nd and num of categories are 16
feature is MasVnrType and num of categories are 5
feature is ExterQual and num of categories are 4
feature is ExterCond and num of categories are 5
feature is Foundation and num of categories are 6
feature is BsmtQual and num of categories are 5
feature is BsmtCond and num of categories are 5
feature is BsmtExposure and num of categories are 5
feature is BsmtFinType1 and num of categories are 7
feature is BsmtFinType2 and num of categories are 7
feature is Heating and num of categories are 6
feature is HeatingQC and num of categories are 5
feature is CentralAir and num of categories are 2
feature is Electrical and num of categories are 6
feature is KitchenQual and num of categories are 4
feature is Functional and num of categories are 7
feature is FireplaceQu and num of categories are 6
feature is GarageType and num of categories are 7
feature is GarageFinish and num of categories are 4
feature is GarageQual and num of categories are 6
feature is GarageCond and num of categories are 6
feature is PavedDrive and num of categories are 3
feature is PoolQC and num of categories are 4
feature is Fence and num of categories are 5
feature is MiscFeature and num of categories are 5
feature is SaleType and num of categories are 9
feature is SaleCondition and num of categories are 6
```

Relationship Between Categorical feature and Sale Price


```
In [33]: 1 for feature in categorical_feature:
2         data = df.copy()
3         data.groupby(feature)['SalePrice'].median().plot.bar()
4         plt.xlabel(feature)
5         plt.ylabel('Sale Price Median')
6         plt.title(feature)
7         plt.show()
```



Feature Engineering

Missing Values for Categorical Features

```
In [34]: 1 features_nan = [feature for feature in df.columns if df[feature].isnull().su
2
3         for feature in features_nan:
4             print('{}: {}% missing values'.format(feature, np.round(df[feature].isnu
```

```
Alley: 93.7671% missing values
MasVnrType: 0.5479% missing values
BsmtQual: 2.5342% missing values
BsmtCond: 2.5342% missing values
BsmtExposure: 2.6027% missing values
BsmtFinType1: 2.5342% missing values
BsmtFinType2: 2.6027% missing values
FireplaceQu: 47.2603% missing values
GarageType: 5.5479% missing values
GarageFinish: 5.5479% missing values
GarageQual: 5.5479% missing values
GarageCond: 5.5479% missing values
PoolQC: 99.5205% missing values
Fence: 80.7534% missing values
MiscFeature: 96.3014% missing values
```

Replacing missing values for Categorical Features with a new label

```
In [35]: 1 def replace_cat_features(df, features_nan):
2         data = df.copy()
3         data[features_nan] = data[features_nan].fillna('Missing')
4         return data
5
6 df = replace_cat_features(df, features_nan)
7 df[features_nan].isnull().sum()
```

```
Out[35]: Alley          0
MasVnrType          0
BsmtQual            0
BsmtCond            0
BsmtExposure        0
BsmtFinType1        0
BsmtFinType2        0
FireplaceQu         0
GarageType          0
GarageFinish        0
GarageQual          0
GarageCond          0
PoolQC              0
Fence               0
MiscFeature         0
dtype: int64
```

```
In [36]: 1 df.head()
```

```
Out[36]:
```

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Utili |
|---|----|------------|----------|-------------|---------|--------|---------|----------|-------------|-------|
| 0 | 1 | 60 | RL | 65.0 | 8450 | Pave | Missing | Reg | Lvl | All |
| 1 | 2 | 20 | RL | 80.0 | 9600 | Pave | Missing | Reg | Lvl | All |
| 2 | 3 | 60 | RL | 68.0 | 11250 | Pave | Missing | IR1 | Lvl | All |
| 3 | 4 | 70 | RL | 60.0 | 9550 | Pave | Missing | IR1 | Lvl | All |
| 4 | 5 | 60 | RL | 84.0 | 14260 | Pave | Missing | IR1 | Lvl | All |

```
In [ ]: 1
```

Missing Values for Numerical Variables

```
In [37]: 1 numerical_with_nan = [feature for feature in df.columns if df[feature].isnull
2
3 for feature in numerical_with_nan:
4     print('{}: {}%missing values'.format(feature,np.round(df[feature].isnull
```

LotFrontage: 17.7397%missing values

MasVnrArea: 0.5479%missing values

GarageYrBlt: 5.5479%missing values

Replacing Numeric Missing Values

```
In [38]: 1 for feature in numerical_with_nan:
2     median_value = df[feature].median()
3
4     ### creating a new feature if feature has Nan then replace it with 1 els
5     df[feature+'nan'] = np.where(df[feature].isnull(),1,0)
6     df[feature].fillna(median_value,inplace=True)
7
8     df[numerical_with_nan].isnull().sum()
```

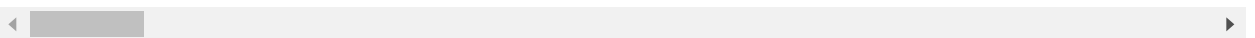
```
Out[38]: LotFrontage      0
MasVnrArea      0
GarageYrBlt     0
dtype: int64
```

In [39]:

```
1 df.head(20)
```

Out[39]:

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Utilities |
|----|----|------------|----------|-------------|---------|--------|---------|----------|-------------|-----------|
| 0 | 1 | 60 | RL | 65.0 | 8450 | Pave | Missing | Reg | Lvl | / |
| 1 | 2 | 20 | RL | 80.0 | 9600 | Pave | Missing | Reg | Lvl | / |
| 2 | 3 | 60 | RL | 68.0 | 11250 | Pave | Missing | IR1 | Lvl | / |
| 3 | 4 | 70 | RL | 60.0 | 9550 | Pave | Missing | IR1 | Lvl | / |
| 4 | 5 | 60 | RL | 84.0 | 14260 | Pave | Missing | IR1 | Lvl | / |
| 5 | 6 | 50 | RL | 85.0 | 14115 | Pave | Missing | IR1 | Lvl | / |
| 6 | 7 | 20 | RL | 75.0 | 10084 | Pave | Missing | Reg | Lvl | / |
| 7 | 8 | 60 | RL | 69.0 | 10382 | Pave | Missing | IR1 | Lvl | / |
| 8 | 9 | 50 | RM | 51.0 | 6120 | Pave | Missing | Reg | Lvl | / |
| 9 | 10 | 190 | RL | 50.0 | 7420 | Pave | Missing | Reg | Lvl | / |
| 10 | 11 | 20 | RL | 70.0 | 11200 | Pave | Missing | Reg | Lvl | / |
| 11 | 12 | 60 | RL | 85.0 | 11924 | Pave | Missing | IR1 | Lvl | / |
| 12 | 13 | 20 | RL | 69.0 | 12968 | Pave | Missing | IR2 | Lvl | / |
| 13 | 14 | 20 | RL | 91.0 | 10652 | Pave | Missing | IR1 | Lvl | / |
| 14 | 15 | 20 | RL | 69.0 | 10920 | Pave | Missing | IR1 | Lvl | / |
| 15 | 16 | 45 | RM | 51.0 | 6120 | Pave | Missing | Reg | Lvl | / |
| 16 | 17 | 20 | RL | 69.0 | 11241 | Pave | Missing | IR1 | Lvl | / |
| 17 | 18 | 90 | RL | 72.0 | 10791 | Pave | Missing | Reg | Lvl | / |
| 18 | 19 | 20 | RL | 66.0 | 13695 | Pave | Missing | Reg | Lvl | / |
| 19 | 20 | 20 | RL | 70.0 | 7560 | Pave | Missing | Reg | Lvl | / |



Datetime Variables

In [40]:

```
1 for feature in ['YearBuilt', 'YearRemodAdd', 'GarageYrBlt']:
2     df[feature] = df['YrSold'] - df[feature]
```

In [41]:

```
1 df.head()
```

Out[41]:

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Utili |
|---|----|------------|----------|-------------|---------|--------|---------|----------|-------------|-------|
| 0 | 1 | 60 | RL | 65.0 | 8450 | Pave | Missing | Reg | Lvl | All |
| 1 | 2 | 20 | RL | 80.0 | 9600 | Pave | Missing | Reg | Lvl | All |
| 2 | 3 | 60 | RL | 68.0 | 11250 | Pave | Missing | IR1 | Lvl | All |
| 3 | 4 | 70 | RL | 60.0 | 9550 | Pave | Missing | IR1 | Lvl | All |
| 4 | 5 | 60 | RL | 84.0 | 14260 | Pave | Missing | IR1 | Lvl | All |

In [42]:

```
1 df[['YearBuilt', 'YearRemodAdd', 'GarageYrBlt']].head()
```

Out[42]:

| | YearBuilt | YearRemodAdd | GarageYrBlt |
|---|-----------|--------------|-------------|
| 0 | 5 | 5 | 5.0 |
| 1 | 31 | 31 | 31.0 |
| 2 | 7 | 6 | 7.0 |
| 3 | 91 | 36 | 8.0 |
| 4 | 8 | 8 | 8.0 |

Transform skewed feature with the help of log normal distribution

In [43]:

```
1 df.head()
```

Out[43]:

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Utili |
|---|----|------------|----------|-------------|---------|--------|---------|----------|-------------|-------|
| 0 | 1 | 60 | RL | 65.0 | 8450 | Pave | Missing | Reg | Lvl | All |
| 1 | 2 | 20 | RL | 80.0 | 9600 | Pave | Missing | Reg | Lvl | All |
| 2 | 3 | 60 | RL | 68.0 | 11250 | Pave | Missing | IR1 | Lvl | All |
| 3 | 4 | 70 | RL | 60.0 | 9550 | Pave | Missing | IR1 | Lvl | All |
| 4 | 5 | 60 | RL | 84.0 | 14260 | Pave | Missing | IR1 | Lvl | All |

In [44]:

```

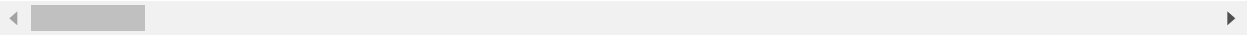
1 num_features = ['LotFrontage', 'LotArea', '1stFlrSF', 'GrLivArea', 'SalePrice']
2
3 for feature in num_features:
4     df[feature] = np.log(df[feature])

```

In [45]: 1 df.head()

Out[45]:

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Uti |
|---|----|------------|----------|-------------|----------|--------|---------|----------|-------------|-----|
| 0 | 1 | 60 | RL | 4.174387 | 9.041922 | Pave | Missing | Reg | Lvl | A |
| 1 | 2 | 20 | RL | 4.382027 | 9.169518 | Pave | Missing | Reg | Lvl | A |
| 2 | 3 | 60 | RL | 4.219508 | 9.328123 | Pave | Missing | IR1 | Lvl | A |
| 3 | 4 | 70 | RL | 4.094345 | 9.164296 | Pave | Missing | IR1 | Lvl | A |
| 4 | 5 | 60 | RL | 4.430817 | 9.565214 | Pave | Missing | IR1 | Lvl | A |



Handling Rare Categorical Features

We will remove categorical features that are present less than 1% of the observations

```
In [46]: 1 categorical_features = [feature for feature in df.columns if df[feature].dtype
2         categorical_features
```

```
Out[46]: ['MSZoning',
          'Street',
          'Alley',
          'LotShape',
          'LandContour',
          'Utilities',
          'LotConfig',
          'LandSlope',
          'Neighborhood',
          'Condition1',
          'Condition2',
          'BldgType',
          'HouseStyle',
          'RoofStyle',
          'RoofMatl',
          'Exterior1st',
          'Exterior2nd',
          'MasVnrType',
          'ExterQual',
          'ExterCond',
          'Foundation',
          'BsmtQual',
          'BsmtCond',
          'BsmtExposure',
          'BsmtFinType1',
          'BsmtFinType2',
          'Heating',
          'HeatingQC',
          'CentralAir',
          'Electrical',
          'KitchenQual',
          'Functional',
          'FireplaceQu',
          'GarageType',
          'GarageFinish',
          'GarageQual',
          'GarageCond',
          'PavedDrive',
          'PoolQC',
          'Fence',
          'MiscFeature',
          'SaleType',
          'SaleCondition']
```

```
In [47]: 1 for feature in categorical_features:
2         temp = df.groupby(feature)['SalePrice'].count()
3         print(temp)
```

```
BrDale      16
BrkSide     58
ClearCr     28
CollgCr    150
Crawfor     51
```

```
Edwards    100
Gilbert     79
IDOTRR      37
MeadowV     17
Mitchel     49
NAMES      225
NPkVill      9
NWAmes      73
NoRidge     41
NridgHt     77
OldTown    113
SWISU       25
Sawyer      74
SawyerH     50
```

```
In [48]: 1 for feature in categorical_features:
2         temp = df.groupby(feature)['SalePrice'].count()/len(df)
3         temp_df = temp[temp > 0.01].index
4         df[feature] = np.where(df[feature].isin(temp_df),df[feature], 'Rare_Var')
```

```
In [49]: 1 df.head()
```

Out[49]:

| | Id | MSSubClass | MSZoning | LotFrontage | LotArea | Street | Alley | LotShape | LandContour | Uti |
|---|----|------------|----------|-------------|----------|--------|---------|----------|-------------|-----|
| 0 | 1 | 60 | RL | 4.174387 | 9.041922 | Pave | Missing | Reg | Lvl | A |
| 1 | 2 | 20 | RL | 4.382027 | 9.169518 | Pave | Missing | Reg | Lvl | A |
| 2 | 3 | 60 | RL | 4.219508 | 9.328123 | Pave | Missing | IR1 | Lvl | A |
| 3 | 4 | 70 | RL | 4.094345 | 9.164296 | Pave | Missing | IR1 | Lvl | A |
| 4 | 5 | 60 | RL | 4.430817 | 9.565214 | Pave | Missing | IR1 | Lvl | A |

```
In [ ]: 1
```