HOUSE PRICE DATASET

Team: Deep Team

Dataset Description ¶

Task 04

DATA DESCRIPTION MSSubClass: Identifies the type of dwelling involved in the sale.

```
    20    1-STORY 1946 & NEWER ALL STYLES
    30    1-STORY 1945 & OLDER
    40    1-STORY W/FINISHED ATTIC ALL AGES
    45    1-1/2 STORY - UNFINISHED ALL AGES
    50    1-1/2 STORY FINISHED ALL AGES
    60    2-STORY 1946 & NEWER
    70    2-STORY 1945 & OLDER
    75    2-1/2 STORY ALL AGES
    80    SPLIT OR MULTI-LEVEL
    85    SPLIT FOYER
    90    DUPLEX - ALL STYLES AND AGES
   120    1-STORY PUD (Planned Unit Development) - 1946 & NEWER
   150    1-1/2 STORY PUD - ALL AGES
   160    2-STORY PUD - 1946 & NEWER
   180    PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
   190    2 FAMILY CONVERSION - ALL STYLES AND AGES
```

MSZoning: Identifies the general zoning classification of the sale.

```
   A    Agriculture
   C    Commercial
   FV    Floating Village Residential
   I    Industrial
   RH    Residential High Density
   RL    Residential Low Density
   RP    Residential Low Density Park
   RM    Residential Medium Density
```

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

```
   Grvl    Gravel
   Pave    Paved
```

Alley: Type of alley access to property

```
Grvl    Gravel
Pave    Paved
NA     No alley access
```

LotShape: General shape of property

```
Reg    Regular
IR1    Slightly irregular
IR2    Moderately Irregular
IR3    Irregular
```

LandContour: Flatness of the property

```
Lvl    Near Flat/Level
Bnk    Banked - Quick and significant rise from street grade to build
ing
HLS    Hillside - Significant slope from side to side
Low    Depression
```

Utilities: Type of utilities available

```
AllPub    All public Utilities (E,G,W,& S)
NoSewr    Electricity, Gas, and Water (Septic Tank)
NoSeWa    Electricity and Gas Only
ELO    Electricity only
```

LotConfig: Lot configuration

```
Inside    Inside lot
Corner    Corner lot
CulDSac    Cul-de-sac
FR2    Frontage on 2 sides of property
FR3    Frontage on 3 sides of property
```

LandSlope: Slope of property

```
Gtl    Gentle slope
Mod    Moderate Slope
Sev    Severe Slope
```

Neighborhood: Physical locations within Ames city limits

```
Blmngtn    Bloomington Heights
Blueste    Bluestem
BrDale    Briardale
BrkSide    Brookside
ClearCr    Clear Creek
CollgCr    College Creek
Crawfor    Crawford
Edwards    Edwards
Gilbert    Gilbert
IDOTRR    Iowa DOT and Rail Road
MeadowV    Meadow Village
Mitchel    Mitchell
Names    North Ames
NoRidge    Northridge
NPkVill    Northpark Villa
NridgHt    Northridge Heights
NWAmes    Northwest Ames
OldTown    Old Town
SWISU    South & West of Iowa State University
Sawyer    Sawyer
SawyerW    Sawyer West
Somerst    Somerset
StoneBr    Stone Brook
Timber    Timberland
Veenker    Veenker
```

Condition1: Proximity to various conditions

```
Artery    Adjacent to arterial street
Feedr    Adjacent to feeder street
Norm    Normal
RRNn    Within 200' of North-South Railroad
RRAn    Adjacent to North-South Railroad
PosN    Near positive off-site feature--park, greenbelt, etc.
PosA    Adjacent to postive off-site feature
RRNe    Within 200' of East-West Railroad
RRAe    Adjacent to East-West Railroad
```

Condition2: Proximity to various conditions (if more than one is present)

```
Artery    Adjacent to arterial street
Feedr    Adjacent to feeder street
Norm    Normal
RRNn    Within 200' of North-South Railroad
RRAn    Adjacent to North-South Railroad
PosN    Near positive off-site feature--park, greenbelt, etc.
PosA    Adjacent to postive off-site feature
RRNe    Within 200' of East-West Railroad
RRAe    Adjacent to East-West Railroad
```

BldgType: Type of dwelling

```
       1Fam    Single-family Detached
       2FmCon    Two-family Conversion; originally built as one-family dwell
   ing
       Duplx    Duplex
       TwnhsE    Townhouse End Unit
       TwnhsI    Townhouse Inside Unit
```

HouseStyle: Style of dwelling

```
       1Story    One story
       1.5Fin    One and one-half story: 2nd level finished
       1.5Unf    One and one-half story: 2nd level unfinished
       2Story    Two story
       2.5Fin    Two and one-half story: 2nd level finished
       2.5Unf    Two and one-half story: 2nd level unfinished
       SFoyer    Split Foyer
       SLvl    Split Level
```

OverallQual: Rates the overall material and finish of the house

```
       10    Very Excellent
       9    Excellent
       8    Very Good
       7    Good
       6    Above Average
       5    Average
       4    Below Average
       3    Fair
       2    Poor
       1    Very Poor
```

OverallCond: Rates the overall condition of the house

```
       10    Very Excellent
       9    Excellent
       8    Very Good
       7    Good
       6    Above Average
       5    Average
       4    Below Average
       3    Fair
       2    Poor
       1    Very Poor
```

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

```
Flat    Flat
Gable    Gable
Gambrel    Gabrel (Barn)
Hip    Hip
Mansard    Mansard
Shed    Shed
```

RoofMatl: Roof material

```
ClyTile    Clay or Tile
CompShg    Standard (Composite) Shingle
Membran    Membrane
Metal    Metal
Roll    Roll
Tar&Grv    Gravel & Tar
WdShake    Wood Shakes
WdShngl    Wood Shingles
```

Exterior1st: Exterior covering on house

```
AsbShng    Asbestos Shingles
AsphShn    Asphalt Shingles
BrkComm    Brick Common
BrkFace    Brick Face
CBlock    Cinder Block
CemntBd    Cement Board
HdBoard    Hard Board
ImStucc    Imitation Stucco
MetalSd    Metal Siding
Other    Other
Plywood    Plywood
PreCast    PreCast
Stone    Stone
Stucco    Stucco
VinylSd    Vinyl Siding
Wd Sdng    Wood Siding
WdShing    Wood Shingles
```

Exterior2nd: Exterior covering on house (if more than one material)

```
AsbShng    Asbestos Shingles
AsphShn    Asphalt Shingles
BrkComm    Brick Common
BrkFace    Brick Face
CBlock    Cinder Block
CemntBd    Cement Board
HdBoard    Hard Board
ImStucc    Imitation Stucco
MetalSd    Metal Siding
Other    Other
Plywood    Plywood
PreCast    PreCast
Stone    Stone
Stucco    Stucco
VinylSd    Vinyl Siding
Wd Sdng    Wood Siding
WdShing    Wood Shingles
```

MasVnrType: Masonry veneer type

```
BrkCmn    Brick Common
BrkFace    Brick Face
CBlock    Cinder Block
None    None
Stone    Stone
```

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

```
Ex    Excellent
Gd    Good
TA    Average/Typical
Fa    Fair
Po    Poor
```

ExterCond: Evaluates the present condition of the material on the exterior

```
Ex    Excellent
Gd    Good
TA    Average/Typical
Fa    Fair
Po    Poor
```

Foundation: Type of foundation

```
BrkTil    Brick & Tile
CBlock    Cinder Block
PConc    Poured Contrete
Slab    Slab
Stone    Stone
Wood    Wood
```

BsmtQual: Evaluates the height of the basement

```
Ex    Excellent (100+ inches)
Gd    Good (90-99 inches)
TA    Typical (80-89 inches)
Fa    Fair (70-79 inches)
Po    Poor (<70 inches
NA    No Basement
```

BsmtCond: Evaluates the general condition of the basement

```
Ex    Excellent
Gd    Good
TA    Typical - slight dampness allowed
Fa    Fair - dampness or some cracking or settling
Po    Poor - Severe cracking, settling, or wetness
NA    No Basement
```

BsmtExposure: Refers to walkout or garden level walls

```
Gd    Good Exposure
Av    Average Exposure (split levels or foyers typically score averag
e or above)
Mn    Mimimum Exposure
No    No Exposure
NA    No Basement
```

BsmtFinType1: Rating of basement finished area

```
GLQ    Good Living Quarters
ALQ    Average Living Quarters
BLQ    Below Average Living Quarters
Rec    Average Rec Room
LwQ    Low Quality
Unf    Unfinshed
NA    No Basement
```

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

```
GLQ    Good Living Quarters
ALQ    Average Living Quarters
BLQ    Below Average Living Quarters
Rec    Average Rec Room
LwQ    Low Quality
Unf    Unfinshed
NA   No Basement
```

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

```
Floor    Floor Furnace
GasA    Gas forced warm air furnace
GasW    Gas hot water or steam heat
Grav    Gravity furnace
OthW    Hot water or steam heat other than gas
Wall    Wall furnace
```

HeatingQC: Heating quality and condition

```
Ex    Excellent
Gd    Good
TA    Average/Typical
Fa    Fair
Po    Poor
```

CentralAir: Central air conditioning

```
N    No
Y    Yes
```

Electrical: Electrical system

```
SBrkr    Standard Circuit Breakers & Romex
FuseA    Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF    60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP    60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix    Mixed
```

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

```
   Ex    Excellent
   Gd    Good
   TA    Typical/Average
   Fa    Fair
   Po    Poor
```

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

```
   Typ    Typical Functionality
   Min1    Minor Deductions 1
   Min2    Minor Deductions 2
   Mod    Moderate Deductions
   Maj1    Major Deductions 1
   Maj2    Major Deductions 2
   Sev    Severely Damaged
   Sal    Salvage only
```

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

```
   Ex    Excellent - Exceptional Masonry Fireplace
   Gd    Good - Masonry Fireplace in main level
   TA    Average - Prefabricated Fireplace in main living area or Masonr
y Fireplace in basement
   Fa    Fair - Prefabricated Fireplace in basement
   Po    Poor - Ben Franklin Stove
   NA    No Fireplace
```

GarageType: Garage location

```
   2Types    More than one type of garage
   Attchd    Attached to home
   Basment    Basement Garage
   BuiltIn    Built-In (Garage part of house - typically has room above
 garage)
   CarPort    Car Port
   Detchd    Detached from home
   NA    No Garage
```

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

```
Fin    Finished
RFn    Rough Finished
Unf    Unfinished
NA   No Garage
```

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

GarageCond: Garage condition

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

PavedDrive: Paved driveway

```
Y   Paved
P   Partial Pavement
N   Dirt/Gravel
```

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

```
Ex    Excellent
Gd    Good
TA    Average/Typical
Fa    Fair
NA    No Pool
```

Fence: Fence quality

```
GdPrv    Good Privacy
MnPrv    Minimum Privacy
GdWo    Good Wood
MnWw    Minimum Wood/Wire
NA    No Fence
```

MiscFeature: Miscellaneous feature not covered in other categories

```
Elev    Elevator
Gar2    2nd Garage (if not described in garage section)
Othr    Other
Shed    Shed (over 100 SF)
TenC    Tennis Court
NA    None
```

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

```
WD     Warranty Deed - Conventional
CWD    Warranty Deed - Cash
VWD    Warranty Deed - VA Loan
New    Home just constructed and sold
COD    Court Officer Deed/Estate
Con    Contract 15% Down payment regular terms
ConLw    Contract Low Down payment and low interest
ConLI    Contract Low Interest
ConLD    Contract Low Down
Oth    Other
```

SaleCondition: Condition of sale

```
   Normal    Normal Sale
   Abnorml    Abnormal Sale -  trade, foreclosure, short sale
   AdjLand    Adjoining Land Purchase
   Alloca    Allocation - two linked properties with separate deeds, typ
ically condo with a garage unit
   Family    Sale between family members
   Partial    Home was not completed when last assessed (associated with
New Homes)
```

In [ ]:

Impoting the necessary Libraries

In [1]:
```
#import libraries fro the dataset exploration
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve,auc
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
import pandas as pd
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import warnings  # Ignoring Warnings
warnings.filterwarnings("ignore")
from scipy import stats
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

Loading the data

In [2]:
```
#reading the csv file
test =pd.read_csv("C:\\Users\\Roxton\\Desktop\\test.csv")
train =pd.read_csv("C:\\Users\\Roxton\\Desktop\\train.csv")
```

understanding the number of rows and columns in the datasets

In [3]:
```
test.shape
```
Out[3]: (1459, 80)

In [4]:
```
train.shape
```
Out[4]: (1460, 81)

Data Description

In [5]:
```
test.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
Id	1459.0	2190.000000	421.321334	1461.0	1825.50	2190.0	2554.50	2919.0
MSSubClass	1459.0	57.378341	42.746880	20.0	20.00	50.0	70.00	190.0
LotFrontage	1232.0	68.580357	22.376841	21.0	58.00	67.0	80.00	200.0
LotArea	1459.0	9819.161069	4955.517327	1470.0	7391.00	9399.0	11517.50	56600.0
OverallQual	1459.0	6.078821	1.436812	1.0	5.00	6.0	7.00	10.0
OverallCond	1459.0	5.553804	1.113740	1.0	5.00	5.0	6.00	9.0
YearBuilt	1459.0	1971.357779	30.390071	1879.0	1953.00	1973.0	2001.00	2010.0
YearRemodAdd	1459.0	1983.662783	21.130467	1950.0	1963.00	1992.0	2004.00	2010.0
MasVnrArea	1444.0	100.709141	177.625900	0.0	0.00	0.0	164.00	1290.0
BsmtFinSF1	1458.0	439.203704	455.268042	0.0	0.00	350.5	753.50	4010.0
BsmtFinSF2	1458.0	52.619342	176.753926	0.0	0.00	0.0	0.00	1526.0
BsmtUnfSF	1458.0	554.294925	437.260486	0.0	219.25	460.0	797.75	2140.0
TotalBsmtSF	1458.0	1046.117970	442.898624	0.0	784.00	988.0	1305.00	5095.0
1stFlrSF	1459.0	1156.534613	398.165820	407.0	873.50	1079.0	1382.50	5095.0
2ndFlrSF	1459.0	325.967786	420.610226	0.0	0.00	0.0	676.00	1862.0
LowQualFinSF	1459.0	3.543523	44.043251	0.0	0.00	0.0	0.00	1064.0
GrLivArea	1459.0	1486.045922	485.566099	407.0	1117.50	1432.0	1721.00	5095.0
BsmtFullBath	1457.0	0.434454	0.530648	0.0	0.00	0.0	1.00	3.0
BsmtHalfBath	1457.0	0.065202	0.252468	0.0	0.00	0.0	0.00	2.0
FullBath	1459.0	1.570939	0.555190	0.0	1.00	2.0	2.00	4.0
HalfBath	1459.0	0.377656	0.503017	0.0	0.00	0.0	1.00	2.0
BedroomAbvGr	1459.0	2.854010	0.829788	0.0	2.00	3.0	3.00	6.0
KitchenAbvGr	1459.0	1.042495	0.208472	0.0	1.00	1.0	1.00	2.0
TotRmsAbvGrd	1459.0	6.385195	1.508895	3.0	5.00	6.0	7.00	15.0
Fireplaces	1459.0	0.581220	0.647420	0.0	0.00	0.0	1.00	4.0
GarageYrBlt	1381.0	1977.721217	26.431175	1895.0	1959.00	1979.0	2002.00	2207.0
GarageCars	1458.0	1.766118	0.775945	0.0	1.00	2.0	2.00	5.0
GarageArea	1458.0	472.768861	217.048611	0.0	318.00	480.0	576.00	1488.0
WoodDeckSF	1459.0	93.174777	127.744882	0.0	0.00	0.0	168.00	1424.0
OpenPorchSF	1459.0	48.313914	68.883364	0.0	0.00	28.0	72.00	742.0
EnclosedPorch	1459.0	24.243317	67.227765	0.0	0.00	0.0	0.00	1012.0
3SsnPorch	1459.0	1.794380	20.207842	0.0	0.00	0.0	0.00	360.0
ScreenPorch	1459.0	17.064428	56.609763	0.0	0.00	0.0	0.00	576.0
PoolArea	1459.0	1.744345	30.491646	0.0	0.00	0.0	0.00	800.0

	count	mean	std	min	25%	50%	75%	max
MiscVal	1459.0	58.167923	630.806978	0.0	0.00	0.0	0.00	17000.0
MoSold	1459.0	6.104181	2.722432	1.0	4.00	6.0	8.00	12.0
YrSold	1459.0	2007.769705	1.301740	2006.0	2007.00	2008.0	2009.00	2010.0

In [6]:
```
d=train.describe().T
```

Finding the datatypes in the datasets

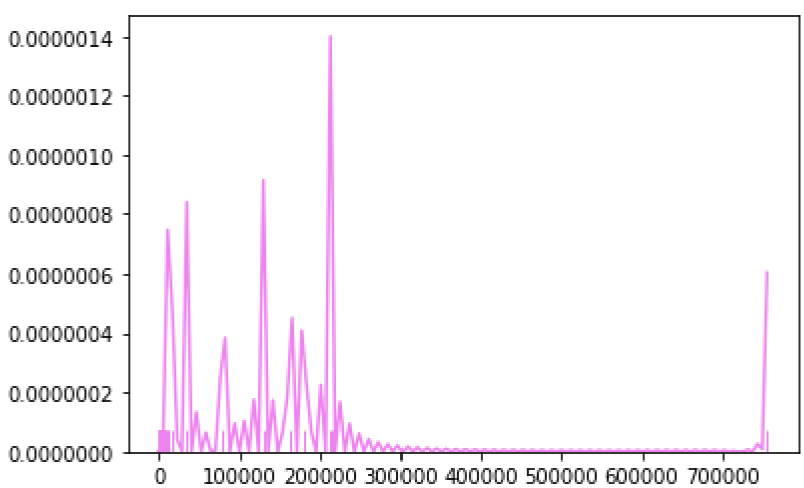
In [7]:
```
train.dtypes
```

Out[7]:
```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 81, dtype: object
```

Statistical distribution of the datasets

In [8]:
```
#statistical distibution of train data in a graph
sns.distplot(d, hist=False, bins=20,rug=True,color ='violet')
```

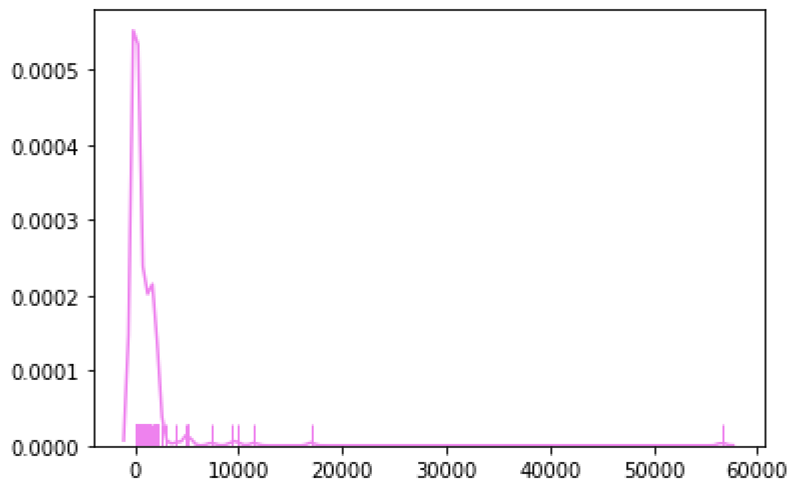
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x332d31ca88>



In [9]:
```
v=test.describe().T
```

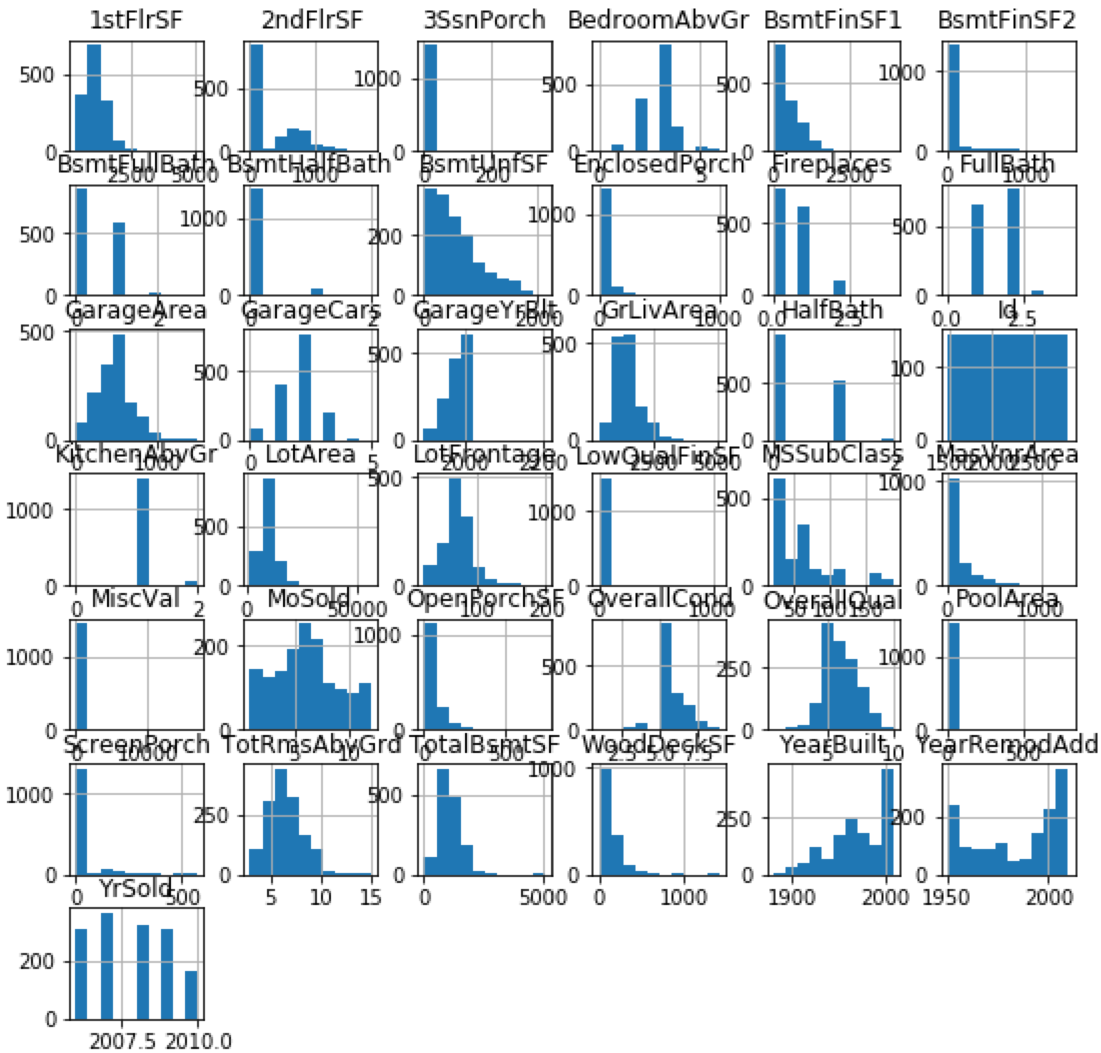
In [10]:
```
#statistical distibution of taring data in a graph
sns.distplot(v, hist=False, bins=20,rug=True,color ='violet')
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x332dac7c08>



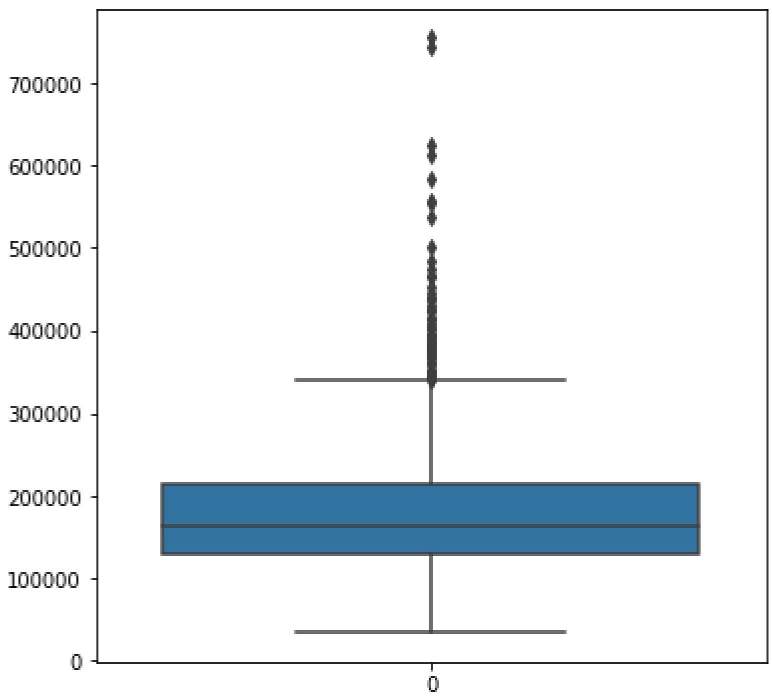
histograms

In [11]:
```
#histogramsfor individual columns
bargraphs =test.hist(figsize=(9,9))
plt.show()
```

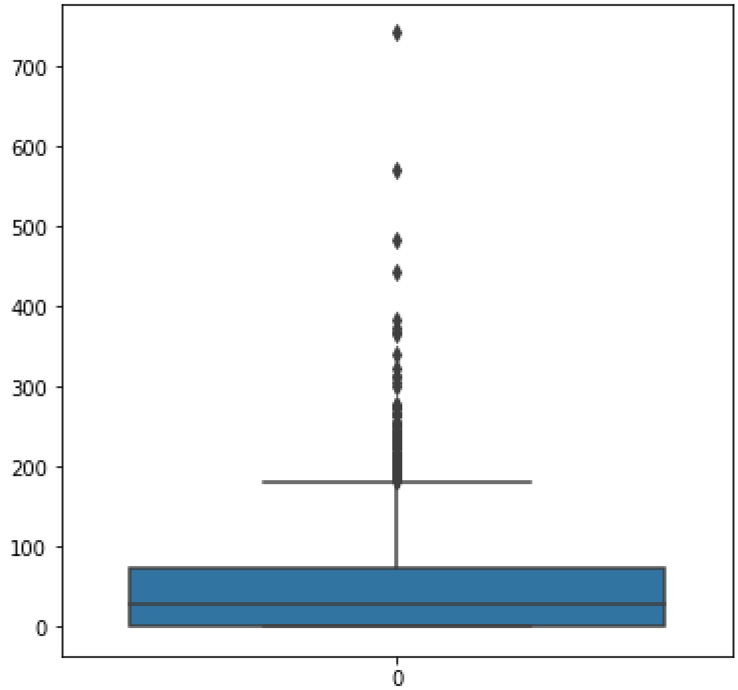


Boxplots

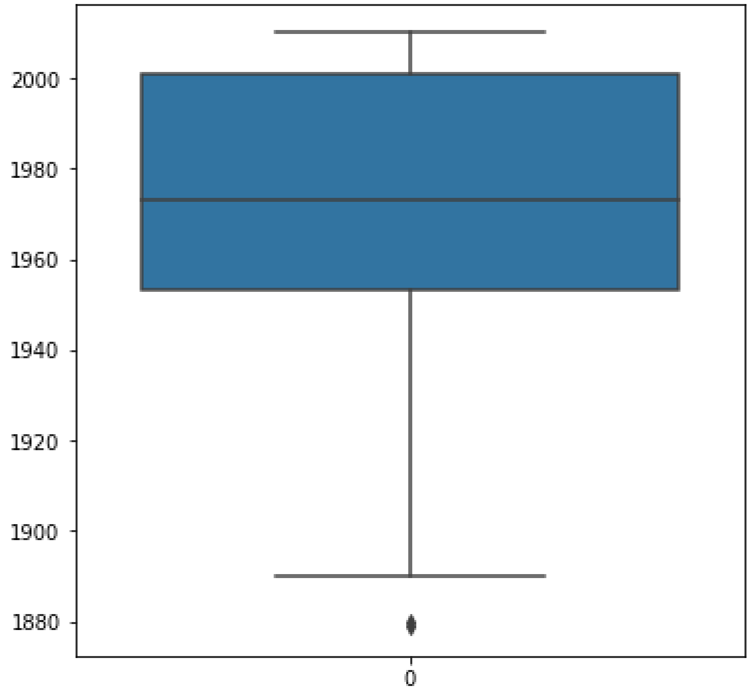
In [12]:
```
#boxplot of the saleprice
plt.figure(figsize=(6,6))
sns.boxplot(data=train['SalePrice'])
plt.show()
```



In [13]:
```
#boxplot of the OpenPorch
plt.figure(figsize=(6,6))
sns.boxplot(data=test['OpenPorchSF'])
plt.show()
```



In [14]:
```
#boxplot of the yearbuilt
plt.figure(figsize=(6,6))
sns.boxplot(data=test['YearBuilt'])
plt.show()
```



Checking for null values filling the columns of the missing values and visualization of missing values

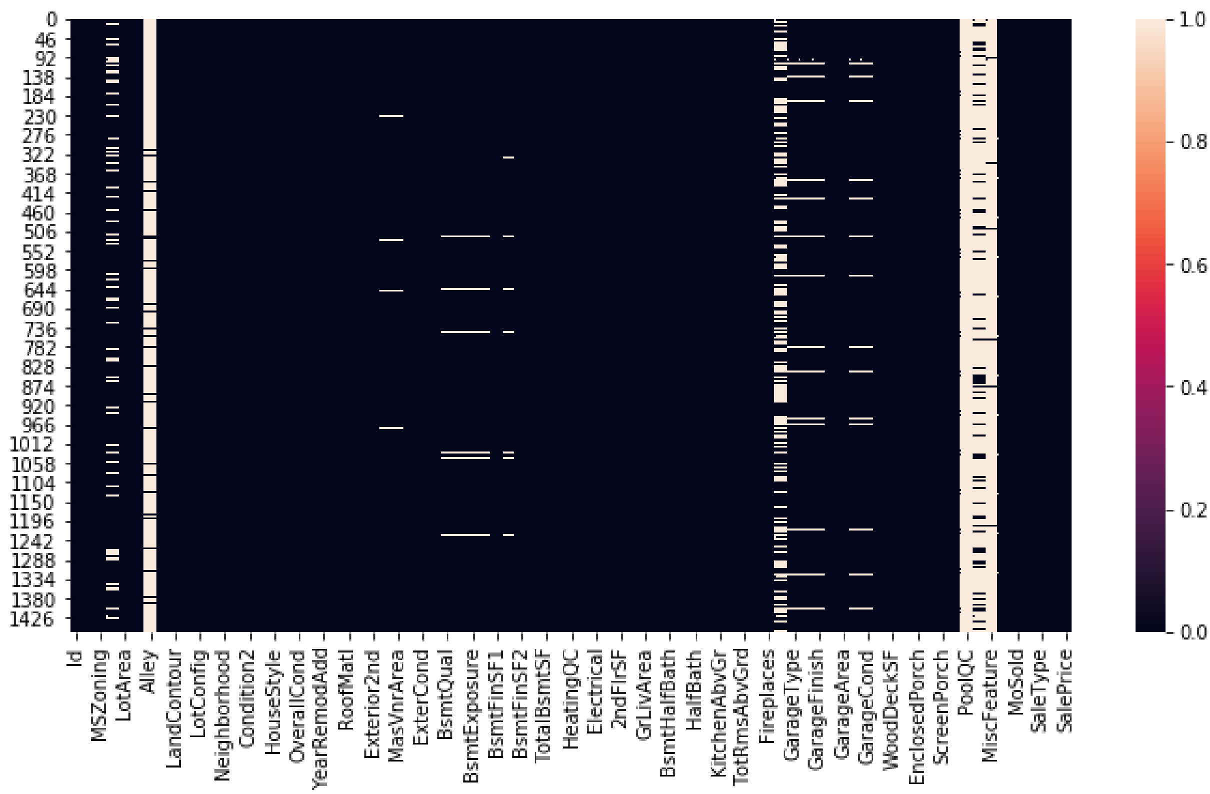
In [15]:
```
#Let's check if the data set has any missing values.
train.columns[train.isnull().any()]
```

Out[15]:
```
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'Electrical', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageQual', 'GarageCond', 'PoolQC', 'Fence',
       'MiscFeature'],
      dtype='object')
```

In [16]:
```
#Let's check if the data set has any missing values.
test.columns[test.isnull().any()]
```

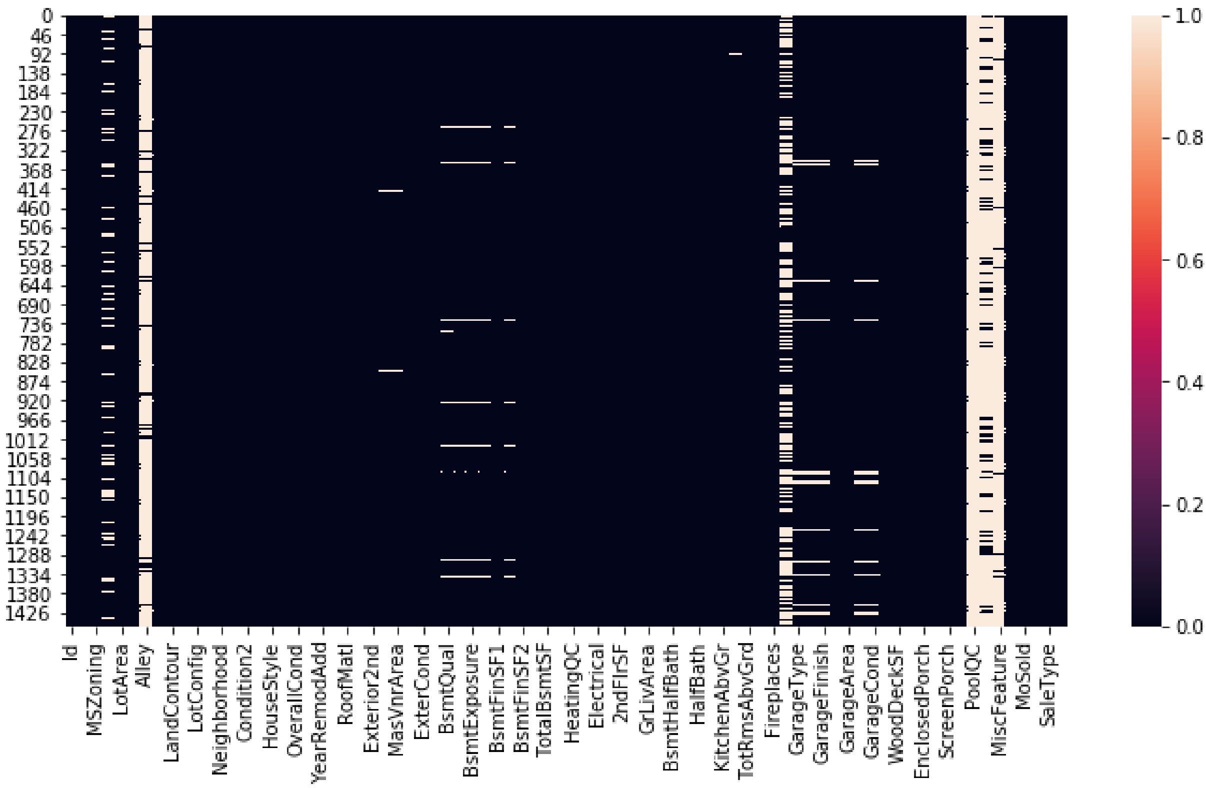
Out[16]:
```
Index(['MSZoning', 'LotFrontage', 'Alley', 'Utilities', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath',
       'BsmtHalfBath', 'KitchenQual', 'Functional', 'FireplaceQu',
       'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea',
       'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeature',
       'SaleType'],
      dtype='object')
```

In [17]:
```
#plot of missing values
plt.figure(figsize=(12, 6))
sns.heatmap(train.isnull())
plt.show()
```



Plot of missing values

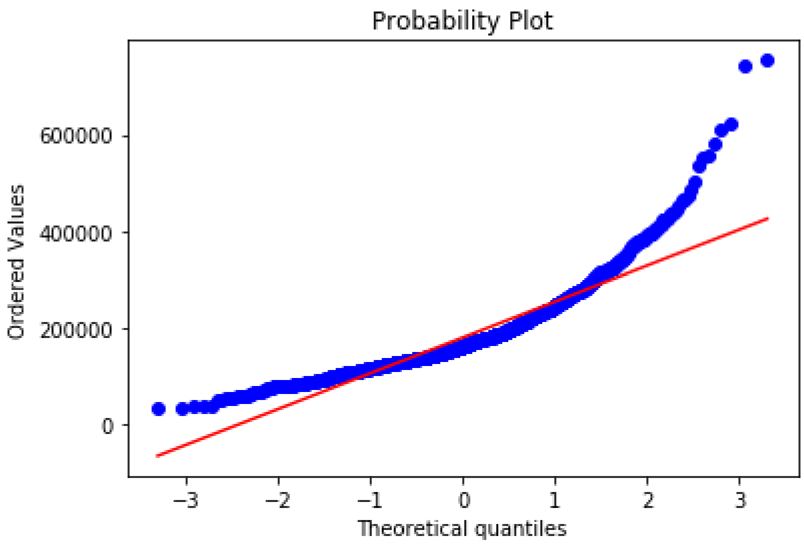
In [18]:
```
#plot of missing values
plt.figure(figsize=(12, 6))
sns.heatmap(test.isnull())
plt.show()
```



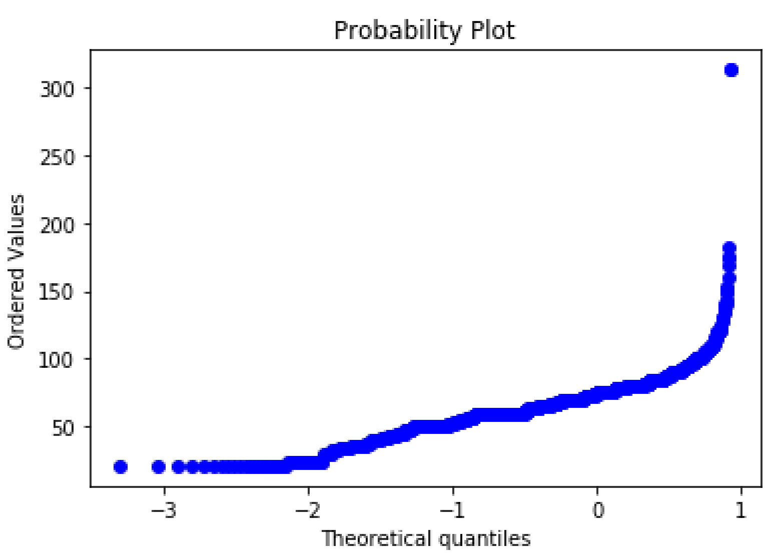
Thete are fewer missing values in the test dataset as compared to the training dataset

Probability Plots of the datasets

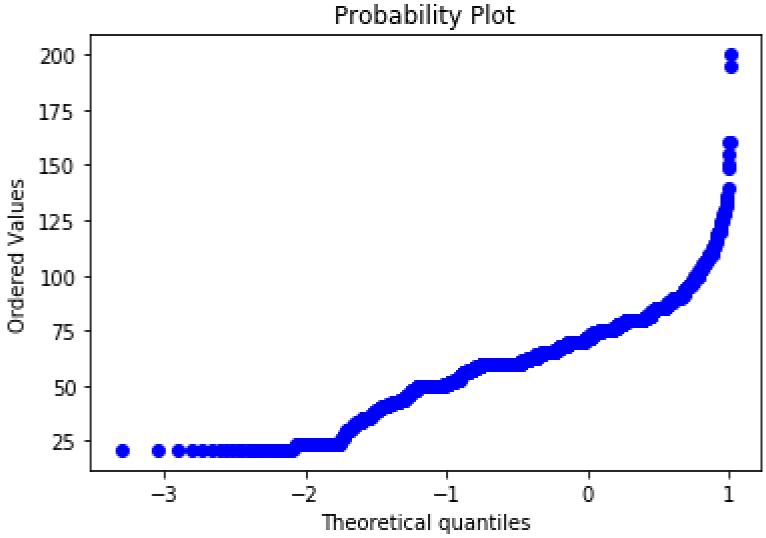
In [19]:
```
#probability of the target variable

fig = plt.figure()
stats.probplot(train['SalePrice'], plot=plt)
plt.show()
```



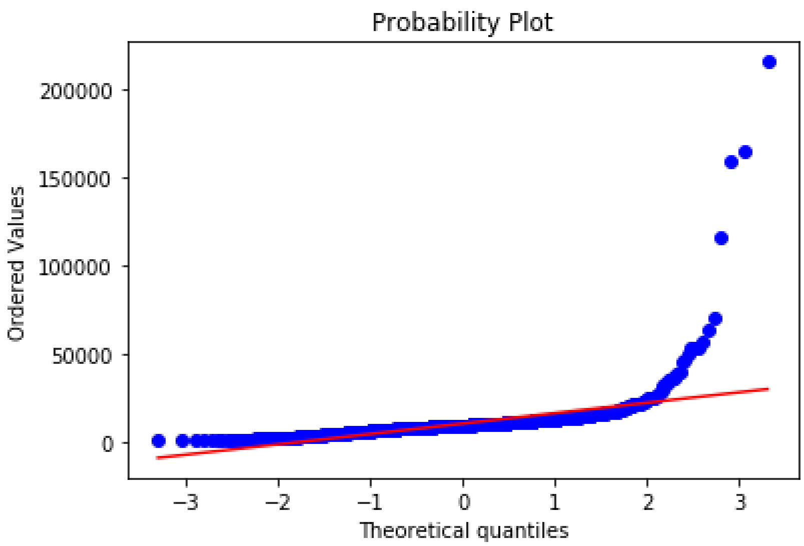
In [20]:
```
fig = plt.figure()
stats.probplot(train['LotFrontage'], plot=plt)
plt.show()
```



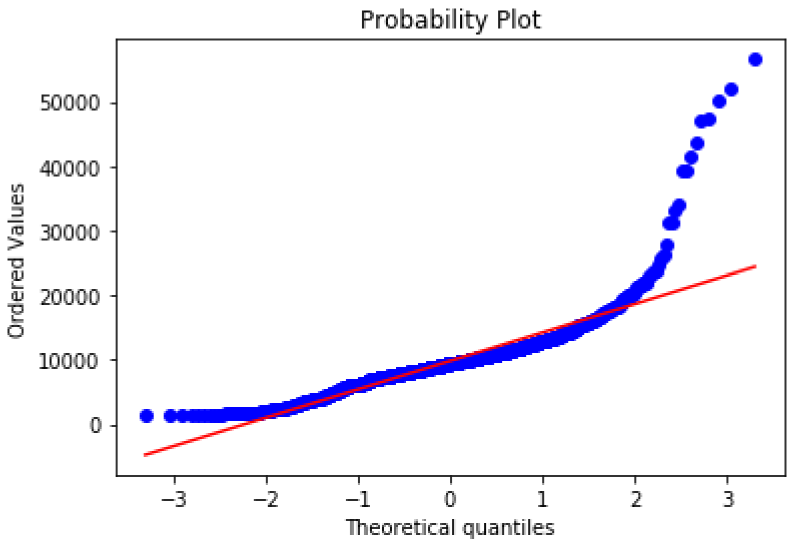
In [21]:
```
fig = plt.figure()
stats.probplot(test['LotFrontage'], plot=plt)
plt.show()
```



In [22]:
```
#probability of the lotArea variable

fig = plt.figure()
stats.probplot(train['LotArea'], plot=plt)
plt.show()
```



In [23]:
```
fig = plt.figure()
stats.probplot(test['LotArea'], plot=plt)
plt.show()
```

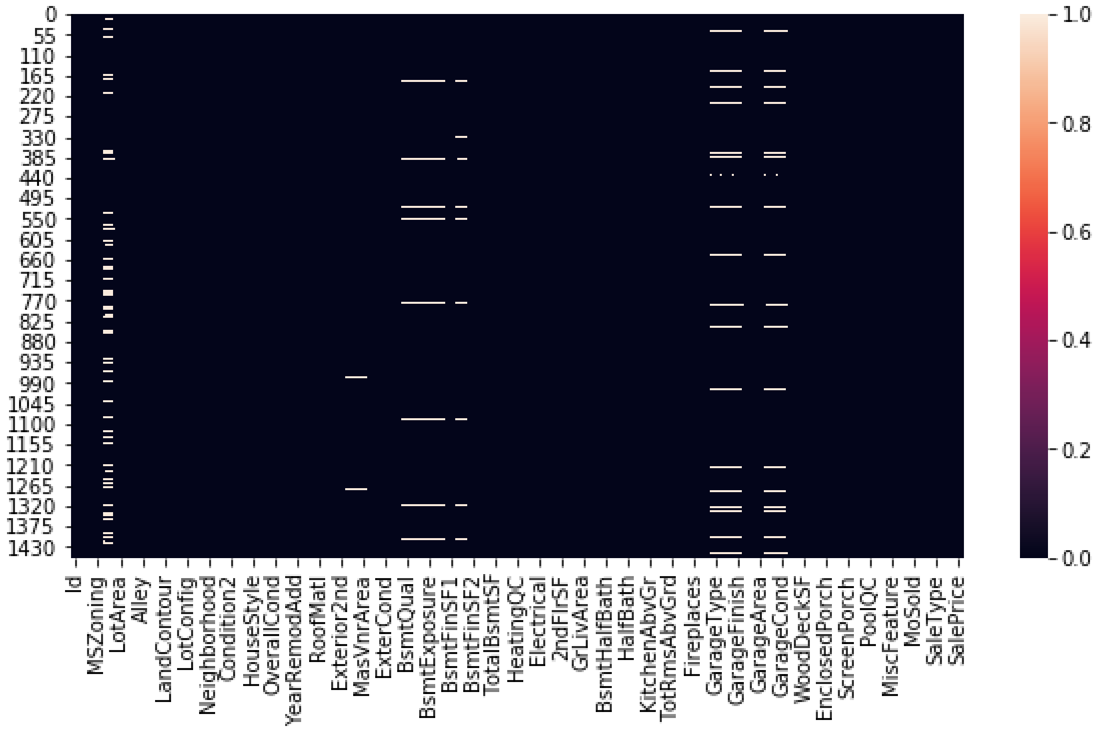


Imputation of the columns

In [24]:
```
#imputation lets fill the missing values in the coulmns with None in the train se
train['MiscFeature'] = train['MiscFeature'].fillna(int(0))
train['Alley'] = train['Alley'].fillna(int(0))
train['Fence'] = train['Fence'].fillna(int(0))
train['FireplaceQu'] = train['FireplaceQu'].fillna(int(0))
train['PoolQC'] = train['PoolQC'].fillna(int(0))
test['MiscFeature'] = test['MiscFeature'].fillna(int(0))
test['Alley'] = test['Alley'].fillna(int(0))
test['Fence'] = test['Fence'].fillna(int(0))
test['FireplaceQu'] = test['FireplaceQu'].fillna(int(0))
test['PoolQC'] = test['PoolQC'].fillna(int(0))
```

In [25]:
```
#Checking there is any null value or not
plt.figure(figsize=(10, 5))
sns.heatmap(train.isnull())
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x3331ff5fc8>



we still have some null columns according to the heatmap

In [26]:
```
#Let's check if the data set has any missing values.
train.columns[train.isnull().any()]
```

Out[26]:
```
Index(['LotFrontage', 'MasVnrType', 'MasVnrArea', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Electrical',
       'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageQual',
       'GarageCond'],
      dtype='object')
```

In [27]:
```
for columns in ('BsmtFinType2', 'BsmtExposure', 'BsmtFinType1', 'BsmtCond', 'Bsmt
    train[columns] = train[columns].fillna(int(0))


for columns in ('BsmtFinType2', 'BsmtExposure', 'BsmtFinType1', 'BsmtCond', 'Bsmt
    test[columns] = test[columns].fillna(int(0))
```

In [28]:
```
#lets fill these columns with zero tis is for easy prediction fill the empty colu
train['GarageType'] = train['GarageType'].fillna(int(0))
train['GarageFinish'] = train['GarageFinish'].fillna(int(0))
train['GarageQual'] = train['GarageQual'].fillna(int(0))
train['GarageCond'] = train['GarageCond'].fillna(int(0))
train['GarageYrBlt'] = train['GarageYrBlt'].fillna(int(0))
train['GarageArea'] = train['GarageArea'].fillna(int(0))
train['GarageCars'] = train['GarageCars'].fillna(int(0))

test['GarageType'] = train['GarageType'].fillna(int(0))
test['GarageFinish'] = train['GarageFinish'].fillna(int(0))
test['GarageQual'] = train['GarageQual'].fillna(int(0))
test['GarageCond'] = train['GarageCond'].fillna(int(0))
test['GarageYrBlt'] = train['GarageYrBlt'].fillna(int(0))
test['GarageArea'] = train['GarageArea'].fillna(int(0))
test['GarageCars'] = train['GarageCars'].fillna(int(0))
```

In [29]:
```
#Let's check if the data set has any missing values.
train.columns[train.isnull().any()]
```

Out[29]:
```
Index(['LotFrontage', 'MasVnrType', 'MasVnrArea', 'Electrical'], dtype='objec
t')
```

In [30]:
```
train['MasVnrArea'] = train['MasVnrArea'].fillna(int(0))
train['MasVnrType'] = train['MasVnrType'].fillna(int(0))
train['Electrical'] = train['Electrical'].fillna(train['Electrical']).mode()[0]
test['MasVnrArea'] = train['MasVnrArea'].fillna(int(0))
test['MasVnrType'] = train['MasVnrType'].fillna(int(0))
test['Electrical'] = train['Electrical'].fillna(train['Electrical']).mode()[0]
```

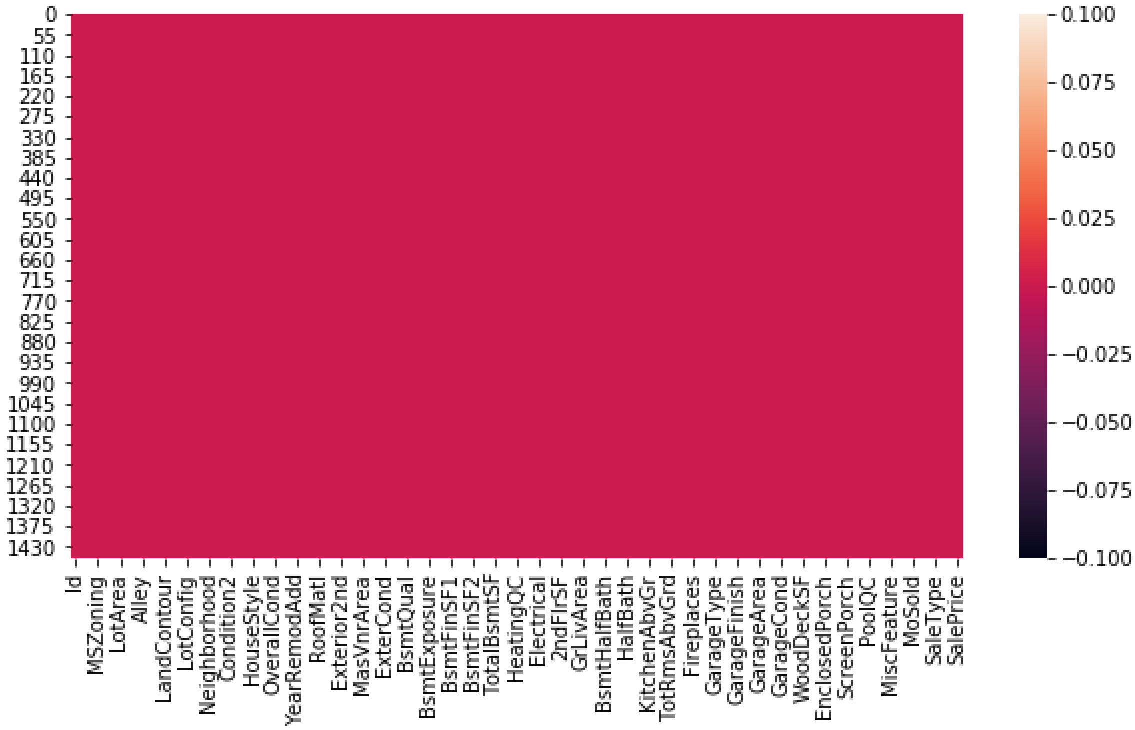
In [31]:
```
#group LotFrontage with neighborhood and fill it with mean of the neigbourhood
train['LotFrontage'] = train.groupby("Neighborhood")["LotFrontage"].transform(
    lambda x: x.fillna(x.std()))
test['LotFrontage'] = train.groupby("Neighborhood")["LotFrontage"].transform(
    lambda x: x.fillna(x.std()))
```

In [32]:
```
#lets check for missing values again
train.columns[train.isnull().any()]
```

Out[32]:
```
Index([], dtype='object')
```

In [33]:
```
#Checking there is any null value or not
plt.figure(figsize=(10, 5))
sns.heatmap(train.isnull())
```

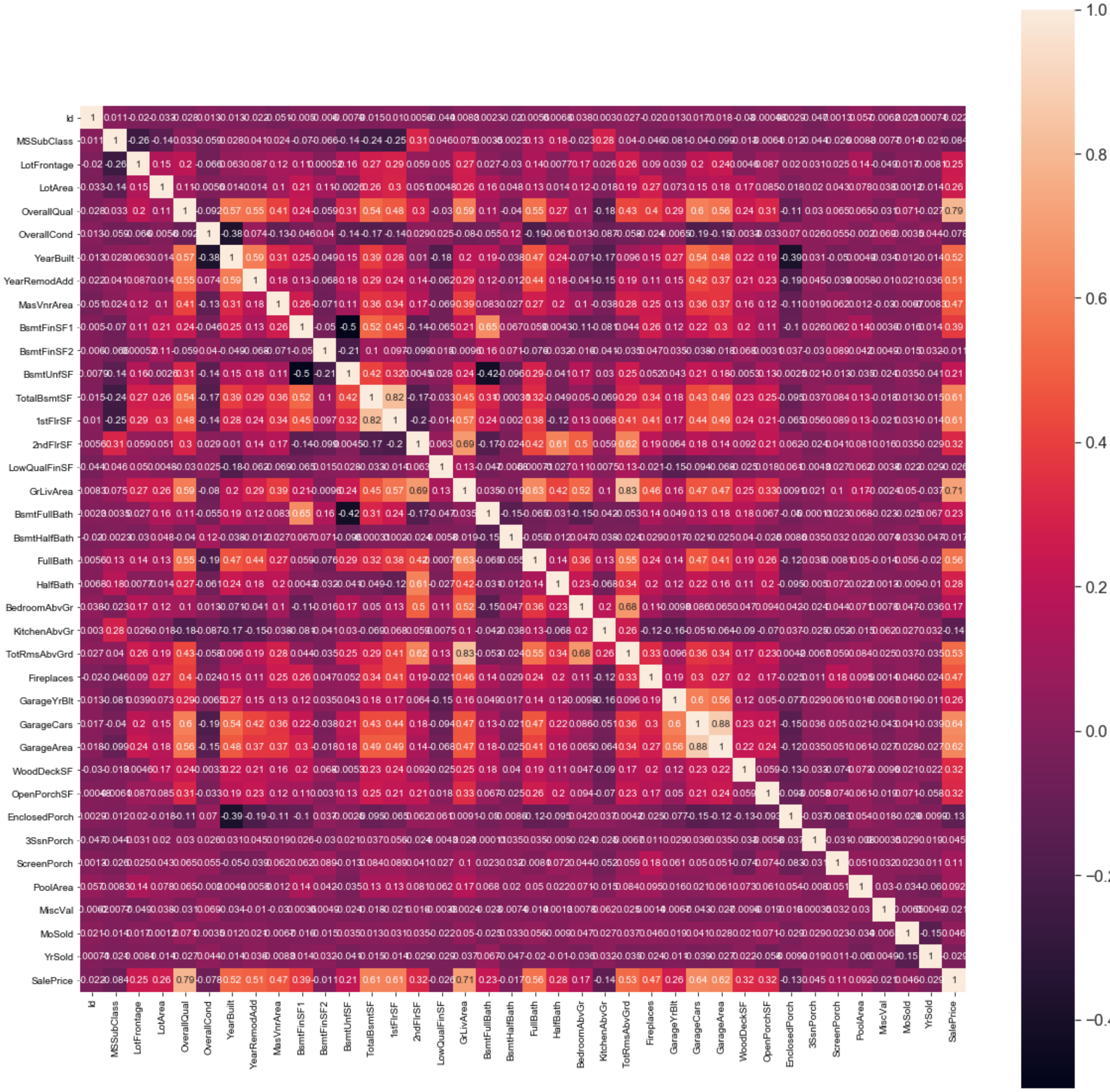
Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x332e8afe48>
```



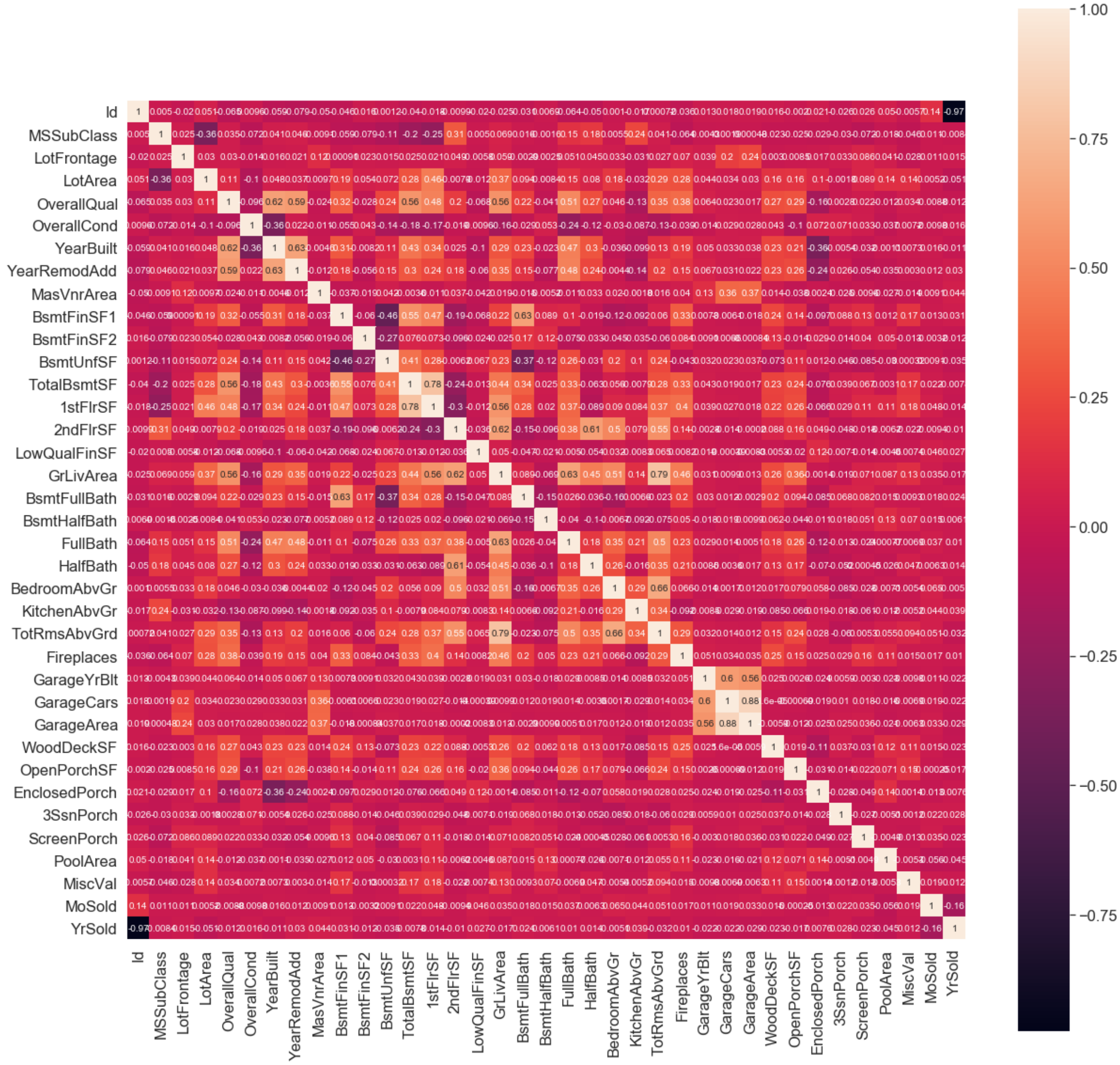
There are no more null values , So we have finished filling in the null columns with some values like none and int(0)

HeatMaps

In [34]:
```
#heatmap for train corollations
fig = plt.subplots(figsize = (20,20))
sns.set(font_scale=1.5)
sns.heatmap(train.corr(),square = True,cbar=True,annot=True,annot_kws={'size': 10
plt.show()
```



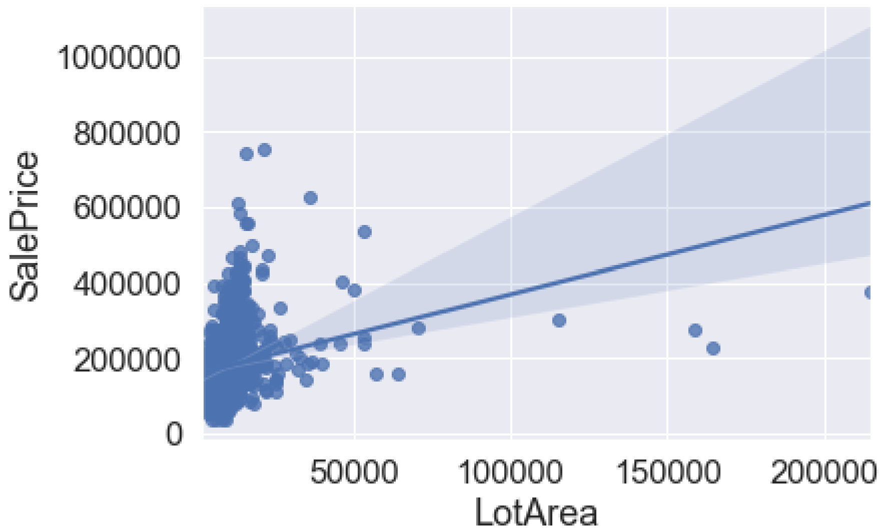
In [35]:
```
#heatmap of the test dataset
fig = plt.subplots(figsize = (20,20))
sns.set(font_scale=1.5)
sns.heatmap(test.corr(),square = True,cbar=True,annot=True,annot_kws={'size': 10
plt.show()
```



Regression Plots

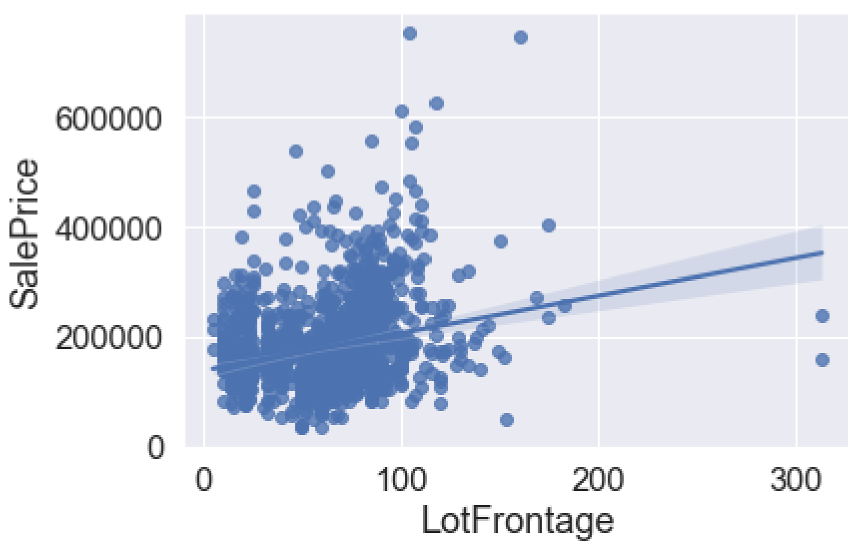
In [36]:
```
sns.regplot(x='LotArea',y='SalePrice', data=train)
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x332e8ceac8>



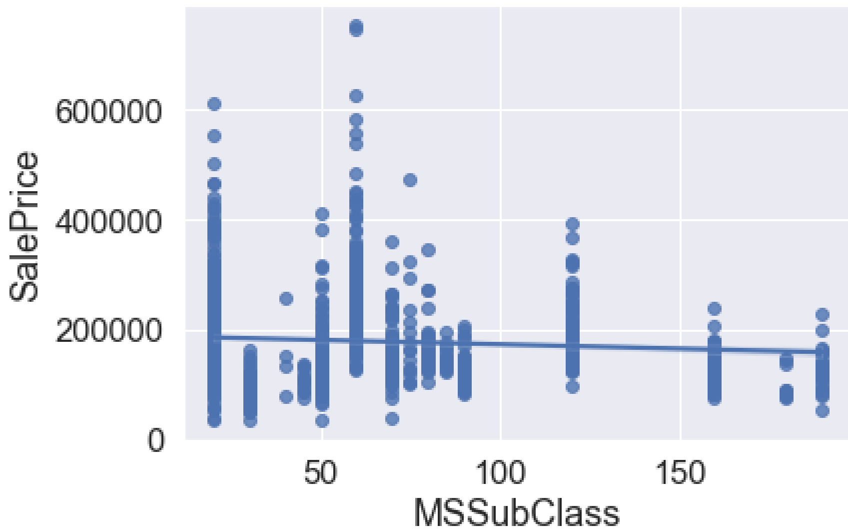
In [37]:
```
sns.regplot(x='LotFrontage',y='SalePrice', data=train)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x332ed7cb08>



In [38]:
```
sns.regplot(x='MSSubClass',y='SalePrice', data=train)
```

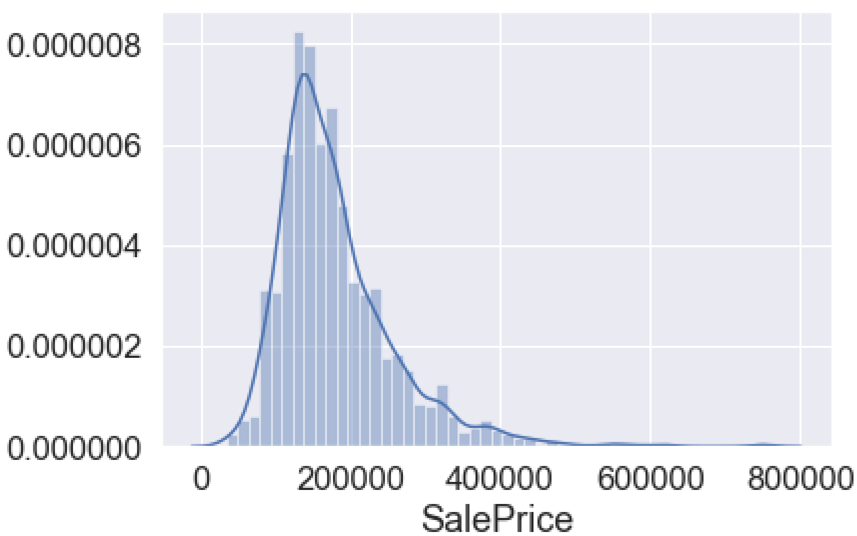
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x33309a1388>



Distribution plots

In [39]:
```
#saleprice distribution
sns.distplot(train['SalePrice'])
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x33312e6548>

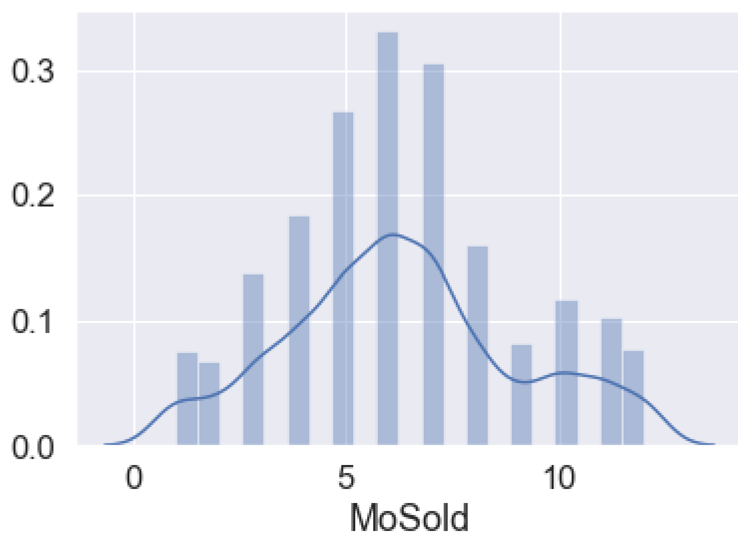


In [40]:
```
#year of sale  distribution
sns.distplot(train['YrSold'])
```

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x3330953508>

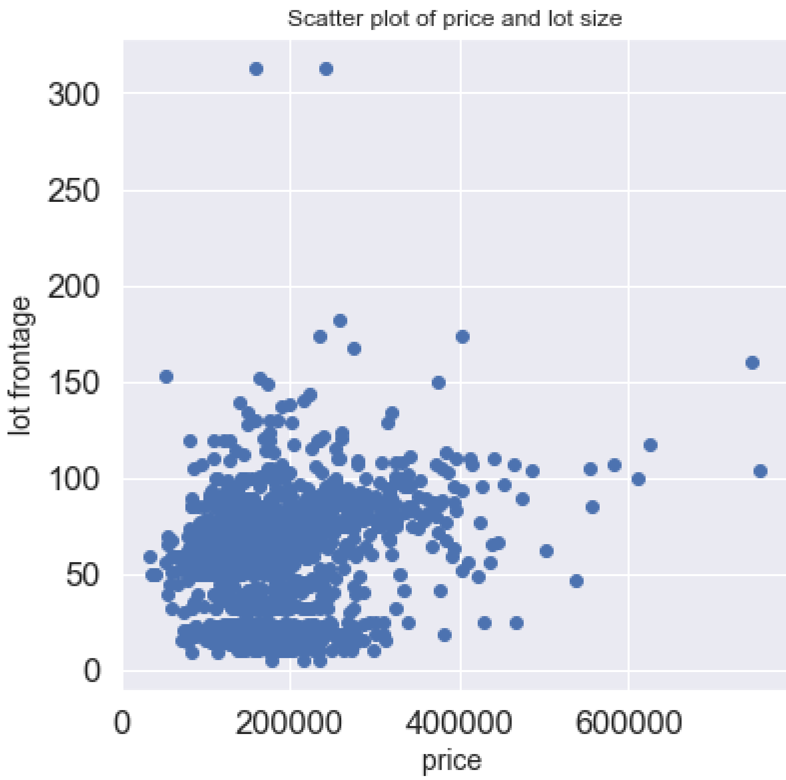
In [41]:
```
#month sold distribution
sns.distplot(train['MoSold'])
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x332eb85388>

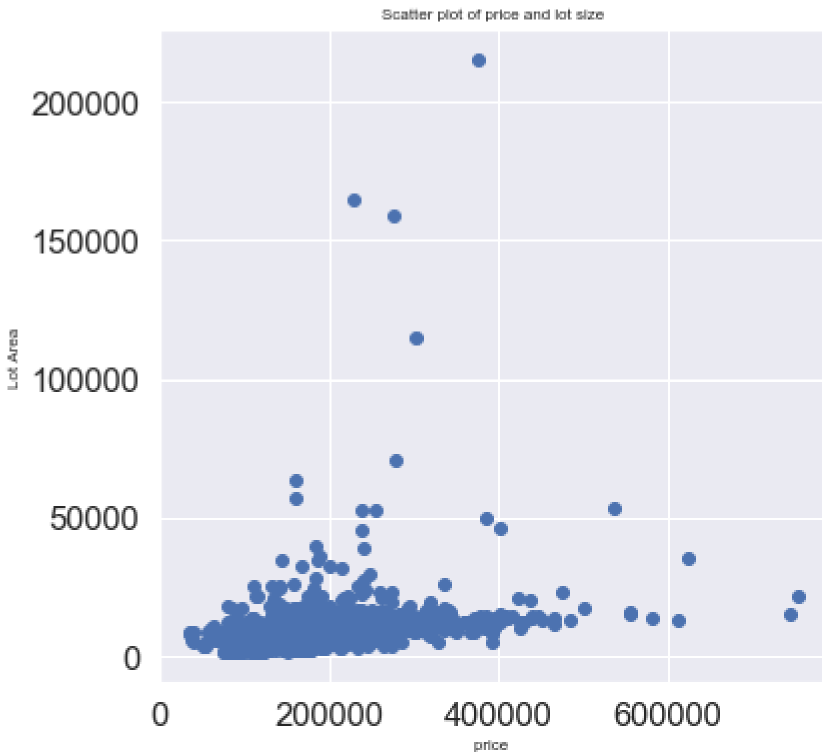


Scatter Plots

In [42]:
```
plt.figure(figsize=(6, 6))
# plot two values price per lot size
plt.scatter(train.SalePrice, train.LotFrontage)
plt.xlabel("price ", fontsize=14)
plt.ylabel("lot frontage", fontsize=14)
plt.title("Scatter plot of price and lot size",fontsize=12)
plt.show()
```

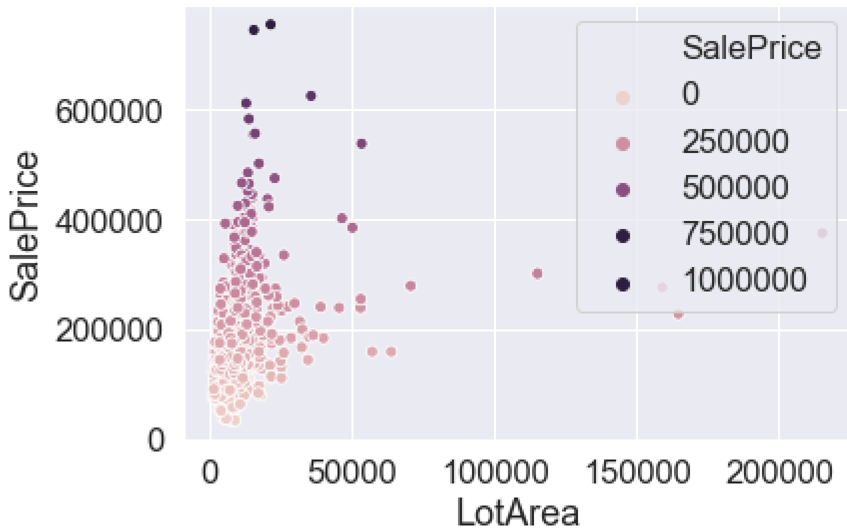


In [43]:
```
plt.figure(figsize=(6, 6))
# plot two values price per lot size
plt.scatter(train.SalePrice, train.LotArea)
plt.xlabel("price ", fontsize=8)
plt.ylabel("Lot Area", fontsize=8)
plt.title("Scatter plot of price and lot size",fontsize=8)
plt.show()
```



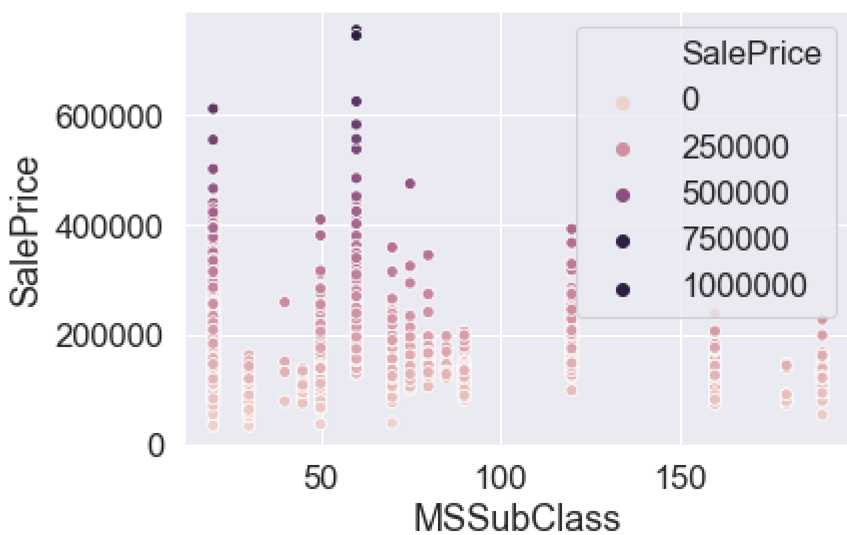
In [44]:
```
sns.scatterplot(x='LotArea',y='SalePrice',data=train,hue='SalePrice')
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x3330e7e948>



In [45]:
```
sns.scatterplot(x='MSSubClass',y='SalePrice',data=train,hue='SalePrice')
```

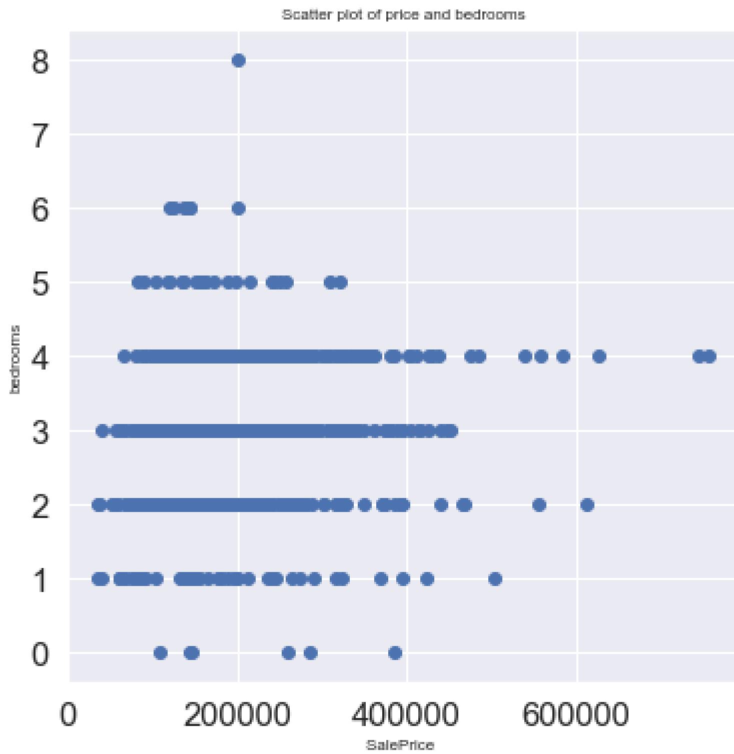
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x3330e4a448>



In [46]:
```
sns.scatterplot(x='OverallCond',y='SalePrice',data=train,hue='SalePrice')
```

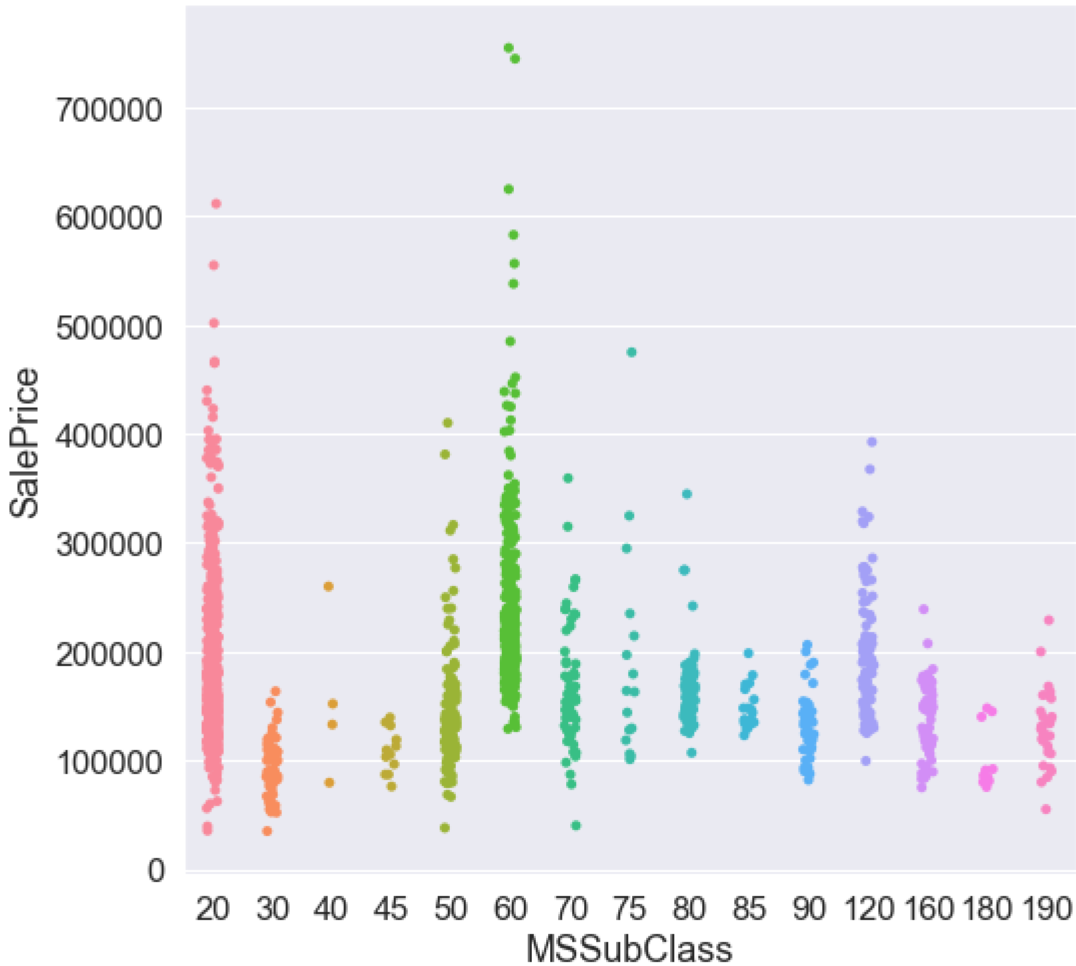
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x3330fa2048>

In [47]:
```
plt.figure(figsize=(6, 6))
# plot two values price per lot size
plt.scatter(train.SalePrice, train.BedroomAbvGr)
plt.xlabel("SalePrice ", fontsize=8)
plt.ylabel("bedrooms", fontsize=8)
plt.title("Scatter plot of price and bedrooms",fontsize=8)
plt.show()
```



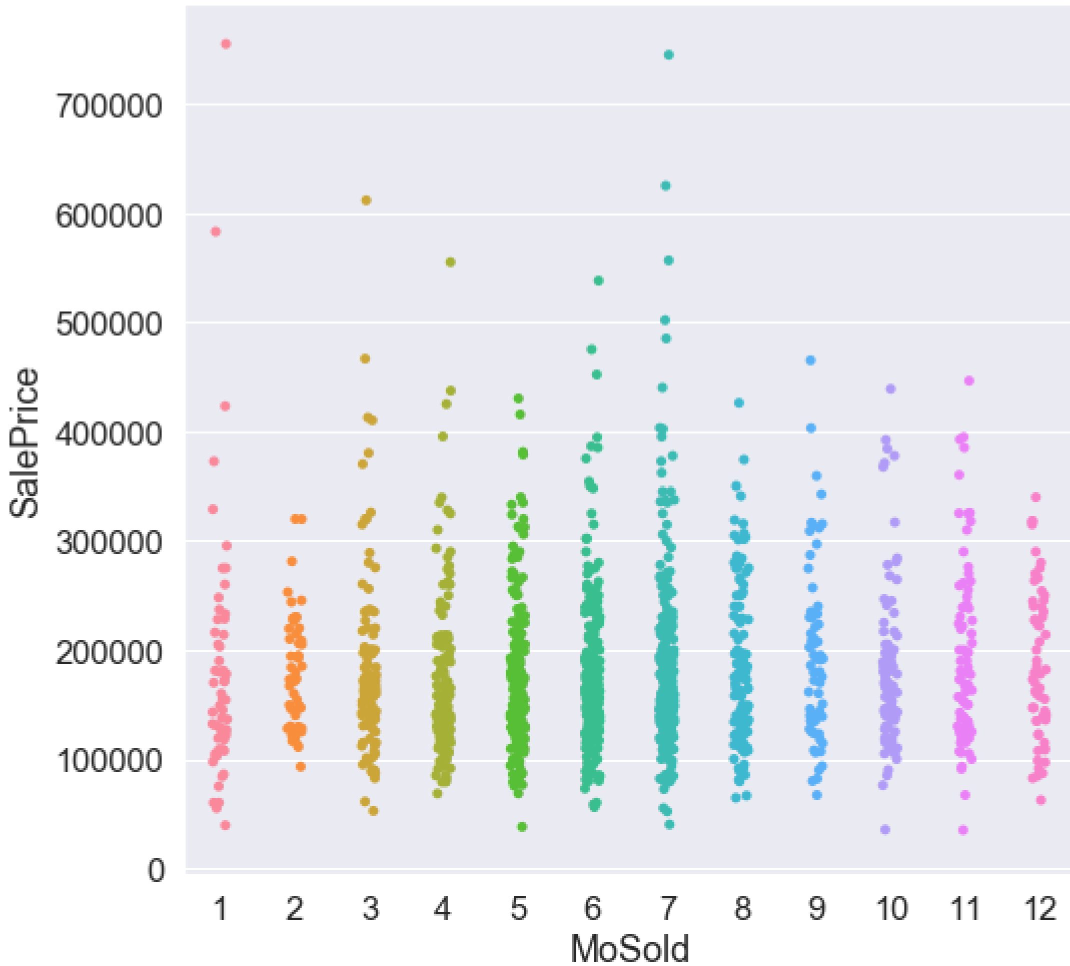
Strip Plots

In [48]:
```
f, ax = plt.subplots(figsize=(8, 8))
sns.stripplot(data = train, x='MSSubClass', y='SalePrice', jitter=.1)
plt.show()
```



In [49]:
```
f, ax = plt.subplots(figsize=(8, 8))
sns.stripplot(data = train, x='YrSold', y='SalePrice', jitter=.1)
plt.show()
```

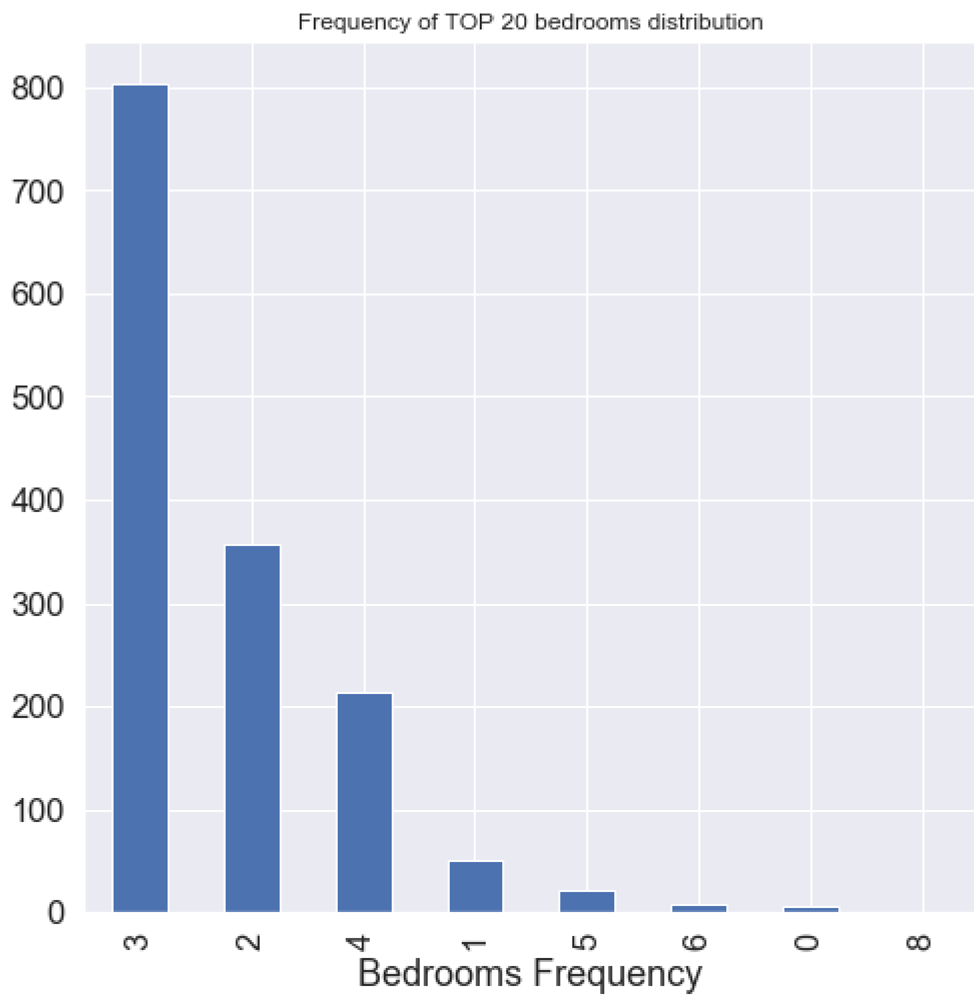
In [50]:
```
f, ax = plt.subplots(figsize=(8, 8))
sns.stripplot(data = train, x='MoSold', y='SalePrice', jitter=.1)
plt.show()
```



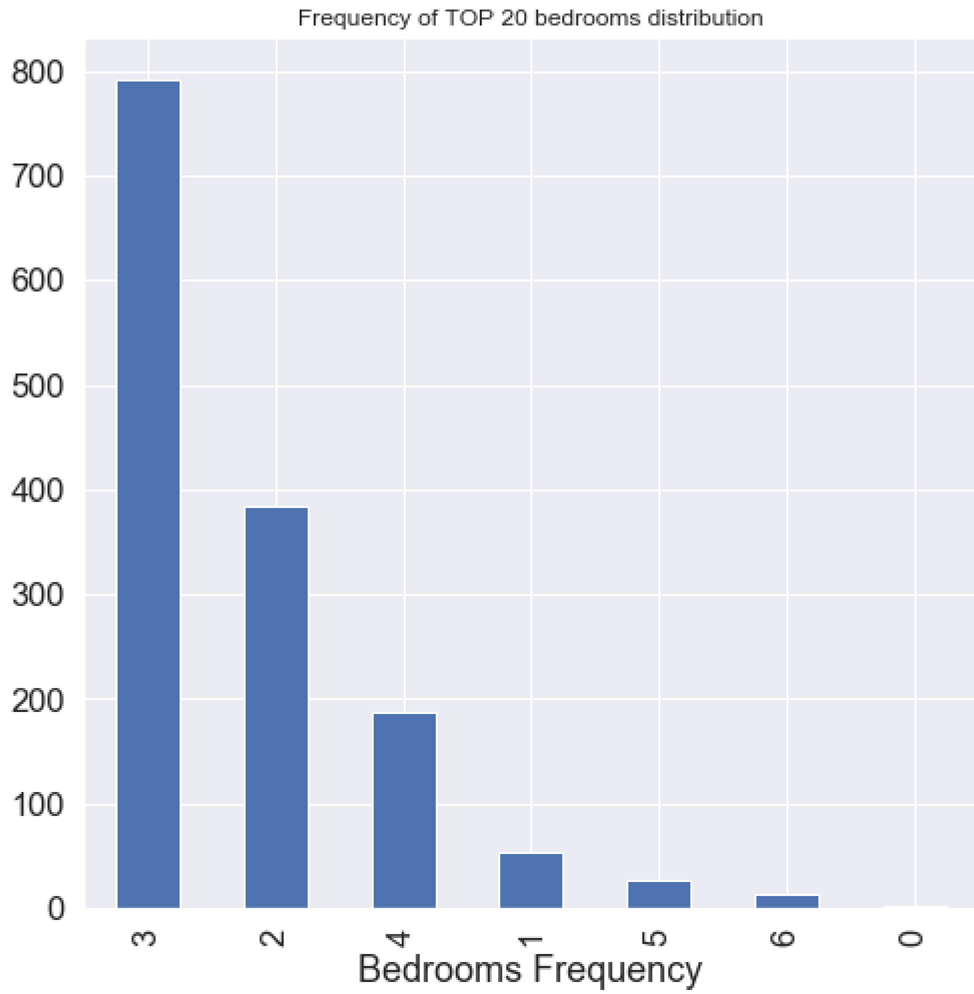
BarPlots

For largest values

In [51]:
```
plt.figure(figsize=(8,8))
train.BedroomAbvGr.value_counts().nlargest(20).plot(kind='bar')
plt.xlabel('Bedrooms Frequency')
plt.title("Frequency of TOP 20 bedrooms distribution",fontsize=12)
plt.show()
```

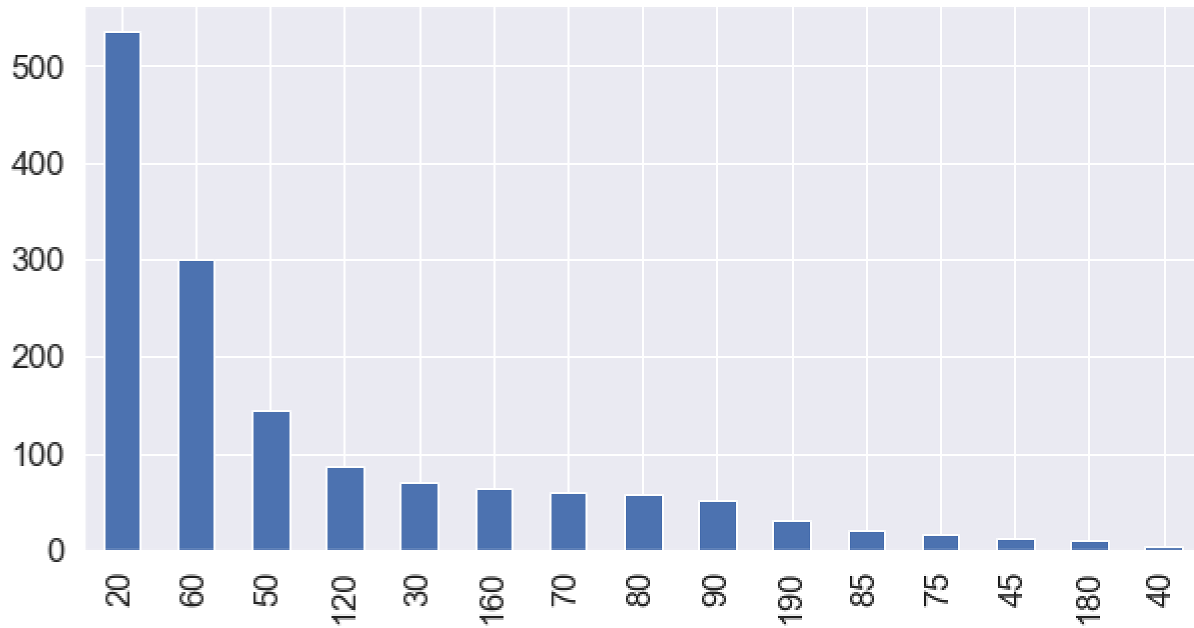


In [52]:
```
plt.figure(figsize=(8,8))
test.BedroomAbvGr.value_counts().nlargest(20).plot(kind='bar')
plt.xlabel('Bedrooms Frequency')
plt.title("Frequency of TOP 20 bedrooms distribution",fontsize=12)
plt.show()
```



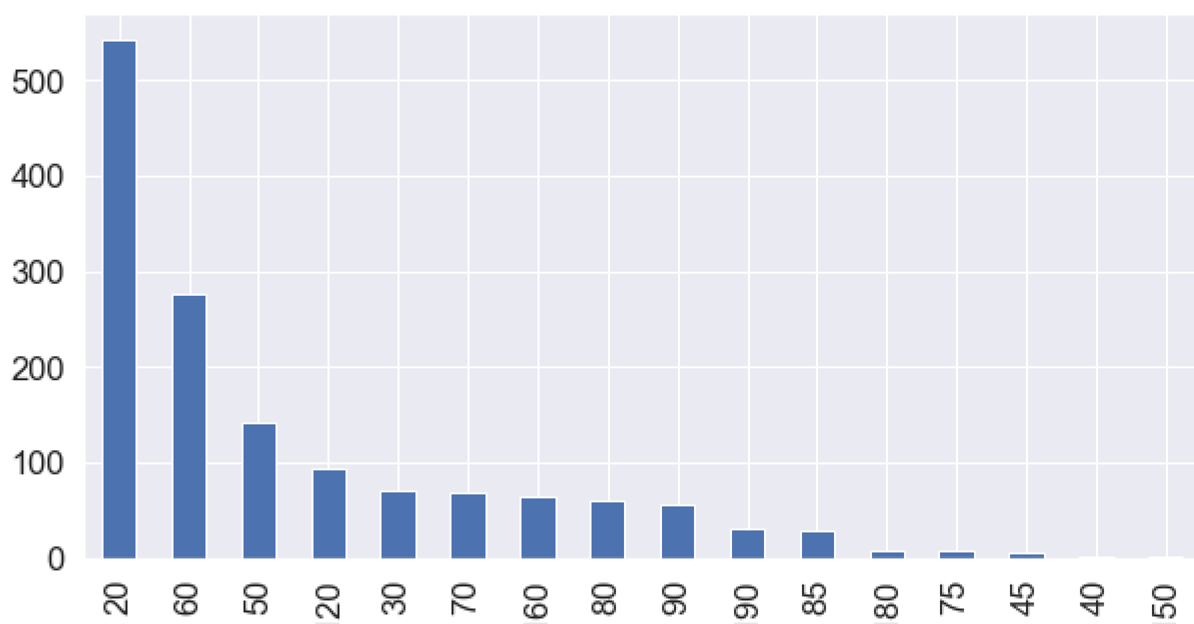
In [53]:
```
train.MSSubClass.value_counts().nlargest(20).plot(kind='bar',figsize=(10,5))
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x33316a97c8>



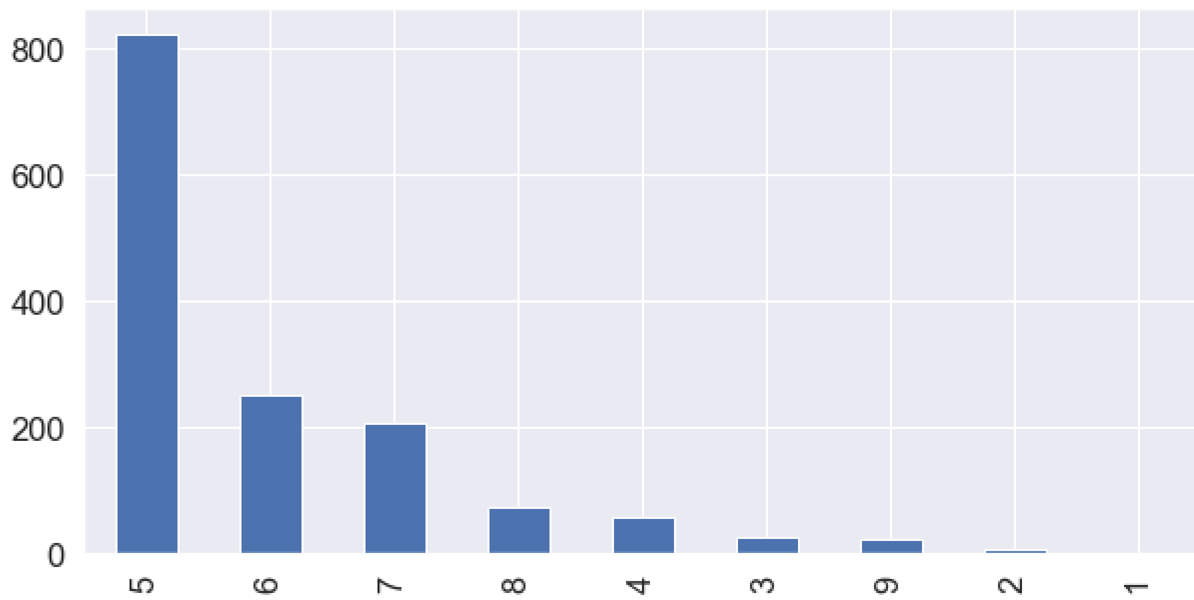
In [54]:
```
test.MSSubClass.value_counts().nlargest(20).plot(kind='bar',figsize=(10,5))
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x3333218288>



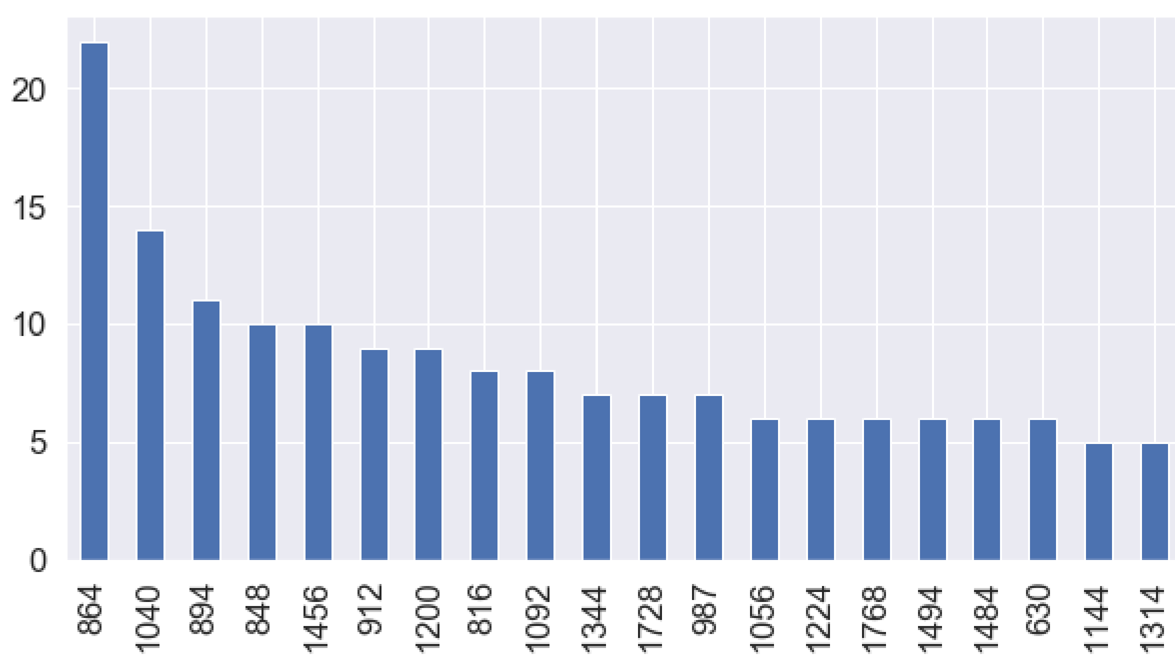
In [55]:
```
train.OverallCond.value_counts().nlargest(20).plot(kind='bar',figsize=(10,5))
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x333147c748>



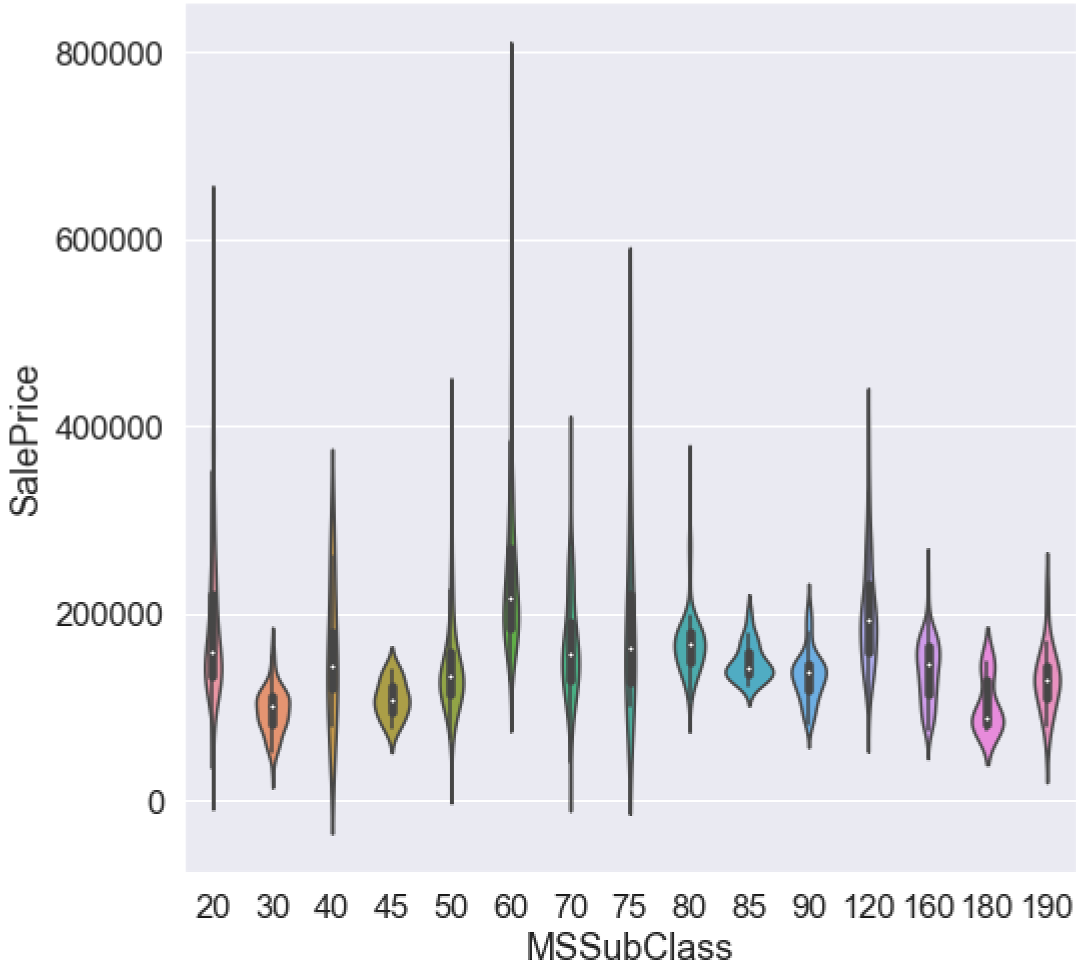
In [56]:
```
train. GrLivArea.value_counts().nlargest(20).plot(kind='bar',figsize=(10,5))
```

Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x3331600608>

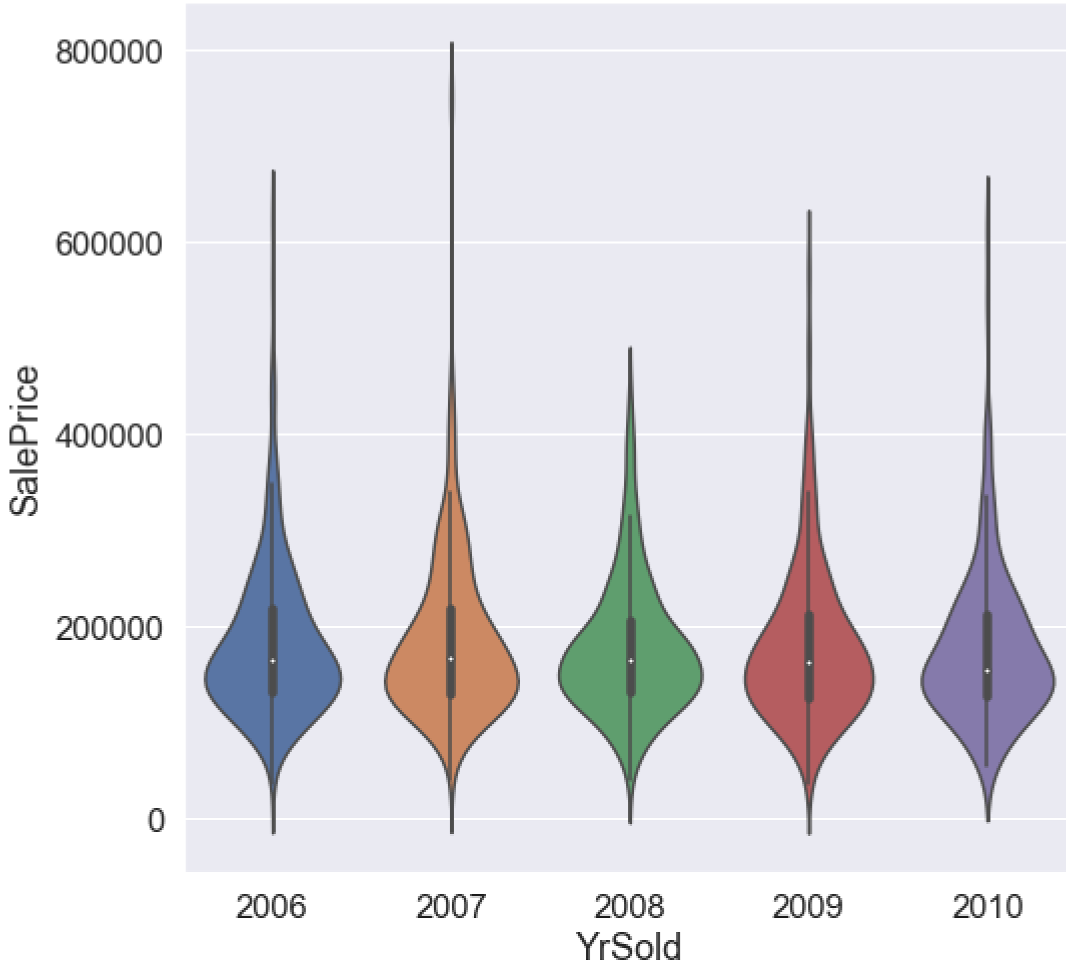


ViolinPlots

In [57]:
```
f, ax = plt.subplots(figsize=(8, 8))
sns.violinplot(data = train, x='MSSubClass', y='SalePrice')
plt.show()
```

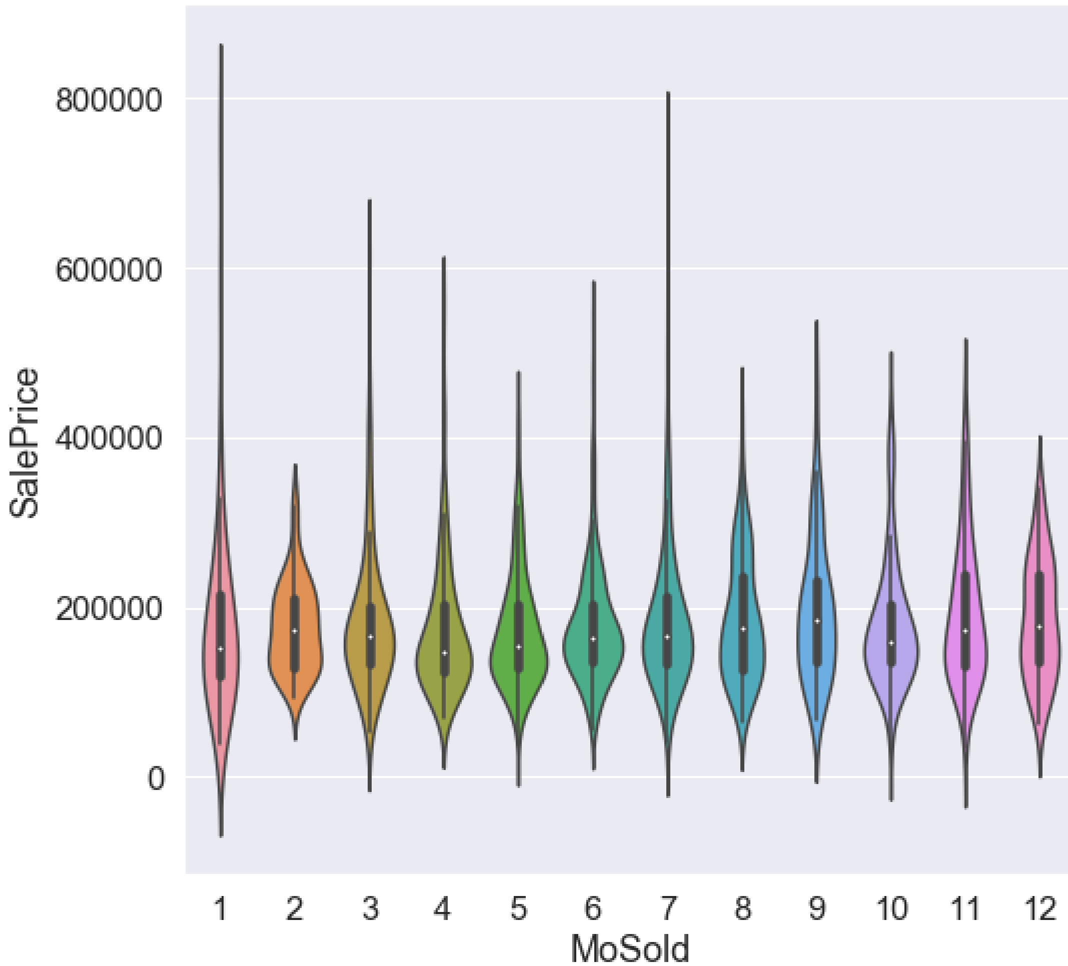


In [58]:
```
f, ax = plt.subplots(figsize=(8, 8))
sns.violinplot(data = train, x='YrSold', y='SalePrice')
plt.show()
```



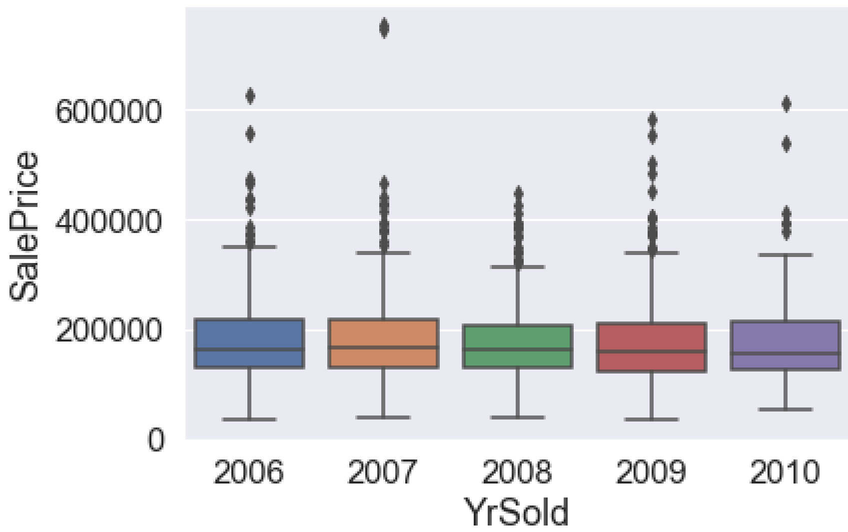
In [59]:

```
f, ax = plt.subplots(figsize=(8, 8))
sns.violinplot(data = train, x='MoSold', y='SalePrice')
plt.show()
```



In [60]:
```
sns.boxplot(train.YrSold,train.SalePrice)
```

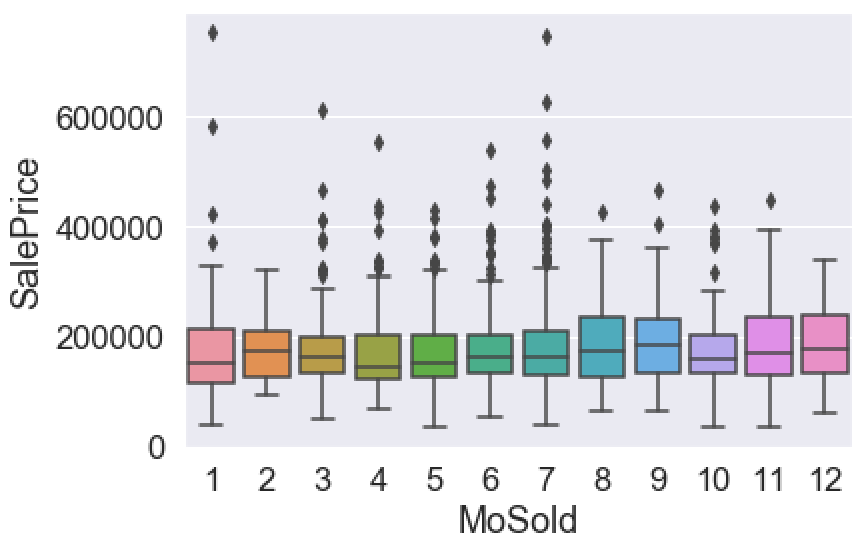
Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x33333aa948>



BoxPlots

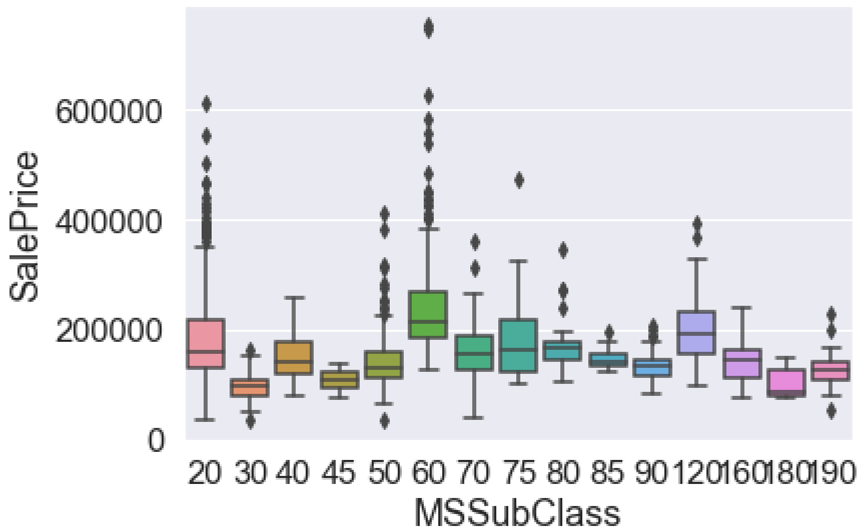
In [61]:
```
sns.boxplot(train.MoSold,train.SalePrice)
```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x33335176c8>



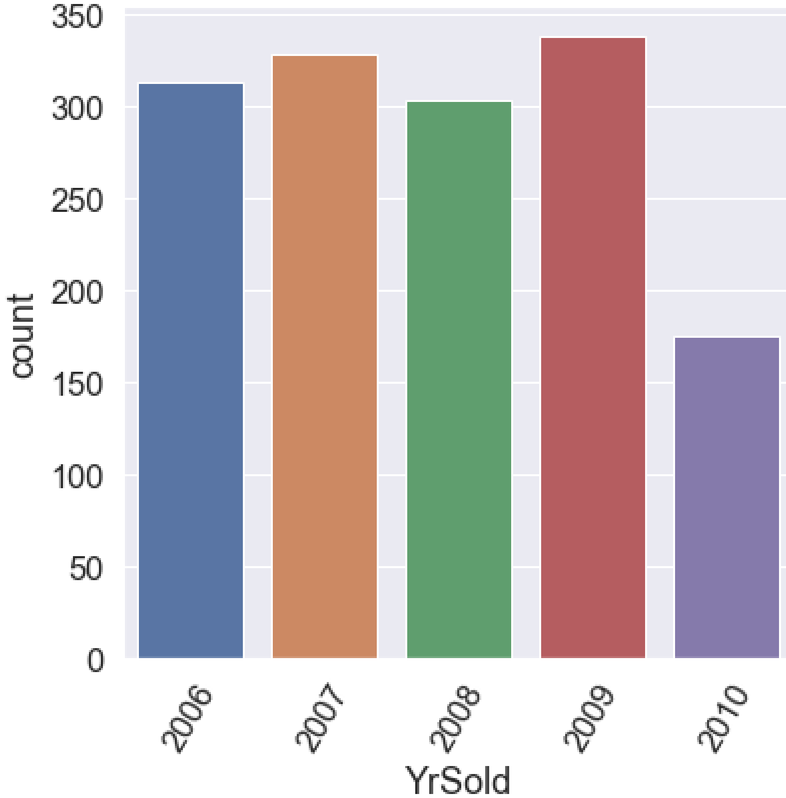
In [62]:
```
sns.boxplot(train.MSSubClass,train.SalePrice)
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x33337aecc8>

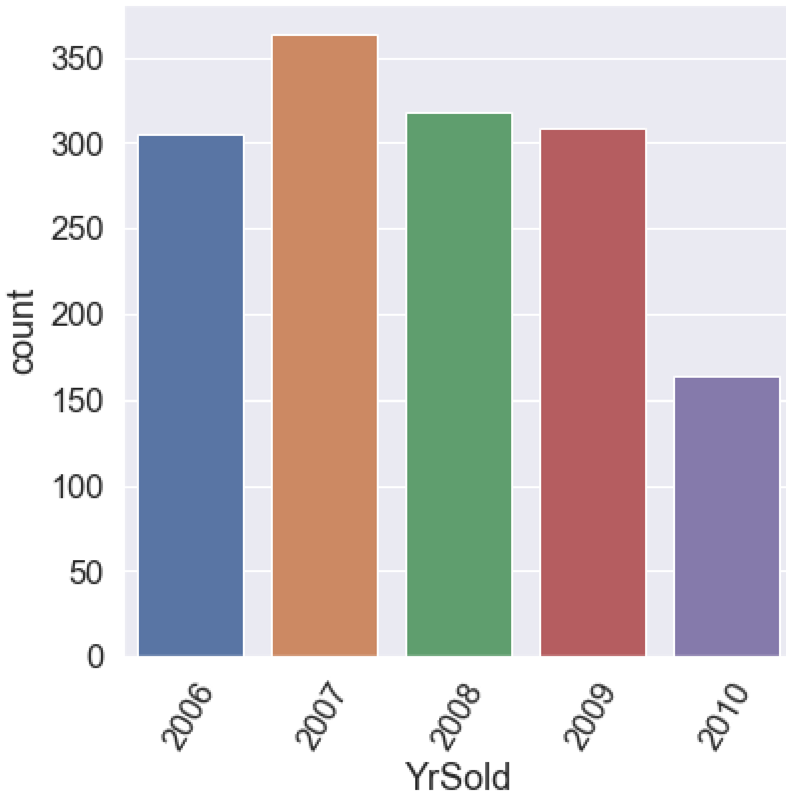


CountPlots

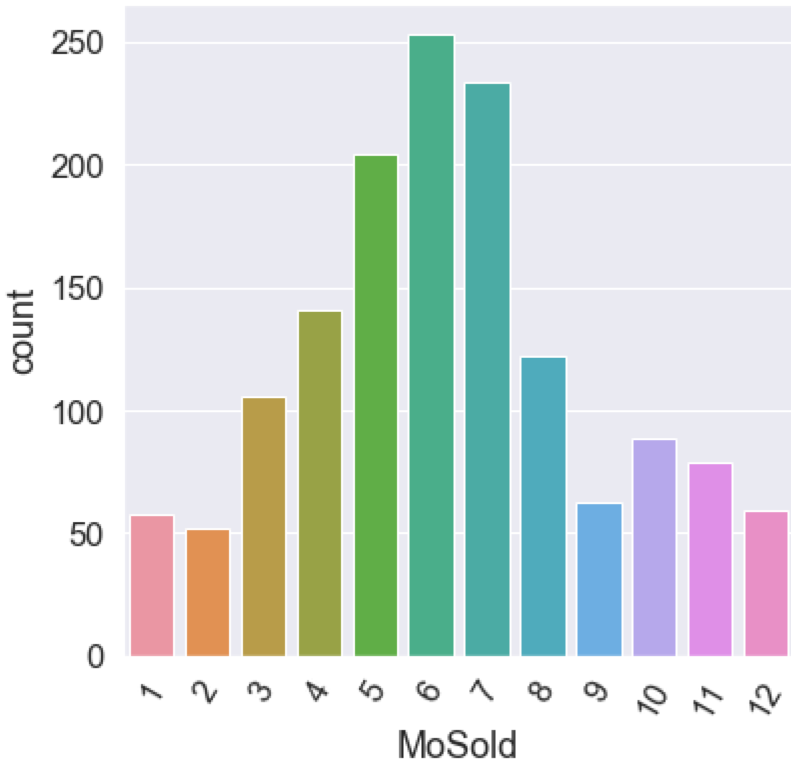
In [63]:
```
plt.figure(figsize=(6,6))
sns.countplot(x=train['YrSold'],data=train)
plt.xticks(rotation=60)
plt.show()
```



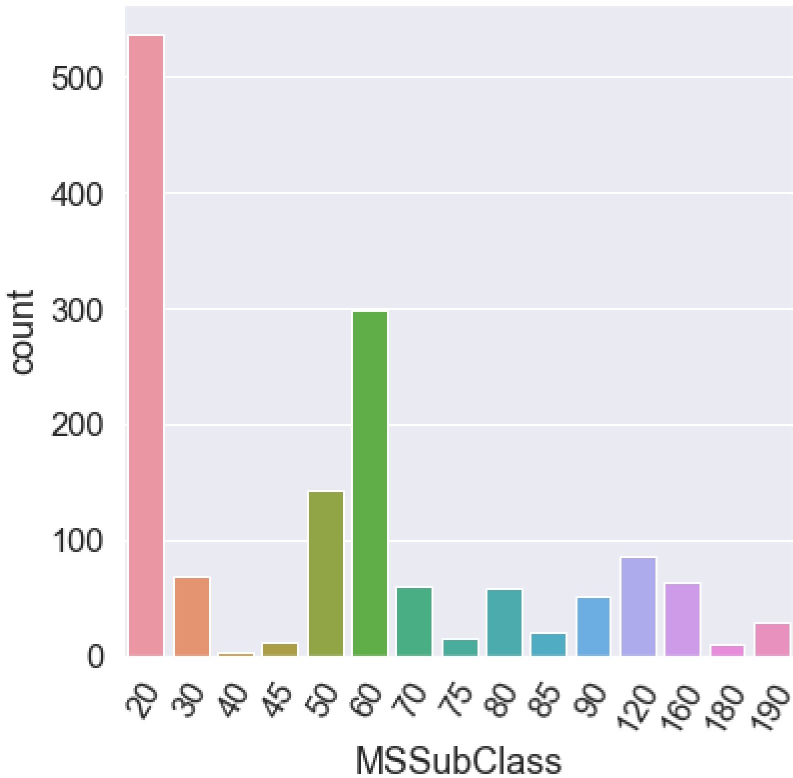
In [64]:
```
plt.figure(figsize=(6,6))
sns.countplot(x=test['YrSold'],data=test)
plt.xticks(rotation=60)
plt.show()
```



In [65]:
```
plt.figure(figsize=(6,6))
sns.countplot(x=train['MoSold'],data=train)
plt.xticks(rotation=60)
plt.show()
```



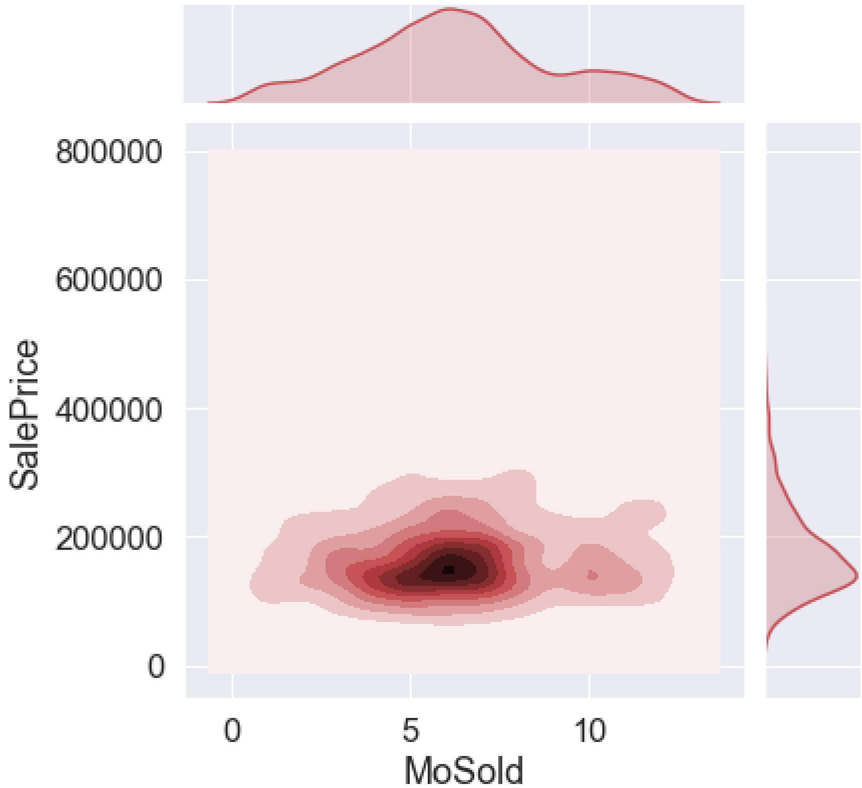
In [66]:
```
plt.figure(figsize=(6,6))
sns.countplot(x=train['MSSubClass'],data=train)
plt.xticks(rotation=60)
plt.show()
```



Joint Plots

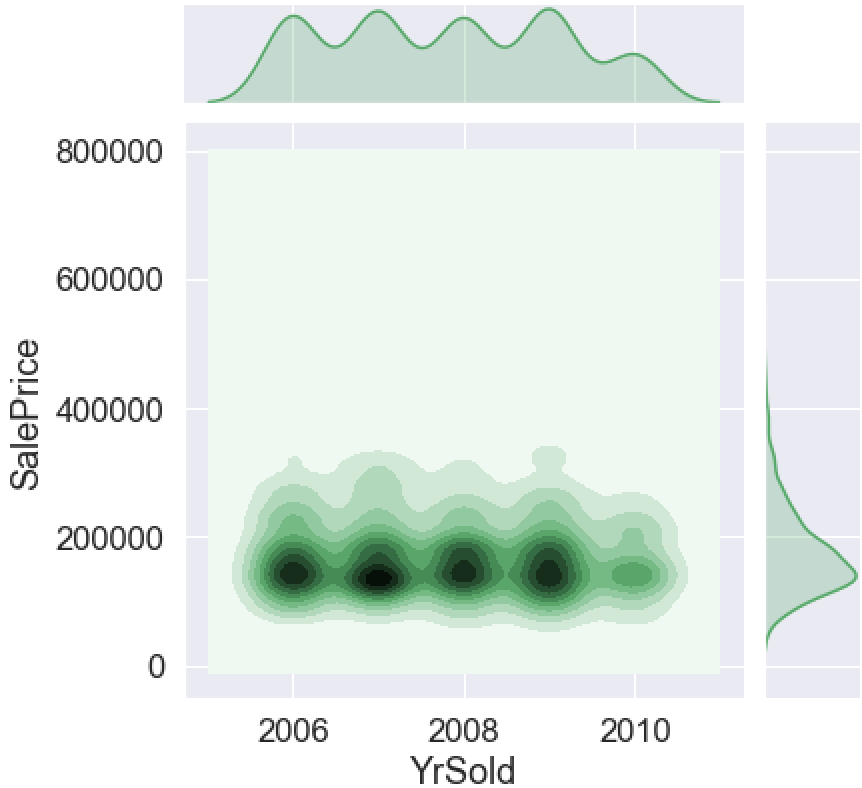
In [67]:
```
sns.jointplot(x='MoSold',y='SalePrice',kind='kde',data=train,color='r')
```

Out[67]: <seaborn.axisgrid.JointGrid at 0x3333a88888>



In [68]:
```
sns.jointplot(x='YrSold',y='SalePrice',kind='kde',data=train,color='g')
```

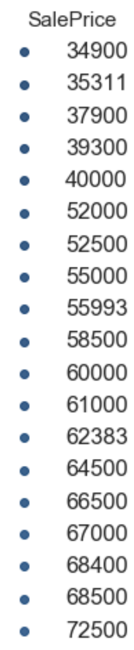
Out[68]: <seaborn.axisgrid.JointGrid at 0x33312a4588>



In [69]:
```
plt.figure(figsize=(6,6))
sns.distplot(train["SalePrice"],color="green",rug=True)
plt.axvline(train["SalePrice"].mean(),
            linestyle="dashed",color="red",
            label='mean',linewidth=2)
plt.legend(loc="best",prop={"size":14})
plt.title(" SalePrice distribution")
plt.show()
```

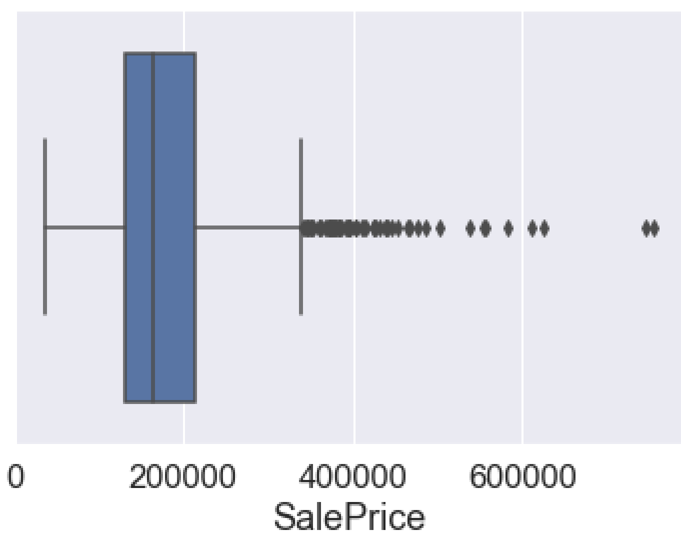
PairPlots

In [70]:
```
g = sns.PairGrid(train, vars=['MoSold', 'YrSold', 'LotArea', 'MSSubClass'],
                 hue='SalePrice', palette='RdBu_r')
g.map(plt.scatter, alpha=0.8)
g.add_legend();
```



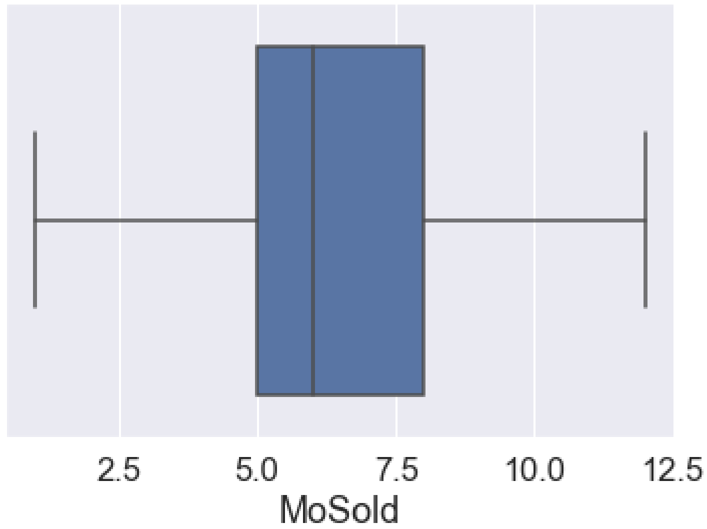
In [71]:
```
# detecting outliers
sns.boxplot(x=train['SalePrice'])
```

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x3336b0aa08>



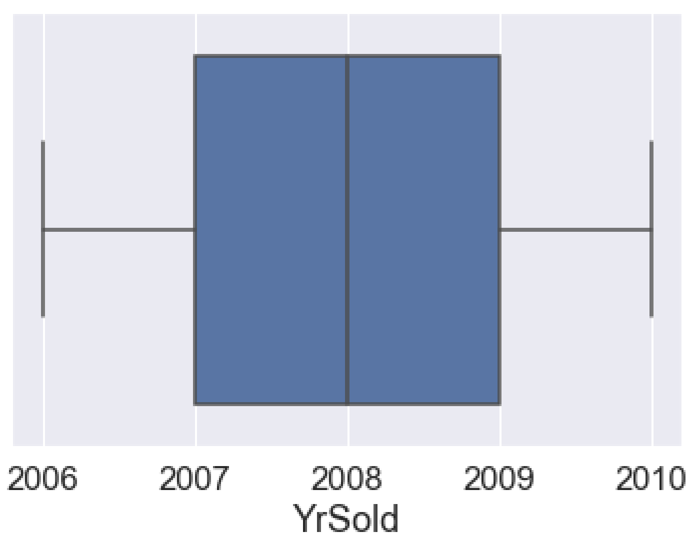
In [72]:
```
# detecting outliers
sns.boxplot(x=train['MoSold'])
```

Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x3342d03408>



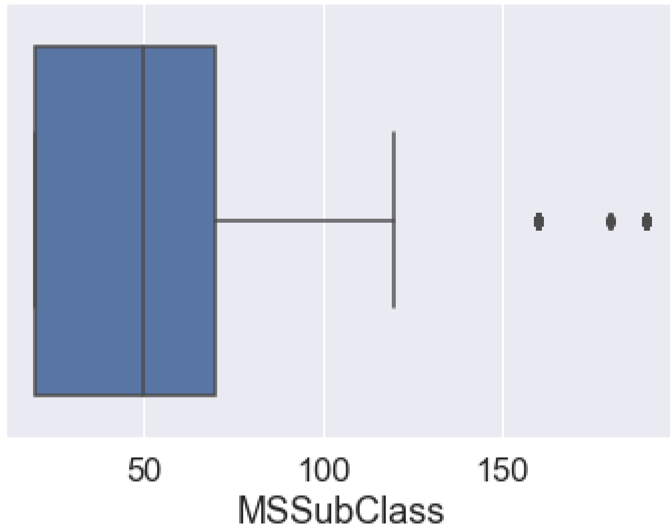
In [73]:
```
# detecting outliers
sns.boxplot(x=train['YrSold'])
```

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x3342d20b88>



In [74]:
```
# detecting outliers
sns.boxplot(x=train['MSSubClass'])
```

Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x3342cdeec8>



Preparing the data for Machine Learning

In [75]:
```
# splitting dataset in feature and target variables
feat_cols =['ExterQual', 'ExterCond','HeatingQC', 'KitchenQual', 'BsmtFinType1',
        'BsmtFinType2', 'Functional', 'BsmtExposure', 'GarageFinish', 'LandSlope'
        'BsmtQual', 'BsmtCond', 'GarageQual', 'GarageCond',
        'LotShape', 'PavedDrive', 'Street', 'Alley', 'CentralAir', 'MSSubClass',
        'YrSold', 'MoSold', 'MSZoning', 'LandContour', 'LotConfig', 'Neighborhoo
        'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofM
        'SaleType', 'SaleCondition', 'Electrical', 'Heating',
           'Exterior2nd', 'MasVnrType', 'Foundation', 'GarageType' ]
```

Encoding of the Features

In [76]:
```
from sklearn.preprocessing import LabelEncoder
#encoding the features in the train dataset
for x in feat_cols:
    lbl = LabelEncoder()
    lbl.fit(list(train[x].values))
    train[x] = lbl.transform(list(train[x].values))
```

In [77]:
```
#encoding data in the test dataset
for x in feat_cols:
    lbl = LabelEncoder()
    lbl.fit(list(test[x].values))
    test[x] = lbl.transform(list(test[x].values))
```

The feature variables and target variable

In [78]:
```
# the target variable and feature varibales
X=train[feat_cols].values #features
y =train.SalePrice.values # target variable
```

Splitting the dataset for training

In [79]:
```
# splitting t#he dataset
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3, random_state=1)
```

Shape of the data after spliting

In [80]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1022, 42)
(438, 42)
(1022,)
(438,)
```

In [81]:
```
# scaling data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Linear Regression

In [105]:
```
#using the linear regression algorithm
clf = LinearRegression(normalize=True)
#fitting the model
clf.fit(X_train,y_train)

#making the prediction
y_pred = clf.predict(X_test)

print(r2_score(y_test,y_pred))
```

```
0.6339769261151975
```

The prediction score is 63.38% based on linear regression Model

Decision Tree Classifier

In [83]:
```
#now the decision tree model
clf = DecisionTreeClassifier()
#then train the decison Tree classifier
clf = clf.fit(X_train,y_train)
#predict the respnse
y_pred =clf.predict(X_test)

# we can then test model accuracy
print('Accuracy:',metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.00684931506849315
```

Support Vector Machine

In [84]:
```
svm = SVC(kernel='rbf', gamma=0.2, C=2.0)

params = {"C":(0.3, 0.4, 1, 3, 6, 10, 11),
          "gamma":(0.01, 0.05, 0.1, 0.2, 0.5, 0.7, 1),
          "kernel":('linear', 'poly', 'rbf')}

svm_grid = GridSearchCV(svm, params, n_jobs=-1, cv=5, verbose=1, scoring="accurac
```

In [85]:
```
spv =SVC(kernel='rbf',random_state=200)
spv.fit(X,y)
spv.score(X_test,y_test)
```

Out[85]:
```
0.01141552511415525
```

Applying Standard Scaler

In [86]:
```
scaler=StandardScaler()

X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
spv =SVC(kernel='rbf',random_state=113)
spv.fit(X_test,y_test)
```

Out[86]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=113, shrinking=True, tol=0.00
1,
    verbose=False)
```

In [87]:
```
spv.score(X_test,y_test)
```

Out[87]:
```
0.24885844748858446
```

Random Forest Clsssifier

In [88]:
```
#classifier
rf =RandomForestClassifier(n_estimators=100,oob_score=True,random_state=1)
rf.fit(X_train,y_train)
pred=rf.predict(X_test)
accuracy=accuracy_score(y_test,pred)
print(accuracy)
```

```
0.0091324200913242
```

KNeighborsClassifier

In [89]:
```
#KNeighborsClassifier
kn =KNeighborsClassifier(n_neighbors=4,metric='euclidean')
kn.fit(X_train,y_train)

y_pred =kn.predict(X_test)
accuracy=accuracy_score(y_test,pred)
print(accuracy)
```

```
0.0091324200913242
```

GradientBoostingRegressor

In [103]:
```
from sklearn.ensemble import GradientBoostingRegressor
GBR = GradientBoostingRegressor(n_estimators=200, max_depth=4)
```

In [104]:
```
GBR.fit(X_train, y_train)
print("Accuracy ", GBR.score(X_test, y_test))
```

```
Accuracy  0.7625844377524957
```

Logistic Regression

In [96]:
```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.0045662100456621
```

In [97]:
```
from sklearn.model_selection import GridSearchCV

#spaced vectors
params = {"C": np.logspace(-4, 4, 20),
          "solver": ["liblinear"]}



grid_search_cv = GridSearchCV(logreg, params, scoring="accuracy", n_jobs=-1, verb
```

In [ ]:

In [ ]:

In [ ]: