Exploratory Data Analysis

Agenda

In this session, we will cover the following concepts with the help of a business use case:

- Discover patterns in the dataset
- Spot anomalies in the dataset
- Frame hypothesis from the dataset
- Gain insights by plotting different kinds of graphs using Python libraries, Seaborn, and Matplotlib

Now let's see the above concepts with help of a use case

Problem Statement:

There are so many variables that impact the price of a house. With dynamic parameters in the residential real state business, it is always important to reach a reasonable price for better business opportunities. As a part of the analytics team in a real state company, you have to come up with the variables that are impacting the price of the house through analyzing and visualizing the data.

Dataset

Download the dataset "housing_data.csv" from Course Resources section and upload the dataset in the lab.

Data Dictionary

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 building
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 dwelling
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 average or abo
ve)
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 Masonry Firepl
ace 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, typically c
ondo with a garage unit
   Family    Sale between family members
   Partial    Home was not completed when last assessed (associated with New Hom
es)
```

Initial Exploration

Learning the basics of exploratory data analysis using Python with Numpy, Matplotlib, and Pandas.

What is EDA?

In Python, EDA is used for data visualization to generate meaningful patterns and insights from data. It is also used for cleaning the data before analyzing it to remove any abnormalities.

Based on the results of EDA, companies also make business decisions, such as:

1- If EDA isn't done correctly, it can affect the further steps used in model building.

2- If EDA is done properly, it can improve the efficiency of any model.

Import necessary packages

In Python, Numpy is a package that includes multidimensional array objects as well as a number of derived objects.

Matplotlib is an amazing visualization library in Python for 2D plots of arrays.

Pandas are used for data manipulation and analysis. So these are the core libraries that are used for the EDA process.

These libraries are written with an import keyword.

In [ ]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
```

In [ ]:
```
1 %matplotlib inline
```

Let's read the data

Observations:

- We have to upload the dataset in the file explorer on the left panel of your lab
- We are reading the file through the dataframe variable
- The file is in CSV format
- We use the **`pd.read_csv()`** function to read a CSV file
- We provide the exact path of the file within the round bracket **`()`**

In [ ]:
```
1 dataframe = pd.read_csv('housing_data.csv', index_col=0)
```

pd.read_csv function is used to read the "housing_data.csv" file. If index_col=0, the function will take the first column as an index by default, whereas if index_col=None, then the first row is not used as the column name.

dataframe is a variable which will store the data read by the csv file.

Print first 5 lines of the dataset

In [ ]:
```
1 dataframe.head(5)
```

Out[13]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig
0	SC60	RL	65.0	8450	Pave	None	Reg	Lvl	AllPub	Inside
1	SC20	RL	80.0	9600	Pave	None	Reg	Lvl	AllPub	FR2
2	SC60	RL	68.0	11250	Pave	None	IR1	Lvl	AllPub	Inside
3	SC70	RL	60.0	9550	Pave	None	IR1	Lvl	AllPub	Corner
4	SC60	RL	84.0	14260	Pave	None	IR1	Lvl	AllPub	FR2

5 rows × 80 columns

head(5) function prints the first five rows from the dataframe.

Print last 5 lines of the dataset

tail(n=5) prints the last five rows from the dataframe.

In [ ]:
```
1 dataframe.tail(5)
```

Out[14]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConf
1455	SC60	RL	62.0	7917	Pave	None	Reg	Lvl	AllPub	Insi
1456	SC20	RL	85.0	13175	Pave	None	Reg	Lvl	AllPub	Insi
1457	SC70	RL	66.0	9042	Pave	None	Reg	Lvl	AllPub	Insi
1458	SC20	RL	68.0	9717	Pave	None	Reg	Lvl	AllPub	Insi
1459	SC20	RL	75.0	9937	Pave	None	Reg	Lvl	AllPub	Insi

5 rows × 80 columns

Check the size of the dataset

Shape is an attribute that returns a tuple representing the dimensionality of the dataframe. Shape is used to define the number of rows and columns in a dataframe.

In [ ]:
```
1 dataframe.shape
```

Out[15]:
```
(1460, 80)
```

It is also good practice to know the columns and their corresponding data types, along with finding whether they contain null values or not. For this, info() method is used to print the information about the dataframe.

In [ ]:
```
1 dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 0 to 1459
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     1460 non-null   object
 1   MSZoning       1460 non-null   object
 2   LotFrontage    1460 non-null   float64
 3   LotArea        1460 non-null   int64
 4   Street         1460 non-null   object
 5   Alley          1460 non-null   object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
```

Observation: ** There are 80 columns in the dataframe

The `describe()` function helps us to get various summary statistics that exclude NaN values.

This function returns the **count, mean, standard deviation, minimum, and maximum** values as well as the quantiles of the data.

In [ ]:
```
1 dataframe.describe()
```

Out[17]:

	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	Bsn
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	146
mean	57.623288	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.117123	44
std	34.664304	9981.264932	1.382997	1.112799	30.202904	20.645407	180.731373	45
min	0.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	
25%	42.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	
50%	63.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	38
75%	79.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	164.250000	71
max	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	564

8 rows × 35 columns

Let's find out the number of numerical features in our dataset.

In [ ]:
```
numerical_feature_columns = list(dataframe._get_numeric_data().columns)
numerical_feature_columns
```

Out[18]:
```
['LotFrontage',
 'LotArea',
 'OverallQual',
 'OverallCond',
 'YearBuilt',
 'YearRemodAdd',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',
```

Let's find out the number of categorical features in our dataset.

In [ ]:
```
categorical_feature_columns = list(set(dataframe.columns)-set(dataframe._get_numeric
categorical_feature_columns
```

Out[19]:
```
['Heating',
 'MSSubClass',
 'BsmtQual',
 'GarageQual',
 'Condition2',
 'FireplaceQu',
 'BsmtFinType2',
 'PavedDrive',
 'PoolQC',
 'MiscFeature',
 'ExterQual',
 'Exterior2nd',
 'HouseStyle',
 'Fence',
 'Electrical',
 'BsmtExposure',
 'RoofMatl',
 'Alley',
 'SaleCondition',
```

Now, we want to analyze the data in order to extract some insights

There are two major types to analyze the data

- Univariate analysis
- Multivariate analysis

Univariate Analysis

- Univariate analysis is the simplest form of analyzing data.
- “Uni” means “one”, so in other words, data has only one variable.
- It doesn’t deal with causes or relationships and its major purpose is to describe. It takes data, summarizes that data, and finds patterns in the data.

- Univariate Analysis can be done either on **numerical or categorical** features.

Let's explore the types of plot for univariate analysis

Histogram

- In the univariate analysis, we use histograms for analyzing and visualizing frequency distribution.
- Plotting histograms in Pandas are very easy and straightforward.
- A **histogram** represents the distribution of data by forming bins along with the range of the data and then drawing bars to show the number of observations that fall in each bin.
- The bin can be of any size.

```
 Note: When dealing with a set of data, usually the first thing is to get a sense of
how the variables are distributed.
We start by identifying a few variables of interest and checking their distribution.
```

In order to make any prediction we need to fit a linear regression model, so we have to make sure the distribution of the variables is almost linear.

Now to check the linearity of the variables for any skewness in the distribution and outliers in the data.

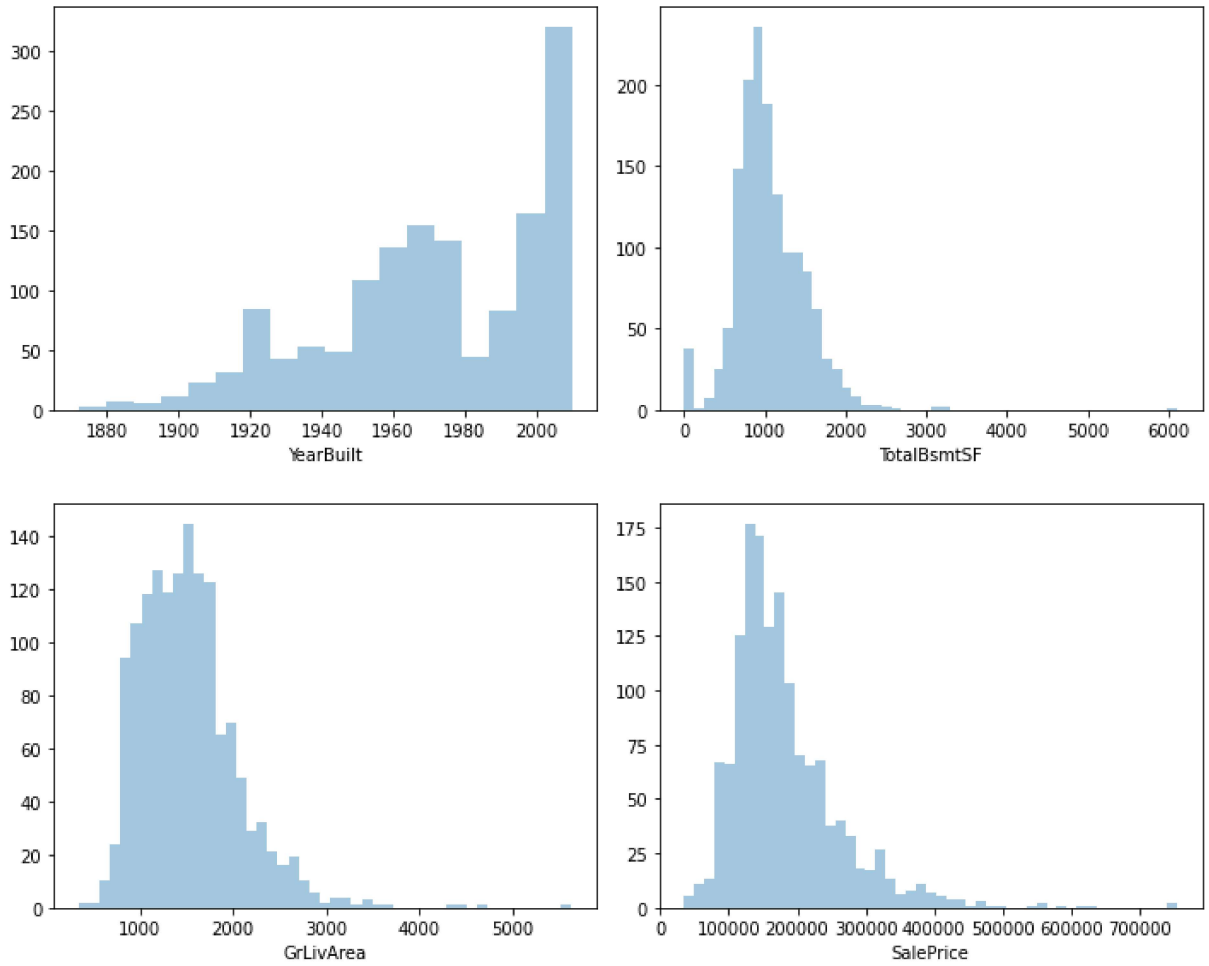
The variables we check for are 'OverallQual', 'YearBuilt', 'TotalBsmtSF', 'GrLivArea', 'Neighborhood' and the target variable 'SalePrice'.

Plot Histogram for 'YearBuilt', 'TotalBsmtSF', 'GrLivArea', and 'SalePrice'

In [ ]:

```
import seaborn as sns

num_cols = ['YearBuilt', 'TotalBsmtSF', 'GrLivArea', 'SalePrice']

for i in range(0,len(num_cols),2):
        plt.figure(figsize=(10,4))
        plt.subplot(121)
        sns.distplot(dataframe[num_cols[i]], kde=False)
        plt.subplot(122)
        sns.distplot(dataframe[num_cols[i+1]], kde=False)
        plt.tight_layout()
        plt.show()
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `d
istplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Observations:

In the above graphs, the YearBuilt graph is left skewed and all the other graphs are right skewed. We need to normalize the data in EDA.

The Kernel Density Estimation

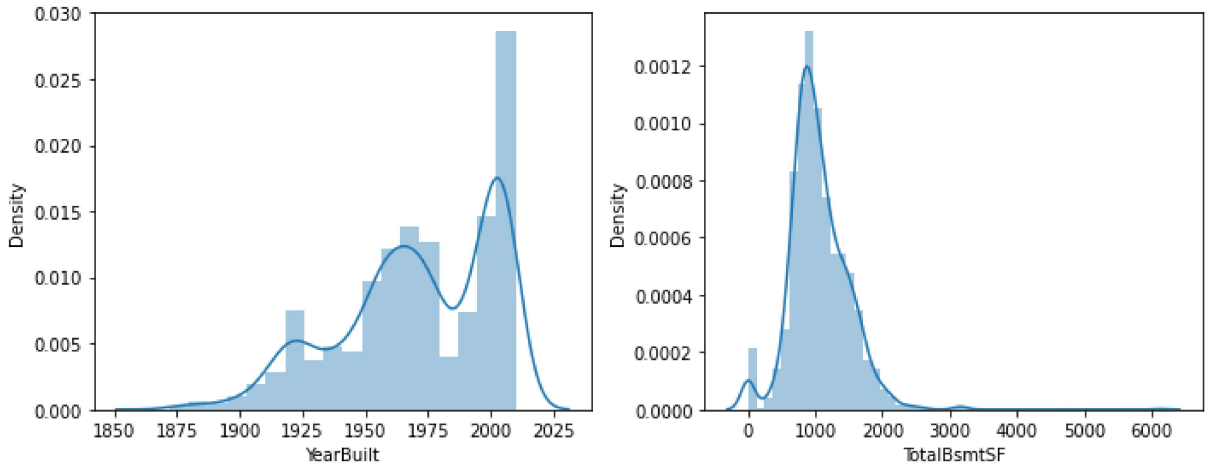
- it is also a useful tool for plotting the shape of a distribution.
- Like the histogram, the KDE plot encodes the density of observations on one axis with height along the other axis.

Plot Kernel Density Function for 'YearBuilt', 'TotalBsmtSF', 'GrLivArea', and 'SalePrice'

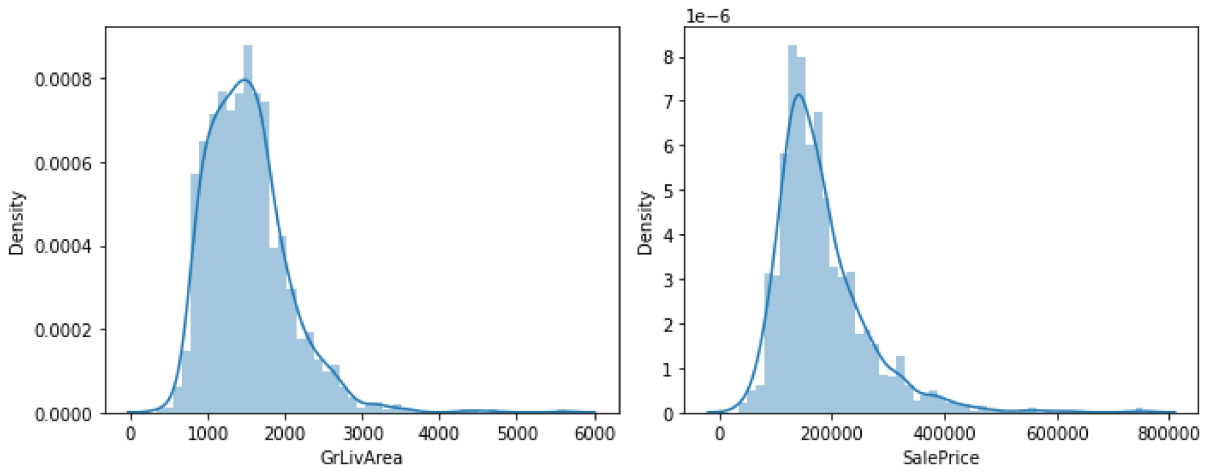
In [ ]:

```
num_cols = ['YearBuilt', 'TotalBsmtSF', 'GrLivArea', 'SalePrice']

for i in range(0,len(num_cols),2):
        plt.figure(figsize=(10,4))
        plt.subplot(121)
        sns.distplot(dataframe[num_cols[i]], hist=True, kde=True)
        plt.subplot(122)
        sns.distplot(dataframe[num_cols[i+1]], hist=True, kde=True)
        plt.tight_layout()
        plt.show()
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `d
istplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `d
istplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `d
istplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or
`histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



We can observe that all of the histograms and KDE plots are left or right skewed, hence a transformation is required to make them linear.

Boxplots

- A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable.
- The box shows the quartile of the dataset while the whiskers extend to show the rest of the distribution.
- Boxplots help to find the features which can be removed.
- This plot is also used in multivariate analysis.

- The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the five-number summary:
  - Minimum
  - First quartile
  - Median
  - Third quartile
  - Maximum
- In the simplest box plot, the central rectangle spans the first quartile to the third quartile (the interquartile range or IQR).
- A segment inside the rectangle shows the median and "whiskers" above and below the box show the locations of the minimum and maximum.

Outliers are either 3×IQR or more above the third quartile or 3×IQR or more below the first quartile.

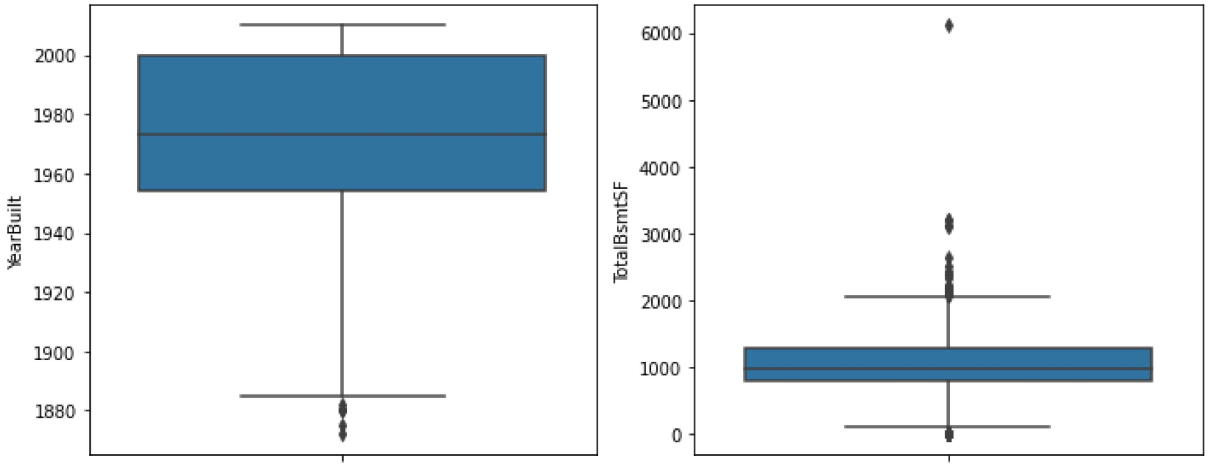
Suspected outliers are slightly more central versions of outliers, either 1.5×IQR or more above the third quartile or 1.5×IQR or more below the first quartile.

Plot Boxplots for 'YearBuilt', 'TotalBsmtSF', 'GrLivArea', and 'SalePrice'

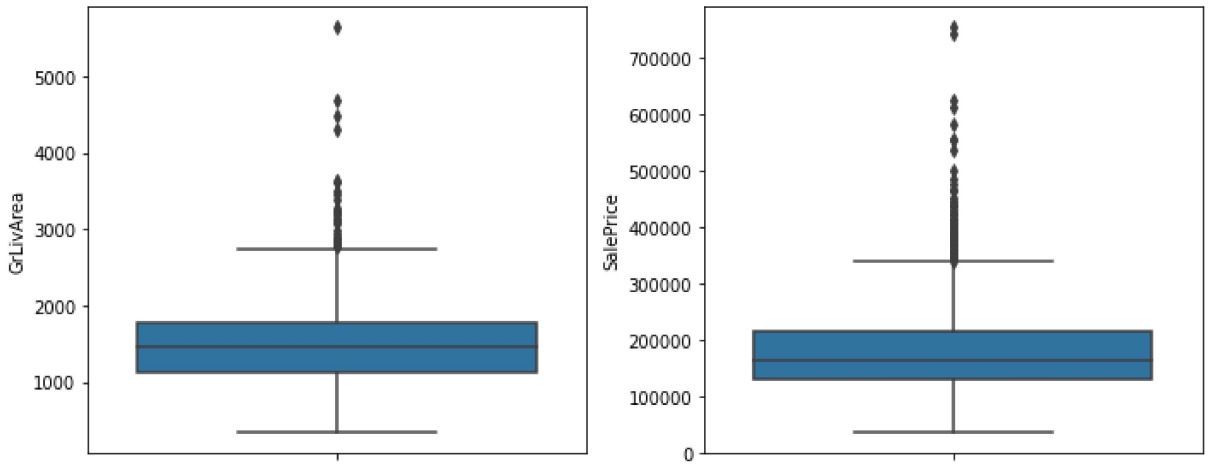
In [ ]:

```
num_cols = ['YearBuilt', 'TotalBsmtSF', 'GrLivArea', 'SalePrice']

facet = None

for i in range(0,len(num_cols),2):
        plt.figure(figsize=(10,4))
        plt.subplot(121)
        sns.boxplot(facet, num_cols[i],data = dataframe)
        plt.subplot(122)
        sns.boxplot(facet, num_cols[i+1],data = dataframe)
        plt.tight_layout()
        plt.show()
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
```




```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
```



Observation:

Plot Boxplots for 'YearBuilt', 'TotalBsmtSF', 'GrLivArea', and 'SalePrice' show some outliers, which are represented by dots.

Univariate Analysis for Categorical Variable

Most of the buildings are built post 1960 and most houses have an area in the range 1-2k sq feet. There are quite a few outliers in the sale price and living area and these might be correlated. We need to check the distribution of the categorical columns.

We chose the following columns from the above categorical_features_columns :

- Neighborhood
- SaleCondition

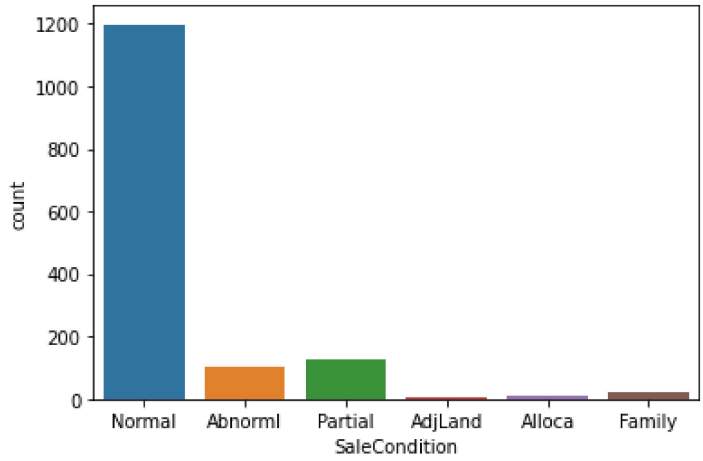
Countplot for Categorical Feature 'SaleCondition'

- This plots histograms of the variable

In [ ]:
```
1 sns.countplot('SaleCondition', data=dataframe)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variable as a keyword arg: x. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will r
esult in an error or misinterpretation.
  FutureWarning
```

Out[23]: <AxesSubplot:xlabel='SaleCondition', ylabel='count'>

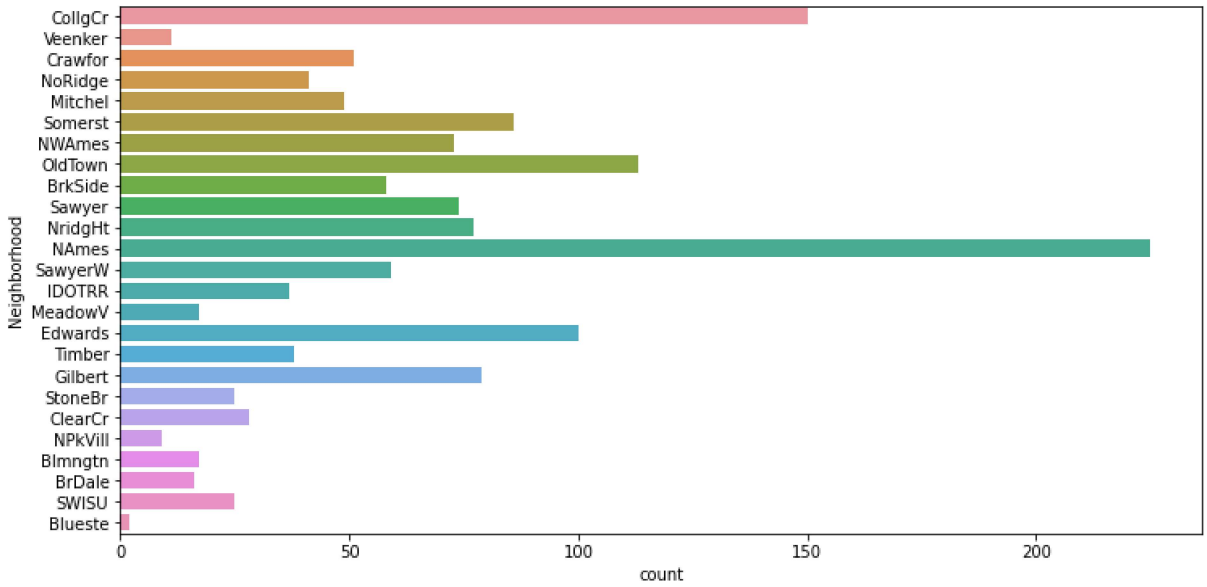


In the above graph, we can see the count of observations in each categorical bin using bars.

Countplot For Categorical Feature 'Neighborhood'

In [ ]:
```
plt.figure(figsize=(12,6))

sns.countplot(y='Neighborhood', data=dataframe)
```

Out[24]: <AxesSubplot:xlabel='count', ylabel='Neighborhood'>



In the above graph, we can see the count of observations in each categorical bin using bars in the y-axis.

Now we will plot some important feature against our target.

Multivariate Analysis

In multivariate analysis, we try to find the relations between multiple variables. Obviously, in real-life problems, variables can be any combination of numeric or categorical variables. The combinations are:

- Numeric vs. Numeric
- Numeric vs. Categorical
- Categorical vs. Categorical

Another aspect of variable combination are:

- Feature vs. Feature
- Feature vs. Target

Numeric vs. Numeric

For these kind of plots we use a scatterplot of the two variables. Although one can use a variety of plots in seaborn to do a quick and dirty EDA to some sophisticated plots. Implot is one of the plots to do a scatterplot in seaborn, it'll by default fit a regression line on top which you can control using 'fit_reg' argument.

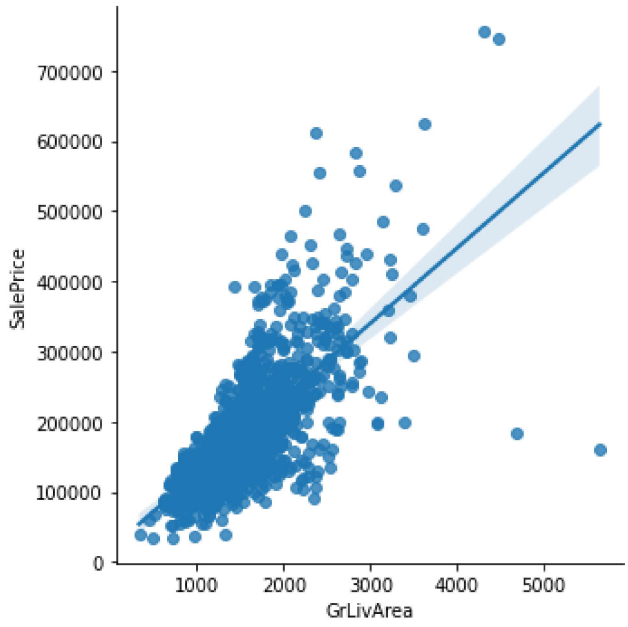
Now we will plot some important feature against our target.

Plot 'GrLivArea' against 'SalePrice'

In [ ]:
```
1 sns.lmplot('GrLivArea', 'SalePrice', data=dataframe, fit_reg=True)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
```

Out[25]:
```
<seaborn.axisgrid.FacetGrid at 0x7f985eaa3f10>
```



- Using a scatterplot we can also detect *multivariate outliers*, in this case there are two houses which have an area above ~4500 and they don't follow the trend. Removing these would give a better fit.

Another plot that we can use is a jointplot which gives a plethora of information in a single plot. It has:

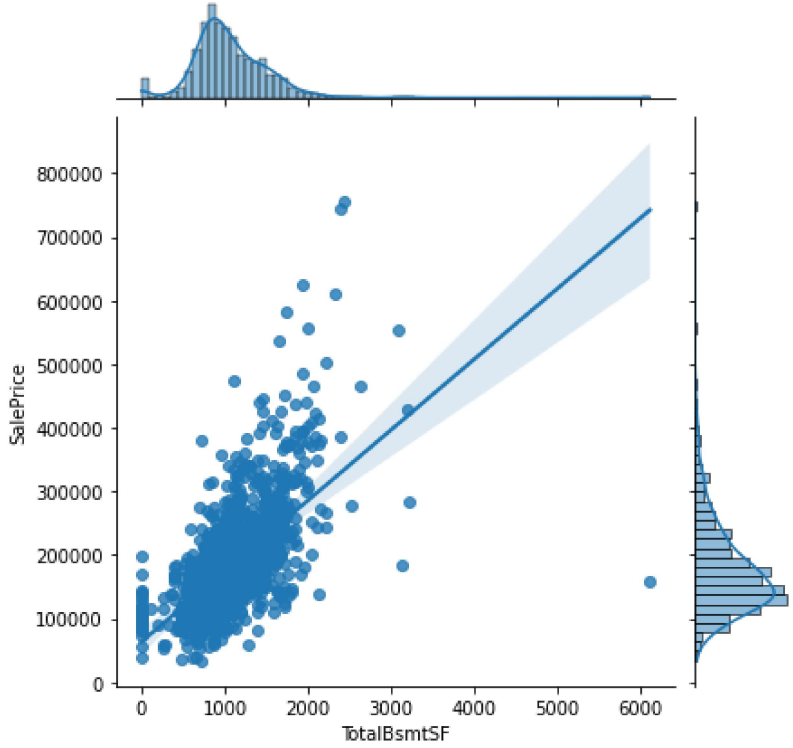
- Scatter Plot
- Regression line fit to the data.
- Histogram and KDE of individual variables.
- Pearson correlation and p value.

Plot 'TotalBsmtSF' against 'SalePrice'

In [ ]:
```
1 sns.jointplot('TotalBsmtSF', 'SalePrice', data=dataframe, kind='reg')
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
```

Out[26]:
```
<seaborn.axisgrid.JointGrid at 0x7f985ea59090>
```



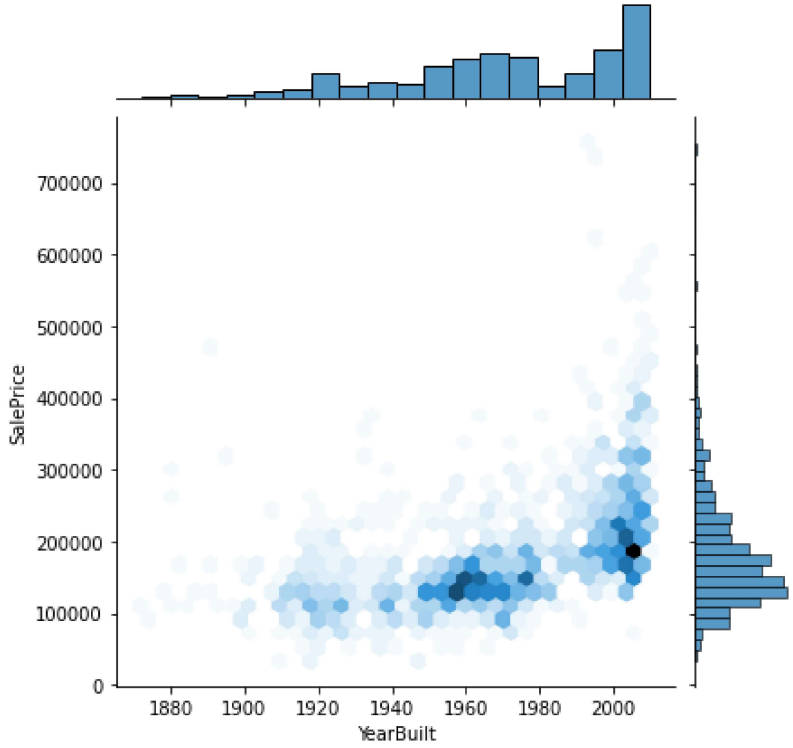
Another plot one can use is a hexplot which plots two numeric variables. Darker colors signify more points

Plot 'YearBuilt' against 'SalePrice'

In [ ]:
```
1 sns.jointplot('YearBuilt', 'SalePrice', data=dataframe, kind='hex')
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
```

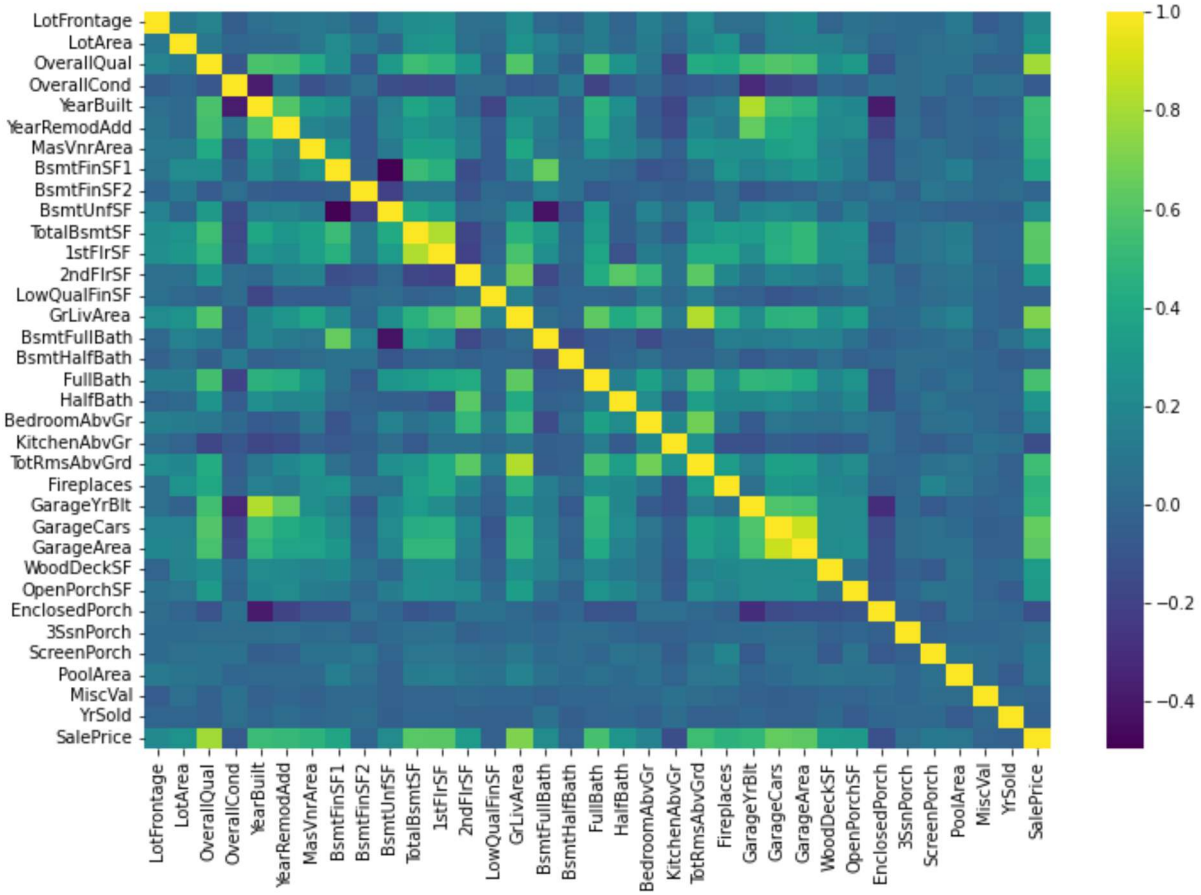
Out[27]:
```
<seaborn.axisgrid.JointGrid at 0x7f985ecbd590>
```



Its necessary to remove correlated variables to improve our model. We must find correlations can visualize the correlation matrix using a heatmap.

In [ ]:
```
plt.figure(figsize=(12,8))
sns.heatmap(dataframe.corr(), cmap='viridis')
```

Out[28]: <AxesSubplot:>



We can see the following inferences from above -

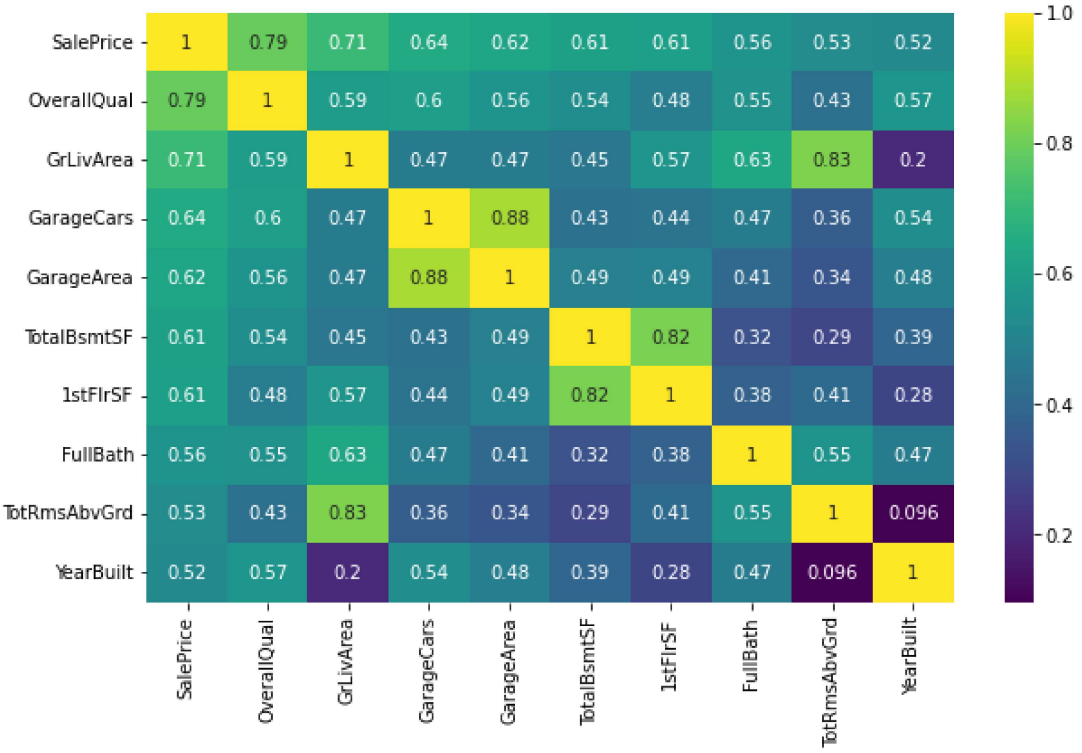
- A lot of variables are correlated to SalePrice which is good.
- GrLivArea is highly correlated with TotRmsAbvGrd.
- Also year the garage was built(GarageYrBlt) is correlated with the year the building was built(YearBuilt).

It's a good idea to remove such correlated variables during feature selection.

Now we plot some top variables to see the relations between them. We starts by filtering top 10 variables which are highly correlated with SalePrice.

In [ ]:
```
#saleprice correlation matrix
k = 10 #number of variables for heatmap
cols = dataframe.corr().nlargest(k, 'SalePrice')['SalePrice'].index
cm = dataframe[cols].corr()
plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True, cmap = 'viridis')
```

Out[29]: <AxesSubplot:>

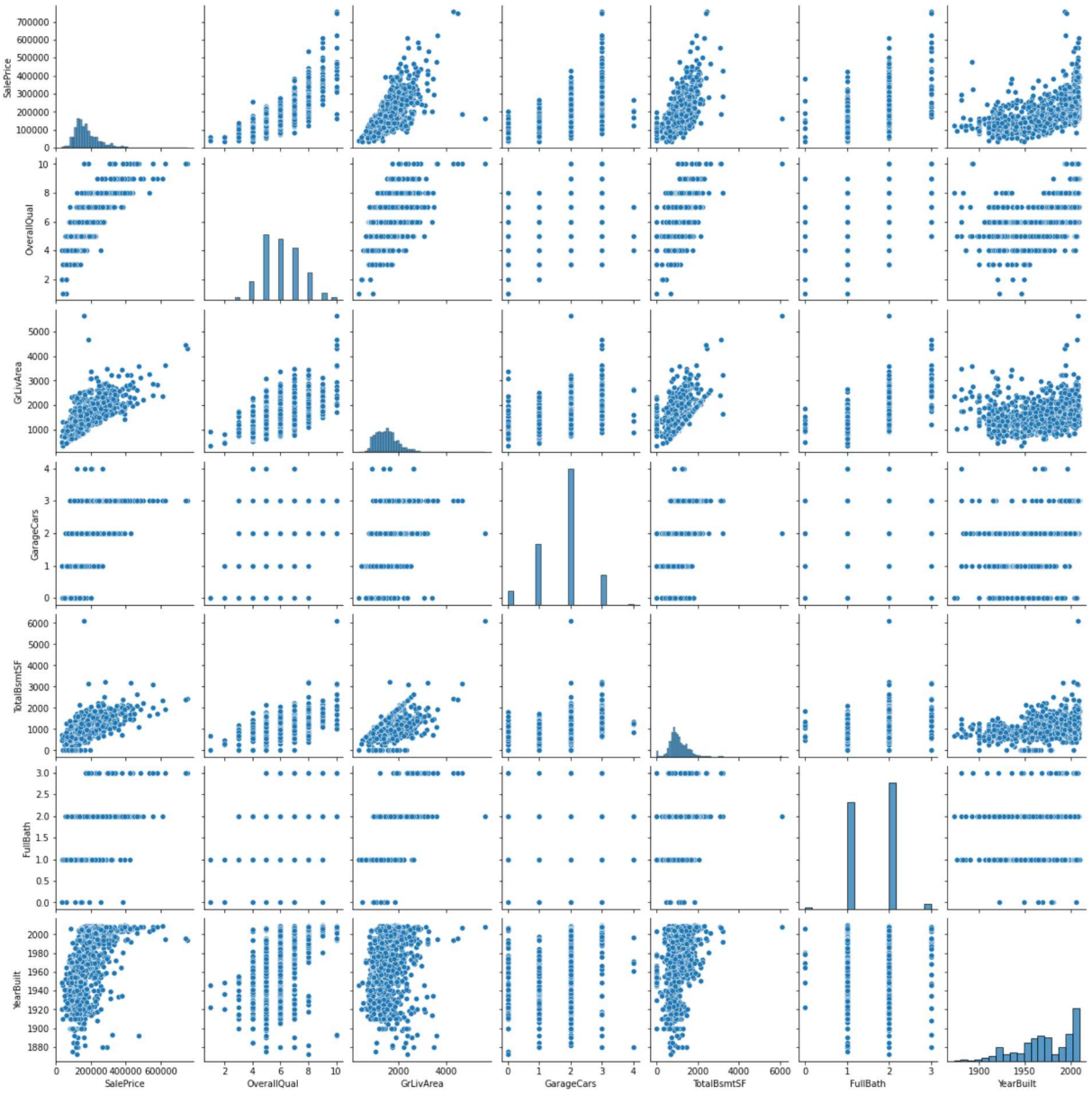


- 'OverallQual', 'GrLivArea' and 'TotalBsmtSF' are strongly correlated with 'SalePrice'.
- 'GarageCars' and 'GarageArea' are also correlated variables. However no. of cars that can fit into a garage is dependent on the garage area and one can remove one of these. Check the correlation between them.
- TotalBsmtSF and 1stFlrSF are also highy correlated. We can drop one of these.
- As pointed out above Yearbuilt and TotRmsAbvGrd are highly correlated, we'll discard TotRmsAbvGrd.

An easy way to plot all possible interactions is between a set of numeric variables is using pairplot function in seaborn.

In [ ]:
```
# Visualizing relations between all major variables
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullB
sns.pairplot(dataframe[cols])
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x7f985e752ad0>



We notice a few interesting things here, in the scatter plot of TotalBsmtSF and GrLivArea, there is a line below which most TotalBsmtSF values fall in which makes sense as the area of basement usually is lesser than living area. Also there is an exponential increase in SalePrice vs Yearbuilt in the recent past.

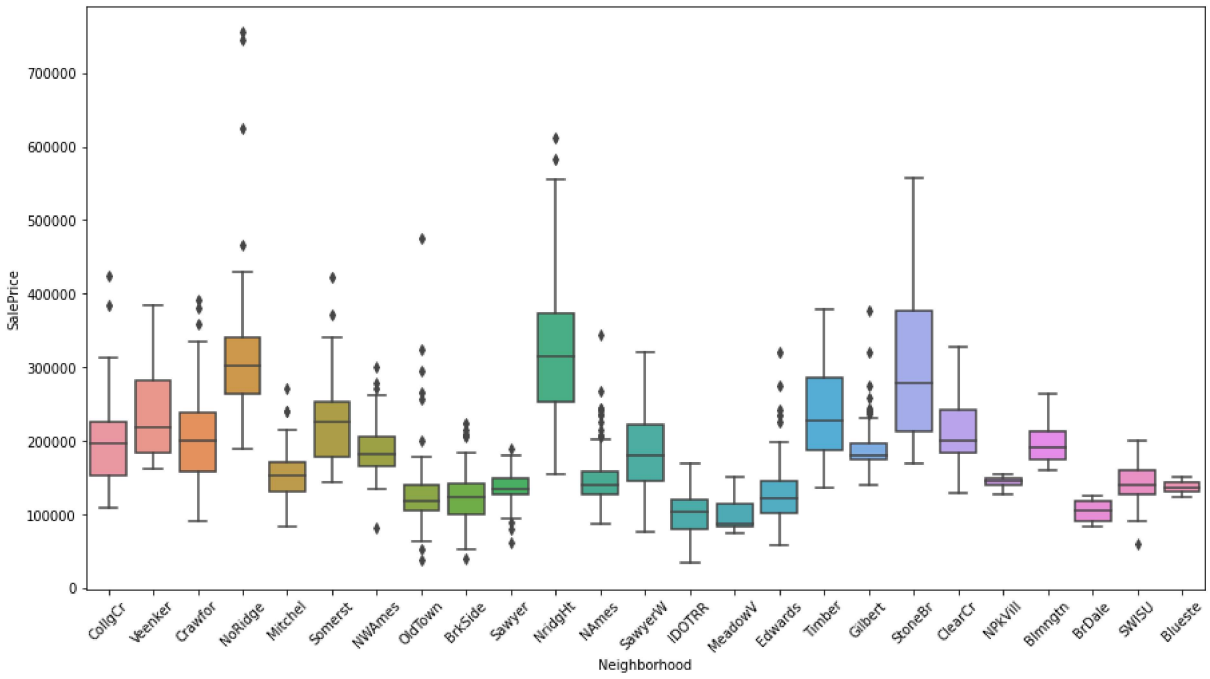
Numeric vs. Categorical

To plot these relations we can use boxplots and swarmplots. We can create these using boxplot function in seaborn. We create a boxplot for numerical variables grouped by categories in categorical variable.

In [ ]:
```
plt.figure(figsize=(15,8))
plt.xticks(rotation = 45)
sns.boxplot('Neighborhood', 'SalePrice', data=dataframe)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
```

Out[31]: <AxesSubplot:xlabel='Neighborhood', ylabel='SalePrice'>

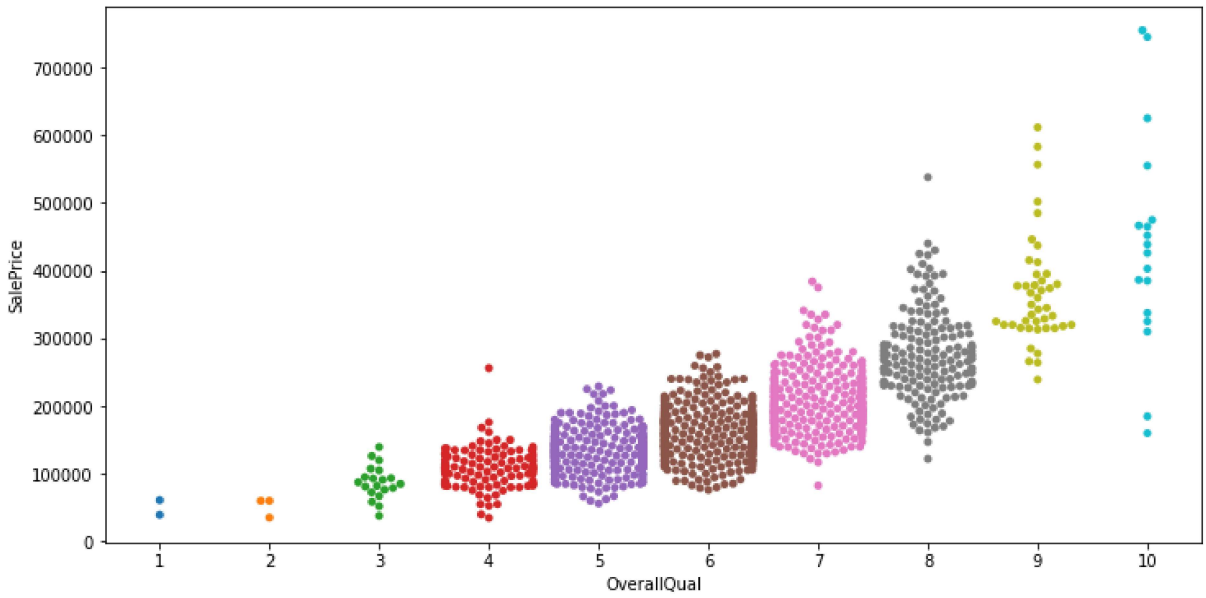


- We can conclude that the distribution of SalePrice changes with the individual neighborhoods and can be a good predictor for it.
- Next we take a look at swarmplots which are similar to boxplots, but they also show no. of points at each value of numerical variable. A denser plot signifies more observations in it.

In [ ]:
```
1 plt.figure(figsize=(12,6))
2 sns.swarmplot('OverallQual', 'SalePrice', data=dataframe)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/categorical.py:1296: UserWarning: 37.1%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/site-packages/seaborn/categorical.py:1296: UserWarning: 71.0%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/site-packages/seaborn/categorical.py:1296: UserWarning: 62.6%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/site-packages/seaborn/categorical.py:1296: UserWarning: 53.0%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/site-packages/seaborn/categorical.py:1296: UserWarning: 20.2%
of the points cannot be placed; you may want to decrease the size of the markers or use
stripplot.
  warnings.warn(msg, UserWarning)
```

Out[32]: <AxesSubplot:xlabel='OverallQual', ylabel='SalePrice'>



Here we can see that there's a marked increase in saleprice as the overall quality increases. So using this variable is a good idea to predict SalePrice

Categorical vs. Categorical

This can be done using using crosstab or graphically using a stacked barplot. We consider two variables 'Neighborhood' and 'OverallQual' and check the relation between them using both of these.

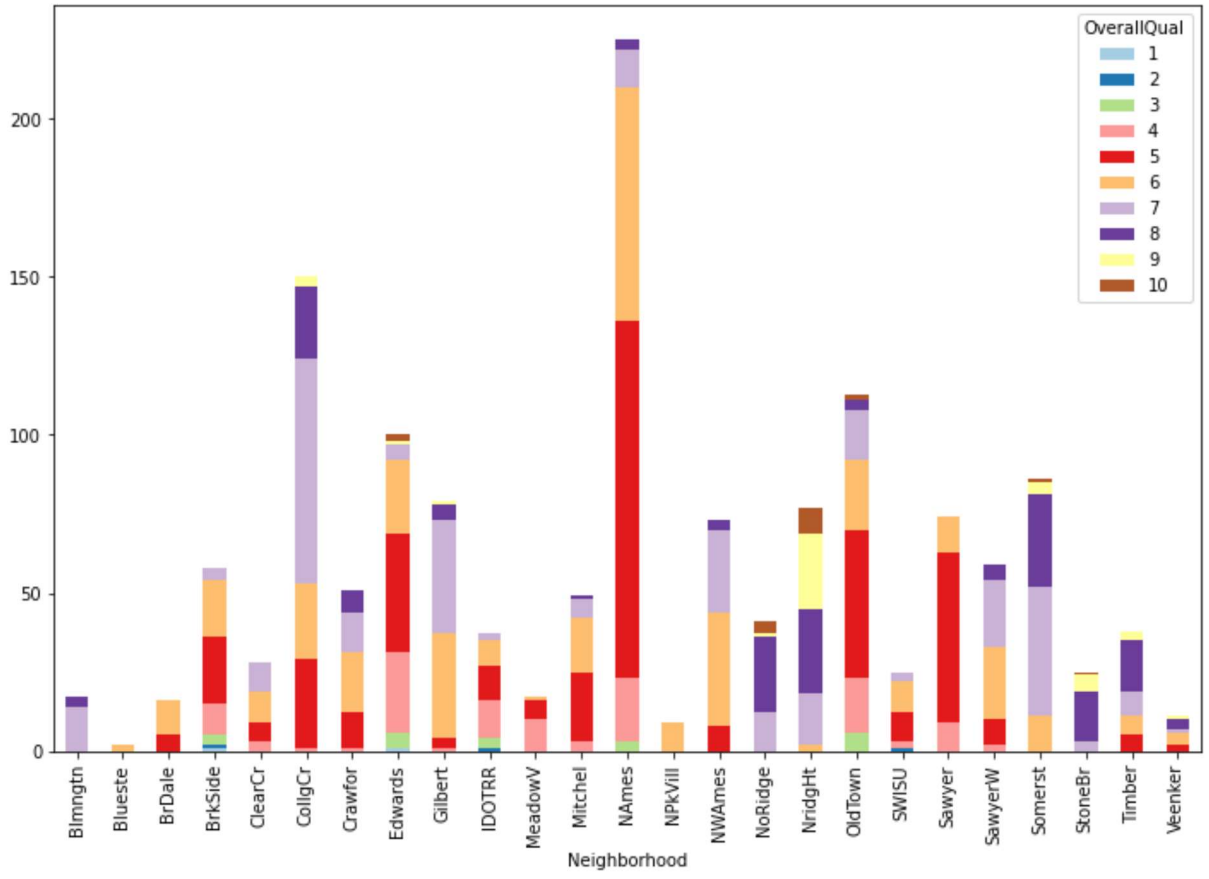
In [ ]:
```
crosstab = pd.crosstab(index=dataframe["Neighborhood"], columns=dataframe["OverallQu
crosstab
```

Out[33]:

OverallQual	1	2	3	4	5	6	7	8	9	10
Neighborhood										
Blmngtn	0	0	0	0	0	0	14	3	0	0
Blueste	0	0	0	0	0	2	0	0	0	0
BrDale	0	0	0	0	5	11	0	0	0	0
BrkSide	1	1	3	10	21	18	4	0	0	0
ClearCr	0	0	0	3	6	10	9	0	0	0
CollgCr	0	0	0	1	28	24	71	23	3	0
Crawfor	0	0	0	1	11	19	13	7	0	0
Edwards	1	0	5	25	38	23	5	0	1	2
Gilbert	0	0	0	1	3	33	36	5	1	0
IDOTRR	0	1	3	12	11	8	2	0	0	0
MeadowV	0	0	0	10	6	1	0	0	0	0

In [ ]:
```
crosstab.plot(kind="bar", figsize=(12,8), stacked=True, colormap='Paired')
```

Out[34]: <AxesSubplot:xlabel='Neighborhood'>



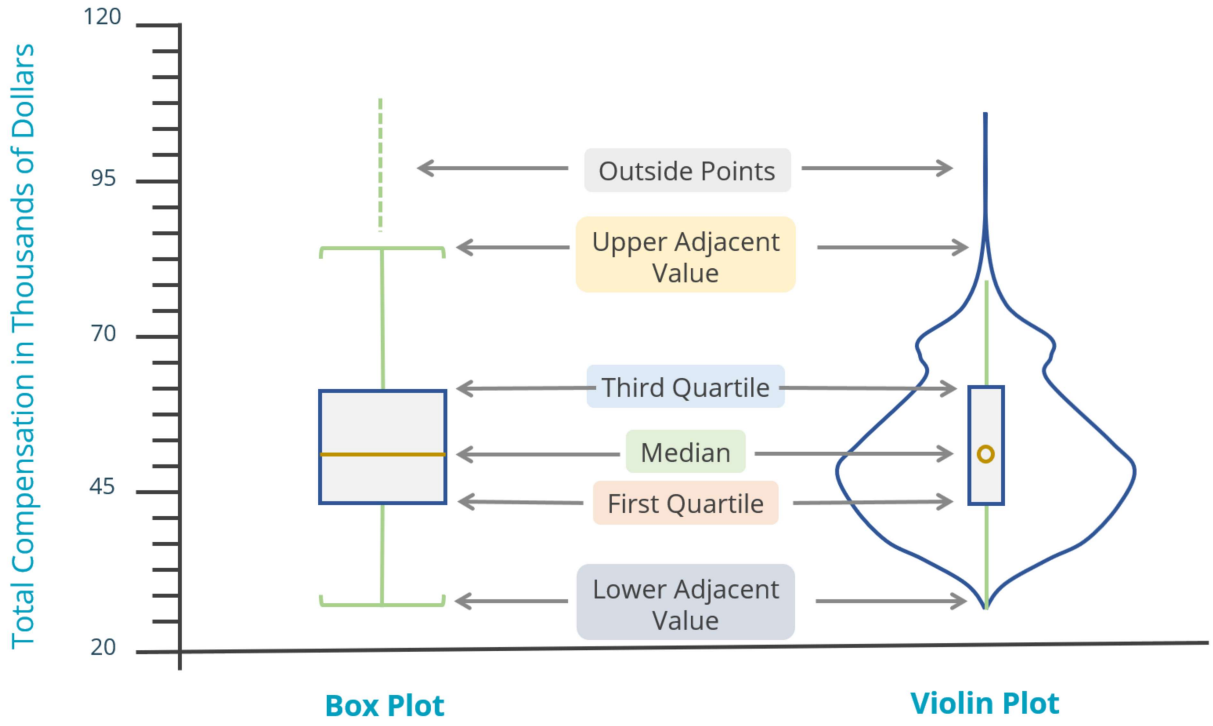
In the above graph, we can see the stacked barplot, which is used to break down and compare parts of a whole. Each bar in the chart represents a whole, and segments in the bar represent different parts or categories of that whole. Here we are considering two variables, 'Neighborhood' and 'OverallQual' to check the relationship between both of these.

Violin Plot

- Violin plots are similar to box plots, but they also display the data's probability density at different values.
- It is a numeric data plotting approach that combines the box and kernel density plots.

- It is more informative than a box plot.
- In general, while a box plot only displays summary statistics including mean, median, and interquartile ranges, a violin plot displays the entire data distribution.

Analysis of the Violin Plot vs. the Boxplot



We can find the same information in a violin plot as we do in a box plot:

- **Median** - It is a white dot on the violin plot.
- **Interquartile Range (IQR)** - It is the black bar in the center of violin.
- **The lower/upper adjacent values -** They are the black lines stretched from the bar and are also defined as the **first quartile - 1.5(IQR)** and the **third quartile + 1.5(IQR)** respectively.

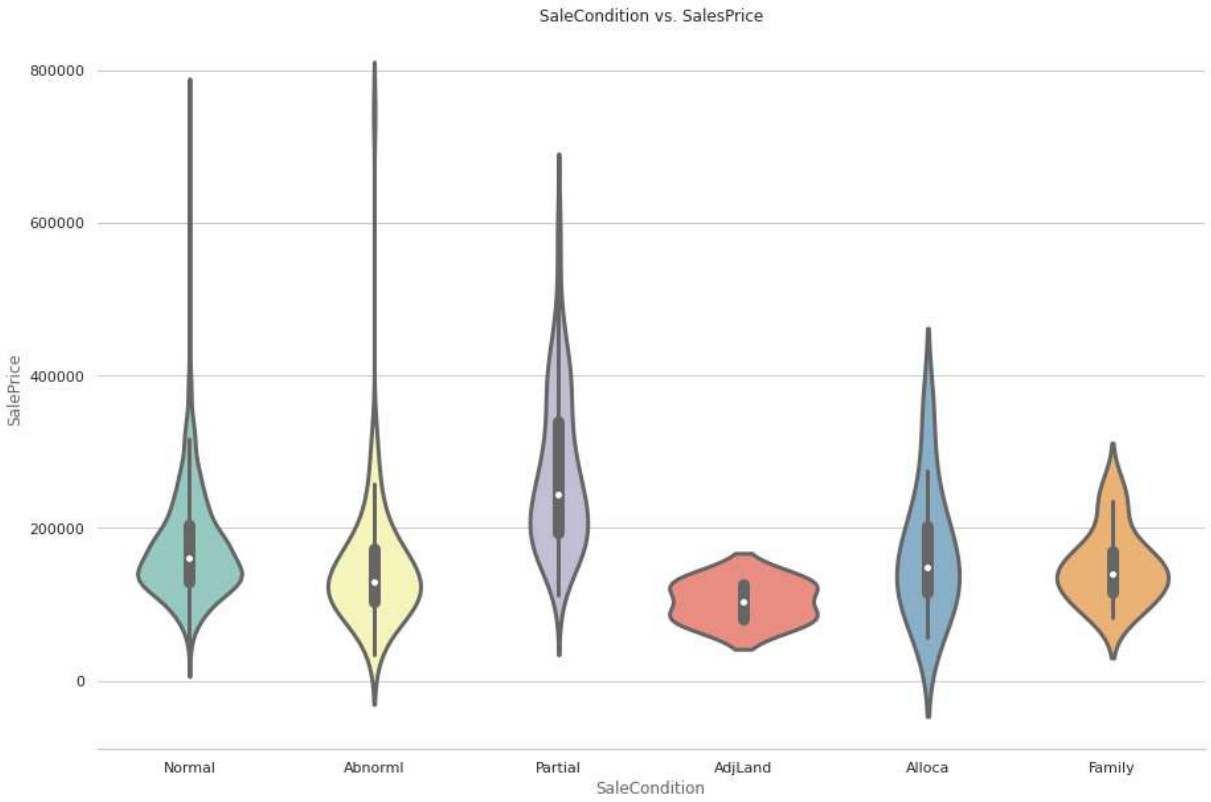
A violin plot contains the same information as a box plot, but has a few additional details which includes:

- A marker for the data's median and a box indicating the interquartile distribution.
- A kernel density estimation that is overlaid on this box plot.
- A violin plot's significance when working with multimodal data (i.e. a distribution with multiple peaks).

Type 1: Vertical Violin Plot

In [ ]:
```
import seaborn as sns
sns.set(style="whitegrid")

f, ax = plt.subplots(figsize=(15, 10))

# Show each distribution with both violins and points
sns.violinplot(x="SaleCondition",y="SalePrice",data=dataframe, inner="box", palette=

sns.despine(left=True)

ax.set_title('SaleCondition vs. SalesPrice')
ax.set_xlabel("SaleCondition", alpha=0.7)
ax.set_ylabel("SalePrice", alpha=0.7)
```

Out[35]:
```
Text(0, 0.5, 'SalePrice')
```



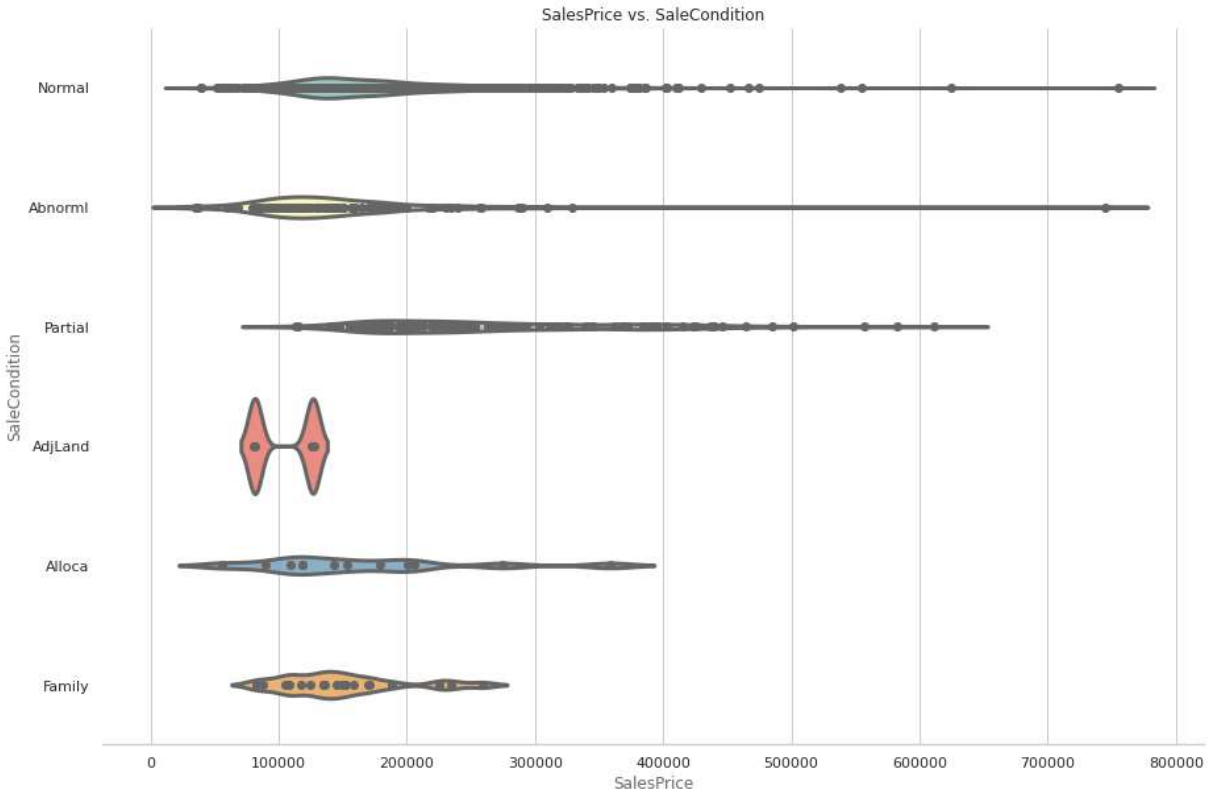
Observations:

The above violin plots show the relationship between SaleCondition and SalePrice. The box plot elements show that the median for AdjLand is lower than other types.

Type 2: Horizontal Violin Plot

In [ ]:
```
import seaborn as sns
sns.set(style="whitegrid")

f, ax = plt.subplots(figsize=(15, 10))

# Show each distribution with both violins and points
sns.violinplot(x="SalePrice",y="SaleCondition",data=dataframe, palette="Set3", inner

sns.despine(left=True)

ax.set_title('SalesPrice vs. SaleCondition')
ax.set_xlabel("SalesPrice", alpha=0.7)
ax.set_ylabel("SaleCondition", alpha=0.7)
```

Out[36]: Text(0, 0.5, 'SaleCondition')

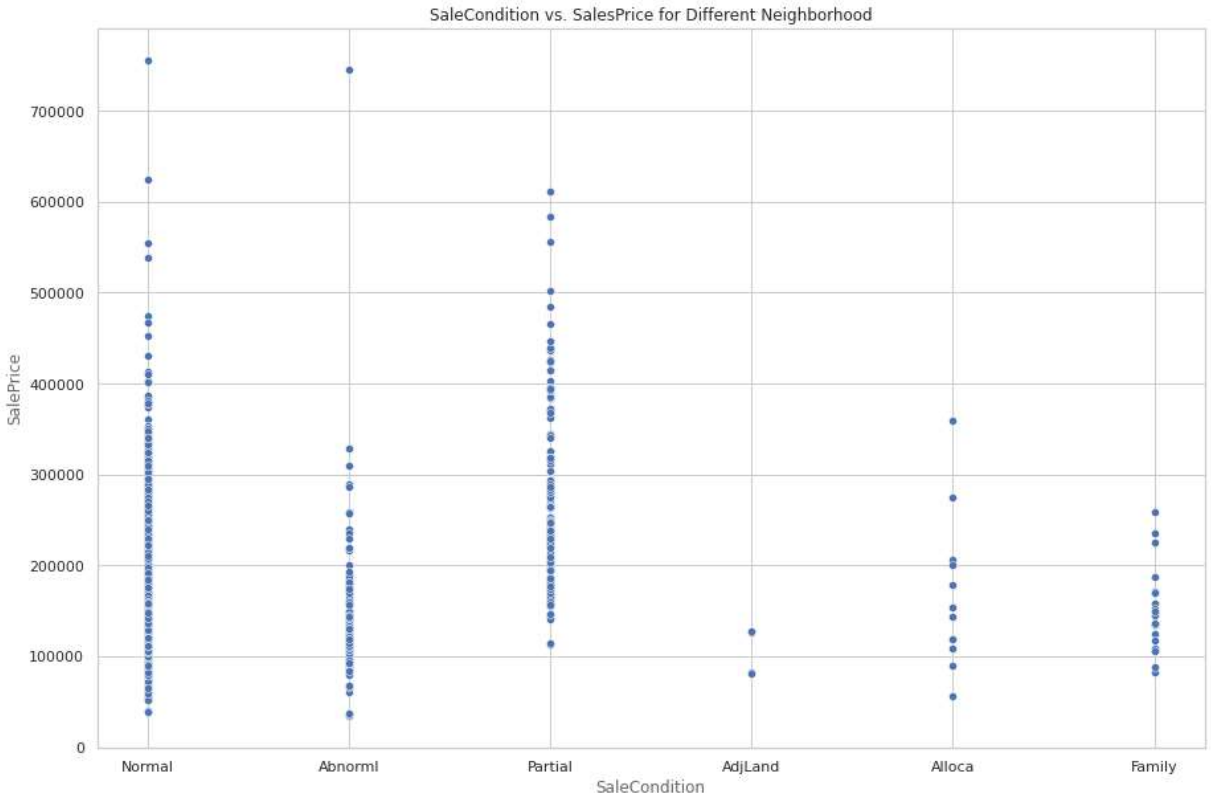


Like vertical bar charts, horizontal violin plots are ideal for dealing with many categories. Swapping axes gives the category labels more room to breathe.

Scatter Plot

In [ ]:
```
import seaborn as sns
sns.set(style="whitegrid")

f, ax = plt.subplots(figsize=(15, 10))

sns.scatterplot(data=dataframe, x="SaleCondition", y="SalePrice", legend=False, size

ax.set_title('SaleCondition vs. SalesPrice for Different Neighborhood')
ax.set_xlabel("SaleCondition", alpha=0.7)
ax.set_ylabel("SalePrice", alpha=0.7)
```

Out[37]:
```
Text(0, 0.5, 'SalePrice')
```



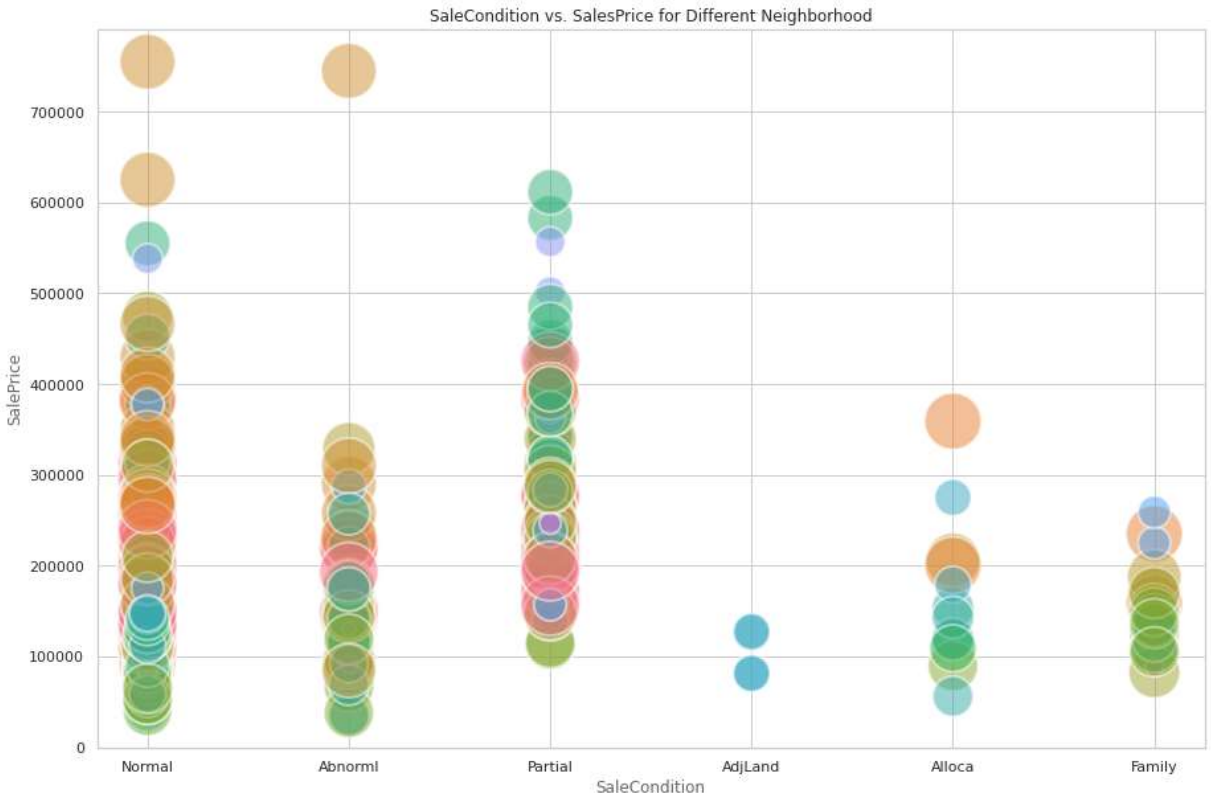
Bubble Plot

- A bubble plot is an extension of a scatter plot and replaces the dots with bubbles. It is used to investigate interactions between three numeric variables.
- In a two-dimensional plot, it shows data as a cluster of circles (bubbles).
- Each bubble in a bubble plot represents a single data point, and the horizontal position, vertical position, and dot size indicate the variables' values for each point.
- The XY coordinates, the size of the bubble, and the color of the bubbles are all necessary data required to create a bubble plot.
- The library can provide the colors required for the bubbles in the bubble plot.

In [ ]:

```
import seaborn as sns
sns.set(style="whitegrid")

f, ax = plt.subplots(figsize=(15, 10))

sns.scatterplot(data=dataframe, x="SaleCondition", y="SalePrice", size="Neighborhood
ax.set_title('SaleCondition vs. SalesPrice for Different Neighborhood')
ax.set_xlabel("SaleCondition", alpha=0.7)
ax.set_ylabel("SalePrice", alpha=0.7)
```

Out[38]: Text(0, 0.5, 'SalePrice')



Analysis of Data in a Bubble Plot

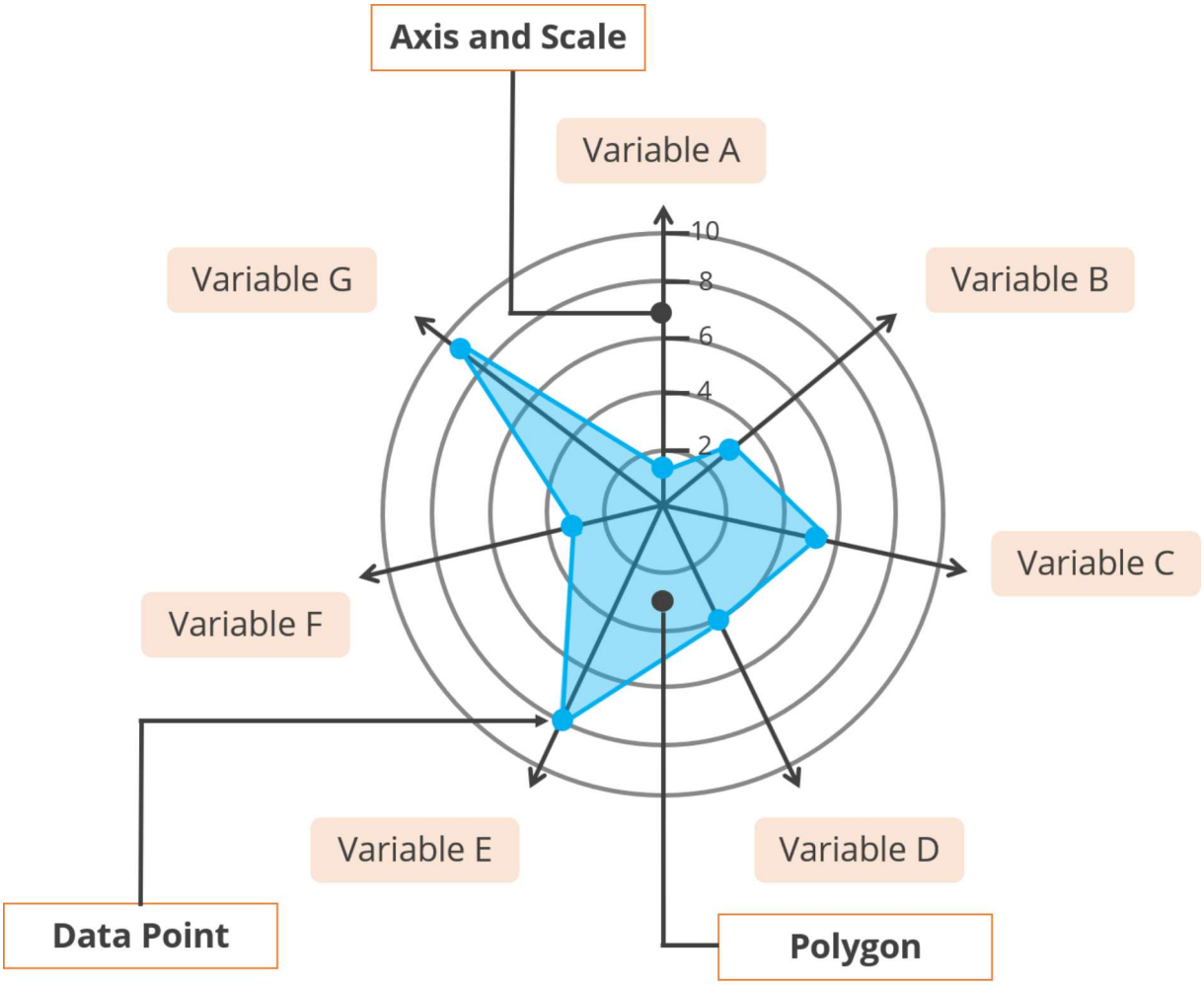
- In comparison to a scatter plot, the addition of a marker scale as a factor in a bubble plot allows for the analysis of three variables instead of only two.
- Three different pairwise comparisons (X vs. Y, Y vs. Z, and X vs. Z) as well as a three-way comparison can be made in a single bubble plot.
- To obtain the same amount of insights as the bubble plot, several two-variable scatter plots would be required; even then, inferring a three-way relationship between data points would not be as straightforward as in a bubble plot.
- A three-column data table is used to make a bubble chart. The horizontal and vertical positions of each point will be represented by two columns, while the size of each point will be represented by a third column. For each row in the table, one point will be plotted.

Radar Chart

- A radar chart (also known as spider chart, web chart, polar chart, or star plot) is a two-dimensional chart that displays multivariate data in the form of quantitative variables represented on axes emanating from the centre.
- Each variable has its own axis, which are arranged in a radial pattern around the central point.
- The aim of a radar chart is to visually represent one or more groups of values over multiple variables.

- The variable order has a significant effect on the presentation of the chart, scales can be skewed, overplotting makes it difficult to interpret, and over-evaluation of differences.
- They are useful to determine which variables have similar values and if certain variables have outliers.
- They can also be used to see which variables in a dataset have high or low scores, making them ideal to visualize performance.
- Radar charts are great to visualize comparisons between observations because they allow you to compare multiple attributes among different observations to see how they stack up.
- The observation with the highest polygon region should be the best if you're looking at overall performance, so it's easy to see overall "top performers" in the radar chat.

Analysis of Radar Chart



- Each variable has its own axis, which starts at the centre.
- All axes are arranged radially, with equal distances between them while ensuring that they also have the same scale.
- Grid lines that link axes are often used as guides.
- The value of each variable is plotted along its own axis, and all the variables in a dataset are connected to form a polygon.
- However, using too many variables results in too many axes, making the map difficult to interpret. Thus, keeping the radar charts clear and limiting the amount of variables used is a good practice.

Example - Radar Chart

Consider the data related to a specific food item of a specific restaurant which is delivered by an online delivery service.

The data is rated on a scale of 1 to 5 for different categories

Plotly Express is a high-level interface which is easy to use and works on different types of data and style figures.

Use line_close=True for closed lines.

In [ ]:
```
import plotly.express as px
import pandas as pd
food_data = pd.DataFrame(dict(
    item_rating=[1, 5, 2, 4, 5],
    criteria=['Packaging Cost','Food Quality','Ingredients Cost',
           'Market Demand', 'Customer Rating']))

food_data.head()
```

Out[3]:

	item_rating	criteria
0	1	Packaging Cost
1	5	Food Quality
2	2	Ingredients Cost
3	4	Market Demand
4	5	Customer Rating

For a filled line in a Radar Chart, use px.line_polar with fig.update_traces.

In [ ]:
```
fig = px.line_polar(food_data, r='item_rating', theta='criteria', line_close=True)
fig.update_traces(fill='toself')
fig.show()
```

In [ ]:
```
plotlyexpress.__version__
```

Note: In this lesson, we saw the use of the exploratory data analysis methods, but in the next lesson we are going to use one of these methods as a sub component for "Feature Selection".