Housing Project

Introduction:

Data Description.csv: This contains the description of data.

train.csv: This contains the dataset on which you will be working upon

test.csv : Predict the output for these data with your best fit model.

Housing Use case: This contains the problem statement and business goal.

Data fields:

Here's a brief version of what you'll find in the data description file.

- **SalePrice** - the property's sale price in dollars. This is the target variable that you're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access
- **Alley**: Type of alley access
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling
- **HouseStyle**: Style of dwelling
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation
- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement
- **BsmtExposure**: Walkout or garden level basement walls

- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area
- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **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**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **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
- **Fence**: Fence quality
- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

Problem Statement/Problem Definition:

Prices of real estate properties are sophisticatedly linked with our economy. Despite this, we do not have accurate measures of housing prices based on the vast amount of data available. Therefore, the goal of this project is to use machine learning to predict the selling prices of houses based on many economic factors.

Data Overview:

```
df=pd.read_csv("train housing.csv")
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

5 rows × 81 columns

About the data:

1. Number of data points in train data:**1460**

2. Number of features in train data: **81**

3. Number of data points in test data: **1459**

4. Number of features in test data: **80**

Data Pre-processing:

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
 14  Condition2     1168 non-null   object
 15  BldgType       1168 non-null   object
 16  HouseStyle     1168 non-null   object
 17  OverallQual    1168 non-null   int64
 18  OverallCond    1168 non-null   int64
 19  YearBuilt      1168 non-null   int64
 20  YearRemodAdd   1168 non-null   int64
 21  RoofStyle      1168 non-null   object
 22  RoofMatl       1168 non-null   object
 23  Exterior1st    1168 non-null   object
 24  Exterior2nd    1168 non-null   object
 25  MasVnrType     1161 non-null   object
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object
 28  ExterCond      1168 non-null   object
 29  Foundation     1168 non-null   object
 30  BsmtQual       1138 non-null   object
 31  BsmtCond       1138 non-null   object
```

- Dataset has two data types: float64, object and integer values.

- Except for the Lot Frontage, Alley columns every column has missing values.

Let's generate descriptive statistics for the dataset using the function describe () in pandas.

Finding Categorical and Numerical Features in a Data set:
#Categorical & Numerical features in Dataset:

List of categorical & Numerical columns

```
numCol=[]
catCol=[]

for col in df.columns:
    if df[col].dtype=='O':
        catCol.append(col)
    else:
            numCol.append(col)
```

```
print("List of categorical columns:",catCol)
```

```
List of categorical columns: ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'M
asVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'He
ating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'Gar
ageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']
```

```
print("List of numerical columns:",numCol)
```

```
List of numerical columns: ['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemod
Add', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivAre
a', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'Gara
geYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'Mi
scVal', 'MoSold', 'YrSold', 'SalePrice']
```

Missing Value Imputation:

There are different ways of handling missing values in the data. We can delete those observations or can fill them with statistical measures. In this case, statistical measures like mode and mean have been used to replace missing values in categorical and numerical variables, respectively.

Machine learning algorithms can't handle missing values and cause problems. So, they need to be addressed in the first place. There are many techniques to identify and impute missing values.

If a dataset contains missing values and loaded using pandas, then missing values get replaced with NaN (Not a Number) values. These NaN values can be identified using methods like ***isna ()*** or ***isnull ()*** and they can be imputed using ***fillna ().*** This process is known as **Missing Data Imputation**.

Missing Value Analysis of Dataset

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

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      214
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

Exploratory Data Analysis (EDA):
1-Descriptive Statistics

Descriptive Statistics

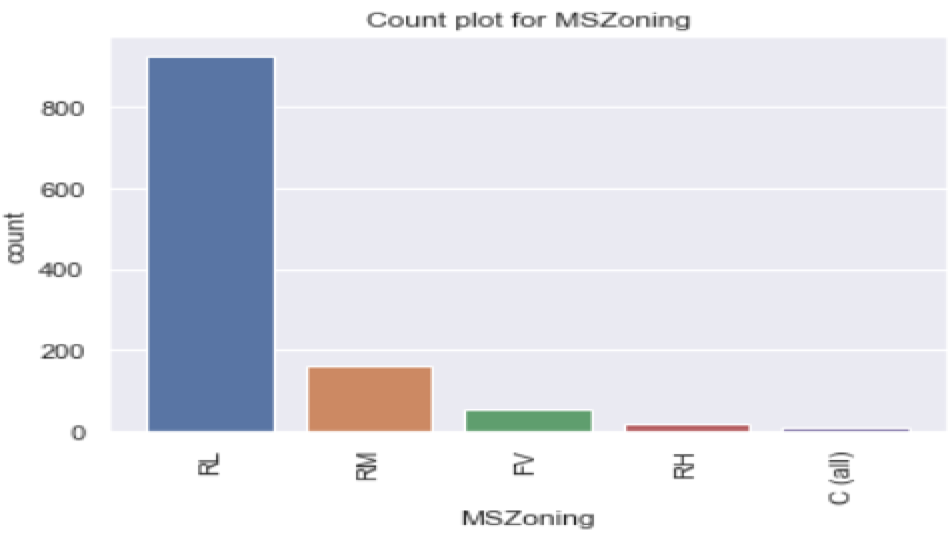
```
df.describe(include='all')
```

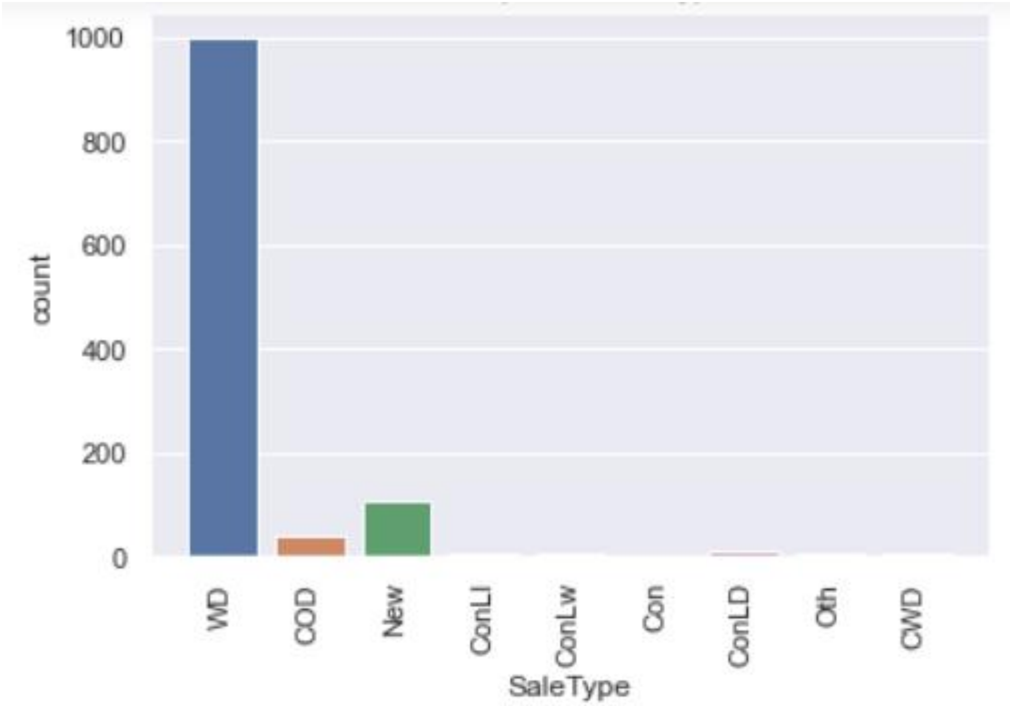
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence
count	1168.000000	1168.000000	1168	954.00000	1168.000000	1168	77	1168	1168	1168	...	1168.000000	7	237
unique	NaN	NaN	5	NaN	NaN	2	2	4	4	1	...	NaN	3	4
top	NaN	NaN	RL	NaN	NaN	Pave	Grvl	Reg	Lvl	AllPub	...	NaN	Gd	MnPrv
freq	NaN	NaN	928	NaN	NaN	1164	41	740	1046	1168	...	NaN	3	129
mean	724.136130	56.767979	NaN	70.98847	10484.749144	NaN	NaN	NaN	NaN	NaN	...	3.448630	NaN	NaN
std	416.159877	41.940650	NaN	24.82875	8957.442311	NaN	NaN	NaN	NaN	NaN	...	44.896939	NaN	NaN
min	1.000000	20.000000	NaN	21.00000	1300.000000	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN
25%	360.500000	20.000000	NaN	60.00000	7621.500000	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN
50%	714.500000	50.000000	NaN	70.00000	9522.500000	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN
75%	1079.500000	70.000000	NaN	80.00000	11515.500000	NaN	NaN	NaN	NaN	NaN	...	0.000000	NaN	NaN
max	1460.000000	190.000000	NaN	313.00000	164660.000000	NaN	NaN	NaN	NaN	NaN	...	738.000000	NaN	NaN

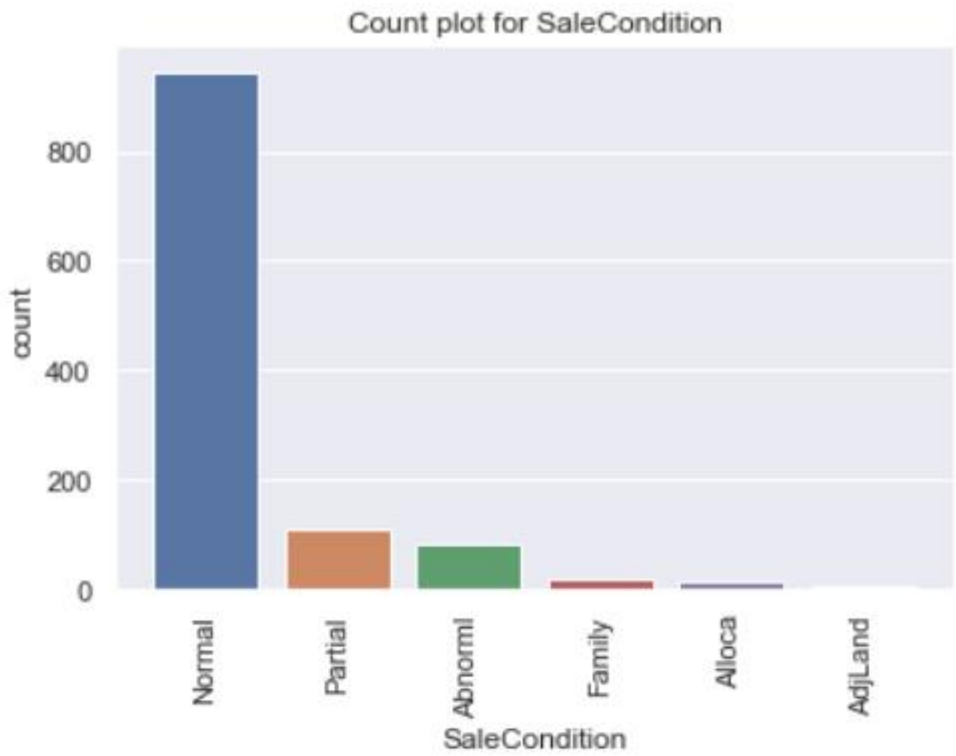
Id ranges from 1.0 to 1460.00 with a standard deviation of 416.15.
More insights can be derived from descriptive statistics. The idea is to get a feel of the data and later on depending on requirements; different parameters can be assessed.

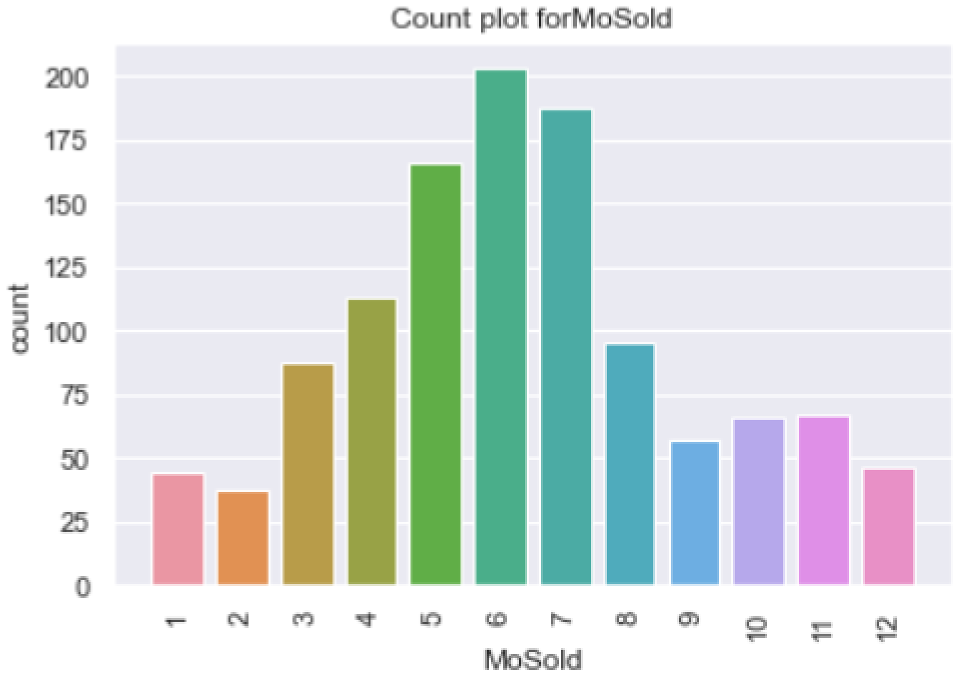
2-Univariate Analysis

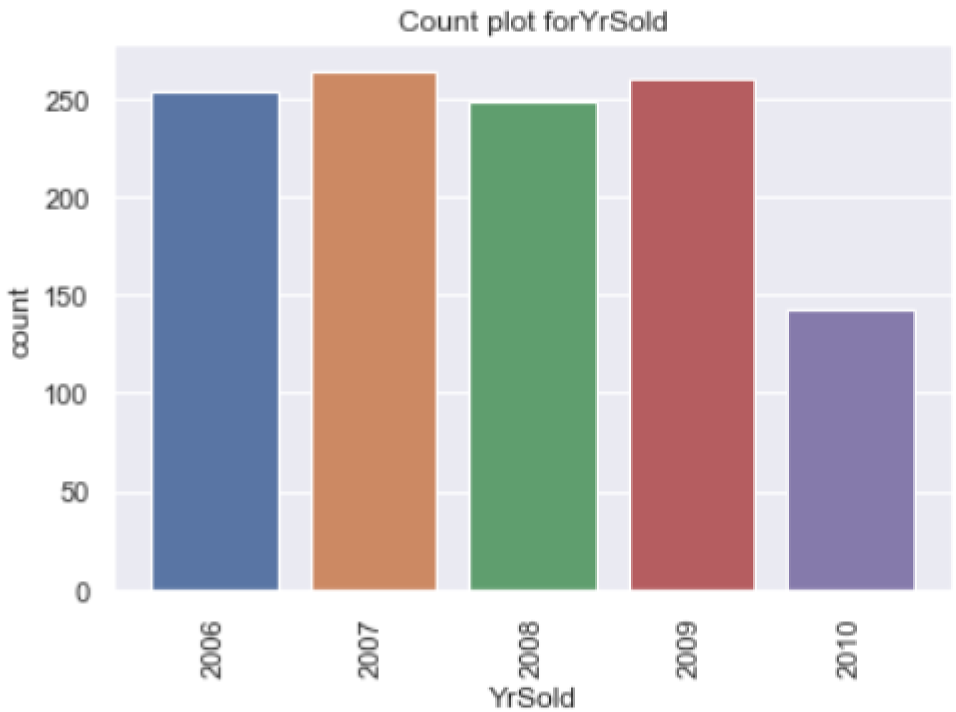
```
for i in catCol:
    plt.figure()
    sns.set_theme(style="darkgrid")
    sns.countplot(df[i])
    plt.xticks(rotation=90)
    plt.title(f"Count plot for {i}")
    plt.plot()
    plt.show()
```




1000
800
600
400
200
0
count
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SaleType


Count plot for SaleCondition
800
600
400
200
0
count
Normal
Partial
Abnorml
Family
Alloca
AdjLand
SaleCondition





Outliers’ detection and treatment:

What is an outlier?

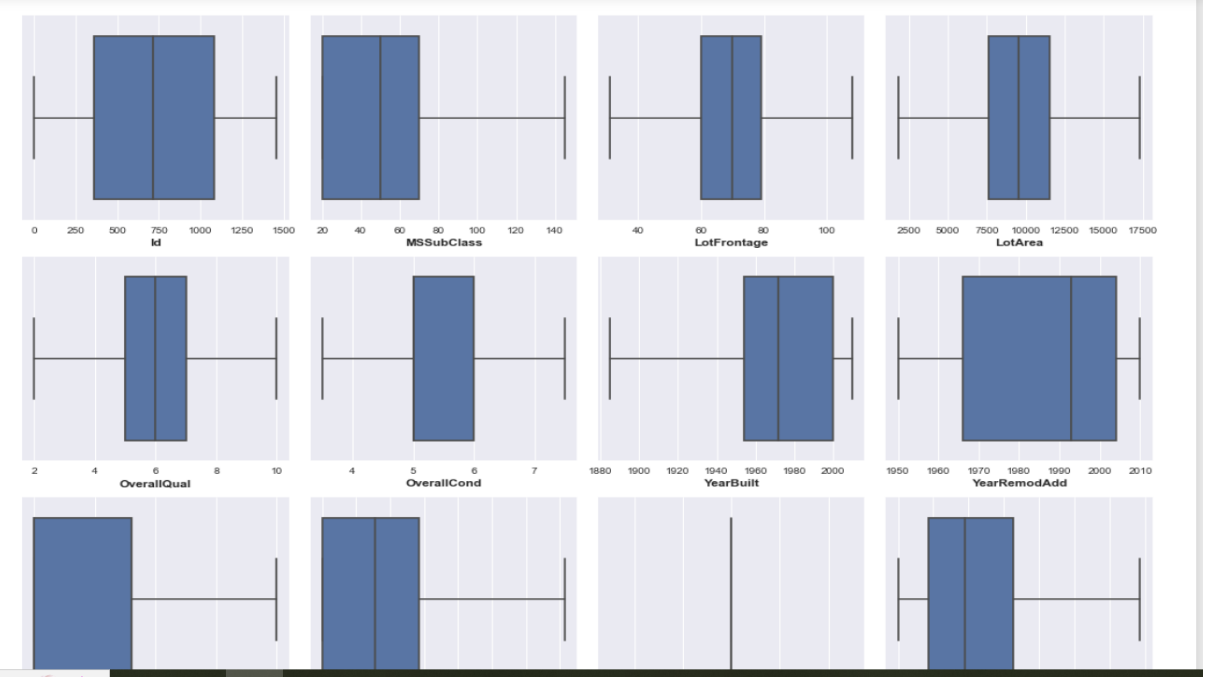
An Outlier is an observation that lies an abnormal distance from other values in a given sample. They can be detected using visualization (like boxplots, scatter plots), Z-score, statistical and probabilistic algorithms, etc.

 # Outlier Treatment to remove outliers from Numerical Features:

Removing Outliers

```
lsUpper = []
lsLower = []
def removeOutliers(numerical):
    for i in range(len(numerical)):
        q1 = df[numerical[i]].quantile(0.25)
        q3 = df[numerical[i]].quantile(0.75)
        IQR = q3-q1
        minimum = q1 - 1.5 * IQR
        maximum = q3 + 1.5 * IQR
        df.loc[(df[numerical[i]] <= minimum), numerical[i]] = minimum
        df.loc[(df[numerical[i]] >= maximum), numerical[i]] = maximum
removeOutliers(numerical)
```

```
num_of_rows = 4
num_of_cols = 4
fig, ax = plt.subplots(num_of_rows, num_of_cols, figsize=(15,15))
print(numerical)
i=0;j=0;k=0;
while i<num_of_rows:
    while j<num_of_cols:
        sns.boxplot(df[numerical[k]], ax=ax[i, j])
        k+=1;j+=1
    j=0;i+=1
plt.savefig('after_removing_outliers_from_numerical_columns.png')
plt.show()
```



Mapping the real-world problem to a Machine Learning Problem:

This problem involves predicting the prices of the houses which are continuous and real valued outputs. Thus, this is a **Regression Problem.**

```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())

display_scores(xgb_scores)
```

```
Scores: [0.13180314 0.13304477 0.16400615 0.16988014 0.1423638  0.1529494
 0.15570337 0.1483882  0.15781064 0.17434866]
Mean: 0.1530298265311983
Standard deviation: 0.013690139889082424
```

```
scores = cross_val_score(rf, X_prepared, log_y,
                         scoring="neg_mean_squared_error", cv=10)
# Scikit-Learn’s cross-validation features expect a utility function
# (greater is better) rather than a cost function (lower is better)
rf_scores = np.sqrt(-scores)
display_scores(rf_scores)
```

```
Scores: [0.13328741 0.12901392 0.14790134 0.1600432  0.11702999 0.1741792
 0.14693491 0.1397776  0.15471056 0.18681641]
Mean: 0.14896945342721601
Standard deviation: 0.01994382525397305
```

```
final_model = RandomForestRegressor(bootstrap=False, max_depth=18, max_features='sqrt',
                                    n_estimators=1650, random_state=10)
final_model.fit(X_prepared, log_y)
```

```
RandomForestRegressor(bootstrap=False, max_depth=18, max_features='sqrt',
                      n_estimators=1650, random_state=10)
```

```
score_model(final_model)
```

```
0.0018557515558789194
```