Predicting Sales Price for Homes in the Ames Housing Dataset

Springboard Data Science - Capstone Project 2 - Presentation

Project Goals

Following are the project goals which we will try to accomplish in this project:

- Predict final sales price of each home in the dataset with acceptable accuracy and precision
- Identify which key variables describing home have significant influence on the sale price
- Apply feature engineering to optimize results and effectivity of the model
- Apply advanced regression techniques like random forest and gradient boosting

Target Applications of The Exercise

Real estate is by far the largest industry in the USA. Real Estate, renting, and leasing constitutes the largest sector of the United States' economy with the GDP value added of $1.898 trillion accounting for 13% of the national GDP.

- The Machine Learning model can be used by various players in the industry like FinTech Companies, Real Estate Agents, Home Builders, Government Authorities, City Planners, Home Buyers, Renters, etc.
- Ability to predict prices and information about the key features which influence Sales Price would be helpful in making informed decisions about buying, selling, building and planning real estate units.

Key Sections Ahead

Key Sections Ahead

- Dataset
- Data Wrangling
- Data Story
- In-Depth Analysis using ML Techniques
- Closing Thoughts, Conclusion

Dataset

Ames Housing Data Set

- Kaggle Link for the dataset: https://www.kaggle.com/c/house-prices-advanced-regression-techniques
- The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.
- With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, the goal is to predict the final price of each home.

Data Set

File descriptions

- train.csv - the training set
- test.csv - the test set

Data Fields

- **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

Data Fields

- **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

Data Fields

- **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

Data Wrangling

EDA

```
In [6]: test.head()
```

Out[6]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeatu
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	120	0	NaN	MnPrv	N
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	G
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	MnPrv	N
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	N
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	144	0	NaN	NaN	N

5 rows × 80 columns

```
In [7]: train.head()
```

Out[7]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows × 81 columns

EDA

In [8]:
```
train.columns
```

Out[8]:
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

EDA

In [10]:
```
train.describe()
```

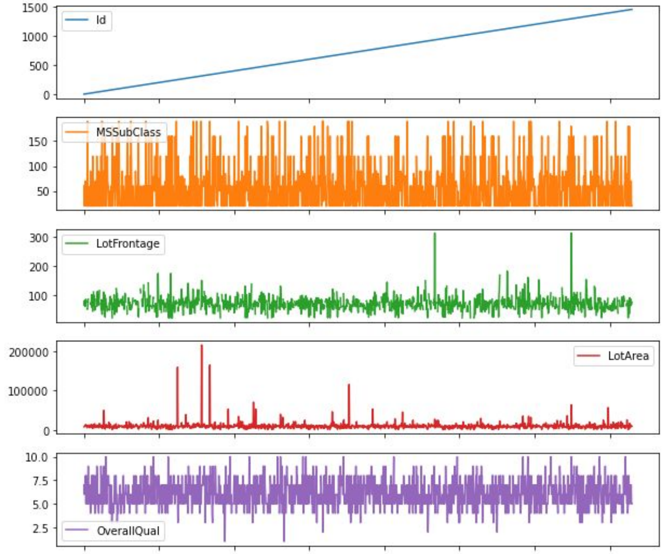
Out[10]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDec
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	...	1460.00
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726	...	94.24
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091	...	125.33
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	...	0.00
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	...	0.00
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	...	0.00
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000	...	168.00
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.00

8 rows × 38 columns

EDA

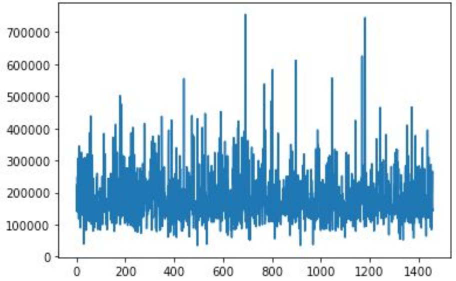
```
In [11]: train.plot(subplots=True, figsize=(10,80))
         plt.show()
```



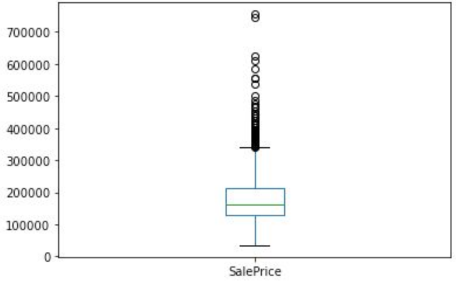
- I have plotted all the attributes in my iPython notebook. Please refer to the iPython notebook for the project for more details. Following is just an excerpt of the original plot output.
- Majority of the attributes have records with values normally spread between the normal range spectrum with minimal or mostly no outliers in the distribution
- However there are a few attributes which have a few outliers These are as follows - LotFrontage, LotArea, BsmtFinSF1, TotalBsmSF, 1stFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, 3SsnPorch, PoolArea, MiscVal

EDA

In [13]:
```
# plt.subplot(191)
train.SalePrice.plot()
plt.show()
```



In [14]:
```
train.SalePrice.plot(kind='box')
plt.show()
```



We can deduce following findings using the data above

- Majority of the houses are in the range of 100,000 to 200,000. There are a few outliers with prices beyond the 400K price range.

EDA

Data Set

Visual inspection and analysis of the dataset indicates that following attributes could have a greater impact on determining the sale price of the house over other attributes. However this is just the initial assessment and this may change as we dive deeper in the data using statistical and regression techniques.

- MSSubClass: Identifies the type of dwelling involved in the sale.
- MSZoning: Identifies the general zoning classification of the sale.
- LotArea: Lot size in square feet
- Utilities: Type of utilities available
- Neighborhood: Physical locations within Ames city limits
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- ExterQual: Evaluates the quality of the material on the exterior
- ExterCond: Evaluates the present condition of the material on the exterior
- Foundation: Type of foundation
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- GrLivArea: Above grade (ground) living area square feet
- KitchenQual: Kitchen quality
- GarageType: Garage location
- PoolQC: Pool quality!

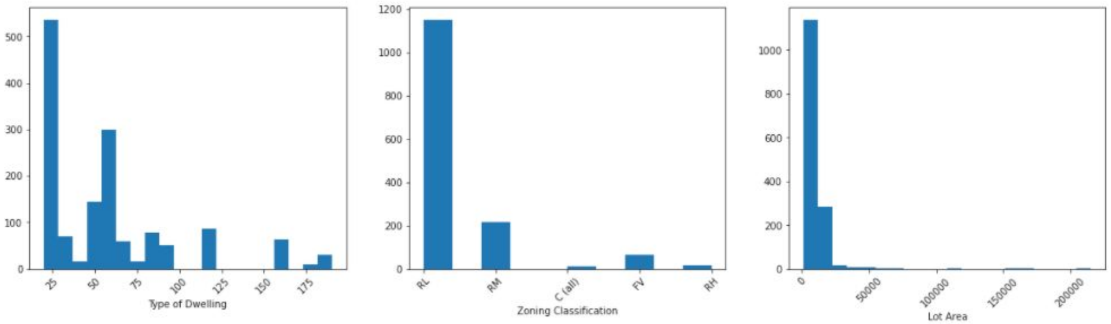
EDA

In [15]:
```
plt.figure(figsize=(20,5))
plt.subplot(131)
plt.hist(train.MSSubClass, bins=20)
plt.xticks(rotation=45)
plt.xlabel('Type of Dwelling')

plt.subplot(132)
plt.hist(train.MSZoning)
plt.xticks(rotation=45)
plt.xlabel('Zoning Classification')

plt.subplot(133)
plt.hist(train.LotArea, bins=20)
plt.xticks(rotation=45)
plt.xlabel('Lot Area')
```

Out[15]:
```
Text(0.5, 0, 'Lot Area')
```



- Large number of dwellings are 1-story 1946 & Newer All Styles category (value = 20)
- Most of the houses fall sidential Low Density zone (value = RL)
- Most of the houses have lot area in the range of 0 to 10K and 10k to 20K

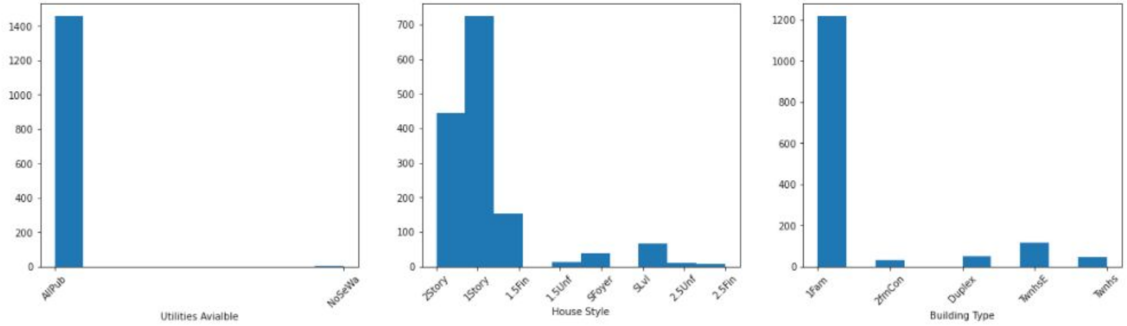
EDA

In [16]:
```
plt.figure(figsize=(20,5))
plt.subplot(131)
plt.hist(train.Utilities)
plt.xticks(rotation=45)
plt.xlabel('Utilities Avialble')

plt.subplot(132)
plt.hist(train.HouseStyle)
plt.xticks(rotation=45)
plt.xlabel('House Style')

plt.subplot(133)
plt.hist(train.BldgType)
plt.xticks(rotation=45)
plt.xlabel('Building Type')
```

Out[16]:
```
Text(0.5, 0, 'Building Type')
```

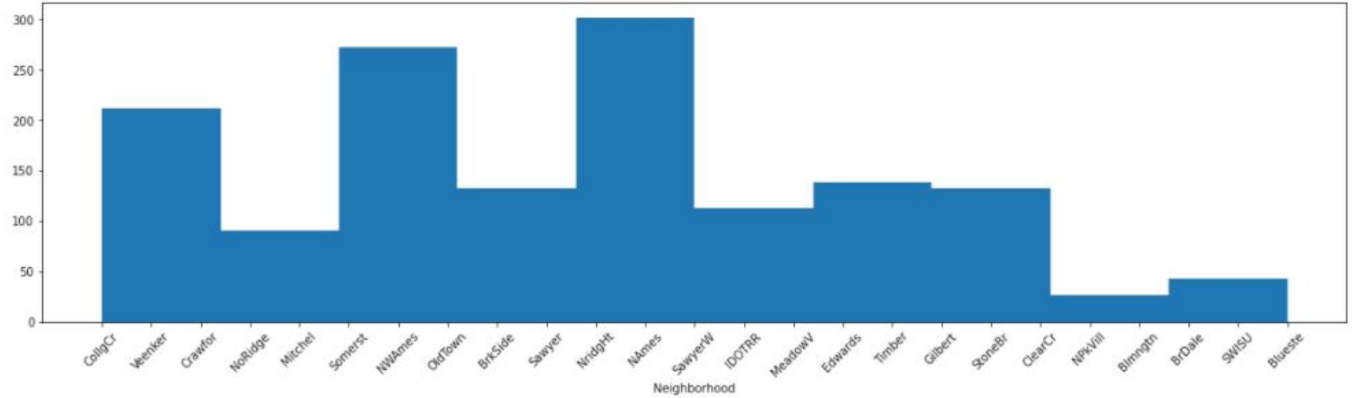


- All houses have all the utilities available
- Most of the houses are either 1 story or 2 story
- Most of the houses are of single familiy type

EDA

```
In [17]: plt.figure(figsize=(20,5))
         plt.hist(train['Neighborhood'])
         plt.xticks(rotation=45)
         plt.xlabel('Neighborhood')

Out[17]: Text(0.5, 0, 'Neighborhood')
```



- Top neighborhoods where large number of houses are located are Northridge Heights, North Ames, Somerset, Northwest Ames, College Creek and Veenker

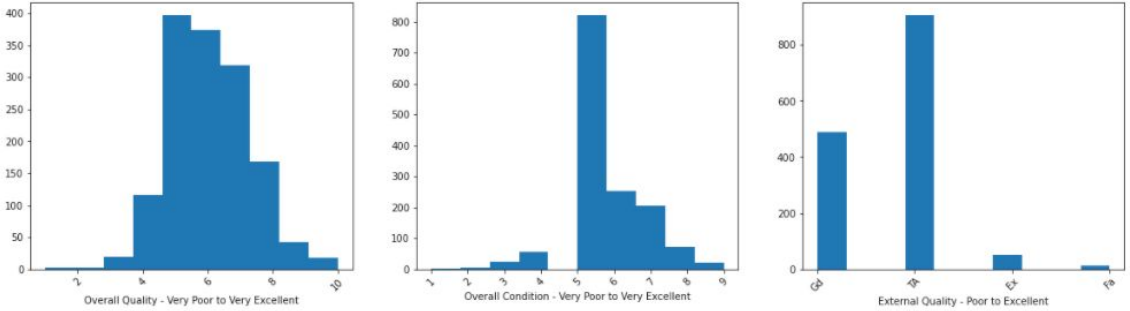
EDA

In [18]:
```
plt.figure(figsize=(20,5))
plt.subplot(131)
plt.hist(train.OverallQual, bins=10)
plt.xticks(rotation=45)
plt.xlabel('Overall Quality - Very Poor to Very Excellent')

plt.subplot(132)
plt.hist(train.OverallCond)
plt.xticks(rotation=45)
plt.xlabel('Overall Condition - Very Poor to Very Excellent')

plt.subplot(133)
plt.hist(train.ExterQual)
plt.xticks(rotation=45)
plt.xlabel('External Quality - Poor to Excellent')
```

Out[18]:
```
Text(0.5, 0, 'External Quality - Poor to Excellent')
```



- Most of the houses fall in Average (5) to Very Good range for Overall Quality (8)
- Similalry, Overall Condition for most of the houses range from Average (5) to Very Good (8)
- External Quality ranges from Average to Good

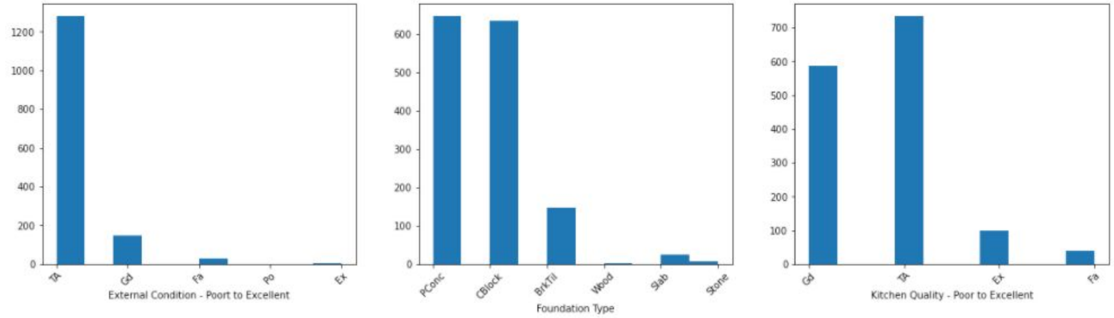
EDA

In [19]:
```
plt.figure(figsize=(20,5))
plt.subplot(131)
plt.hist(train.ExterCond)
plt.xticks(rotation=45)
plt.xlabel('External Condition - Poort to Excellent')

plt.subplot(132)
plt.hist(train.Foundation)
plt.xticks(rotation=45)
plt.xlabel('Foundation Type')

plt.subplot(133)
plt.hist(train.KitchenQual)
plt.xticks(rotation=45)
plt.xlabel('Kitchen Quality - Poor to Excellent')
```

Out[19]:
```
Text(0.5, 0, 'Kitchen Quality - Poor to Excellent')
```



- Most of the houses are of Average External Condition
- Foundation type is primarily Poured Concrete or Ciner Block type
- Kitchn Quality ranges from Average to Good for most of the houses

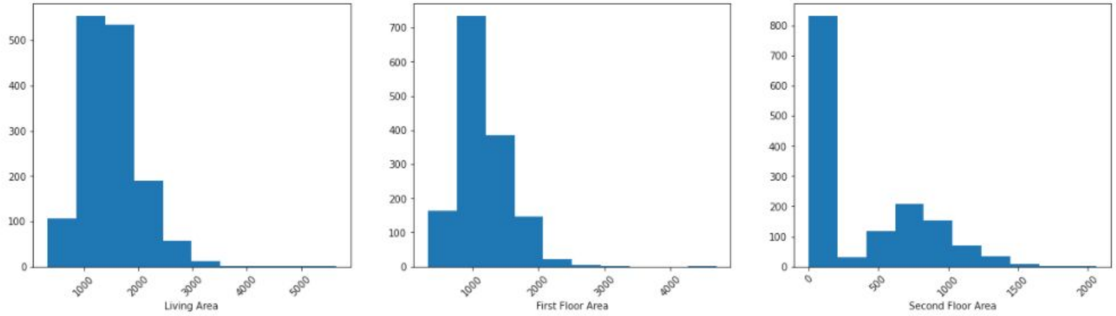
EDA

In [20]:
```
plt.figure(figsize=(20,5))
plt.subplot(131)
plt.hist(train.GrLivArea)
plt.xticks(rotation=45)
plt.xlabel('Living Area')

plt.subplot(132)
plt.hist(train['1stFlrSF'])
plt.xticks(rotation=45)
plt.xlabel('First Floor Area')

plt.subplot(133)
plt.hist(train['2ndFlrSF'])
plt.xticks(rotation=45)
plt.xlabel('Second Floor Area')
```

Out[20]:
```
Text(0.5, 0, 'Second Floor Area')
```



- Most of the houses range from 1000 SqFt to 2000 SqFt of Living Area
- First Floor area range from 300 to 2000 SqFt
- Large number of houses with second floor area have second floor with area in the range of 200 SqFt, with rest of the houses second floor area ranging between 500 to 1500 SqFt

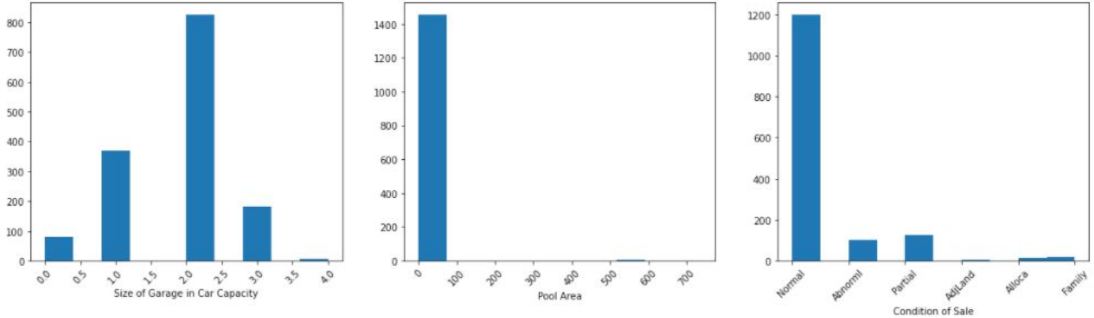
EDA

```
In [21]: plt.figure(figsize=(20,5))
         plt.subplot(131)
         plt.hist(train.GarageCars)
         plt.xticks(rotation=45)
         plt.xlabel('Size of Garage in Car Capacity')

         plt.subplot(132)
         plt.hist(train['PoolArea'])
         plt.xticks(rotation=45)
         plt.xlabel('Pool Area')

         plt.subplot(133)
         plt.hist(train['SaleCondition'])
         plt.xticks(rotation=45)
         plt.xlabel('Condition of Sale')

Out[21]: Text(0.5, 0, 'Condition of Sale')
```



- Most of the houses have garages which can accomodate 2 cars, and rest of the houses of garages which can accommodate just 1 or in some cases 3 cars
- Pool area ranges from 0 to 100 SqFt for most of the houses
- Most of the houses were sold as Normal Condition

Data Story

Data Story

Explore numerical features of Data

In [27]:
```
#Explore Numerical Features
train_numerical = train3.select_dtypes(include = [np.number])
train_numerical
```

Out[27]:

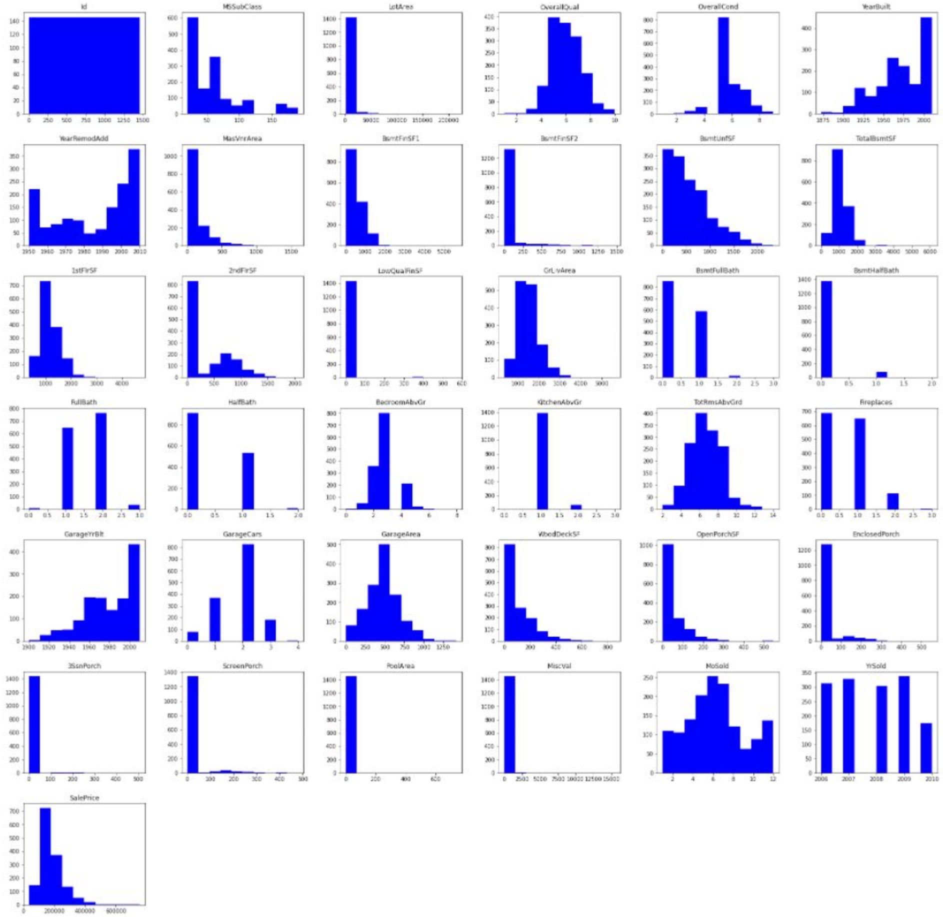
	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeckSF	OpenPorchSF
0	1	60	8450	7	5	2003	2003	196.0	706	0	...	0	61
1	2	20	9600	6	8	1976	1976	0.0	978	0	...	298	0
2	3	60	11250	7	5	2001	2002	162.0	486	0	...	0	42
3	4	70	9550	7	5	1915	1970	0.0	216	0	...	0	35
4	5	60	14260	8	5	2000	2000	350.0	655	0	...	192	84
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60	7917	6	5	1999	2000	0.0	0	0	...	0	40
1456	1457	20	13175	6	6	1978	1988	119.0	790	163	...	349	0
1457	1458	70	9042	7	9	1941	2006	0.0	275	0	...	0	60
1458	1459	20	9717	5	6	1950	1996	0.0	49	1029	...	366	0
1459	1460	20	9937	5	6	1965	1965	0.0	830	290	...	736	68

1460 rows × 37 columns

Data Story

In the above histogram, there are many variables such as PoolArea,ScreenPorch, EnclosedPorch, MiscVal, 3SsnPorch, LowQualFinSF, BsmtFinSF2,and KitchenAbvGr having too many rows being too many same values and reflect extreme outliers. Outliers can affect a regression model by pulling our estimated regression line further away from the true population regression line.

In [28]:
```
train_numerical.hist(figsize = (30,30), grid = False, color ='blue')
plt.show()
```



Data Story

In [29]:
```python
# Let's drop these features from the dataset
train_numerical = train_numerical.drop(columns = ['PoolArea','ScreenPorch',
                 'EnclosedPorch','MiscVal','3SsnPorch','LowQualFinSF',
                'BsmtFinSF2', 'KitchenAbvGr','BsmtHalfBath','LowQualFinSF'])
train_numerical
```

Out[29]:

	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtUnfSF	...	TotRmsAbvGrd	Fireplaces
0	1	60	8450	7	5	2003	2003	196.0	706	150	...	8	0
1	2	20	9600	6	8	1976	1976	0.0	978	284	...	6	1
2	3	60	11250	7	5	2001	2002	162.0	486	434	...	6	1
3	4	70	9550	7	5	1915	1970	0.0	216	540	...	7	1
4	5	60	14260	8	5	2000	2000	350.0	655	490	...	9	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60	7917	6	5	1999	2000	0.0	0	953	...	7	1
1456	1457	20	13175	6	6	1978	1988	119.0	790	589	...	7	2
1457	1458	70	9042	7	9	1941	2006	0.0	275	877	...	9	2
1458	1459	20	9717	5	6	1950	1996	0.0	49	0	...	5	0
1459	1460	20	9937	5	6	1965	1965	0.0	830	136	...	6	0

1460 rows × 28 columns

In [30]:
```python
# We have to also repeat this process for test dataset
test_numerical = test3.select_dtypes(include = [np.number])
test_numerical = test_numerical.drop(columns = ['PoolArea','ScreenPorch',
                 'EnclosedPorch','MiscVal','3SsnPorch','LowQualFinSF',
                'BsmtFinSF2', 'KitchenAbvGr','BsmtHalfBath','LowQualFinSF'])
test_numerical
```

Out[30]:

	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtUnfSF	...	BedroomAbvGr	TotRmsAbv
0	1461	20	11622	5	6	1961	1961	0.0	468.0	270.0	...	2	
1	1462	20	14267	6	6	1958	1958	108.0	923.0	406.0	...	3	
2	1463	60	13830	5	5	1997	1998	0.0	791.0	137.0	...	3	
3	1464	60	9978	6	6	1998	1998	20.0	602.0	324.0	...	3	
4	1465	120	5005	8	5	1992	1992	0.0	263.0	1017.0	...	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1454	2915	160	1936	4	7	1970	1970	0.0	0.0	546.0	...	3	
1455	2916	160	1894	4	5	1970	1970	0.0	252.0	294.0	...	3	
1456	2917	20	20000	5	7	1960	1996	0.0	1224.0	0.0	...	4	
1457	2918	85	10441	5	5	1992	1992	0.0	337.0	575.0	...	3	
1458	2919	60	9627	7	5	1993	1994	94.0	758.0	238.0	...	3	

1459 rows × 27 columns

Assessment of Correlation between numerical features and target (Sale Price)

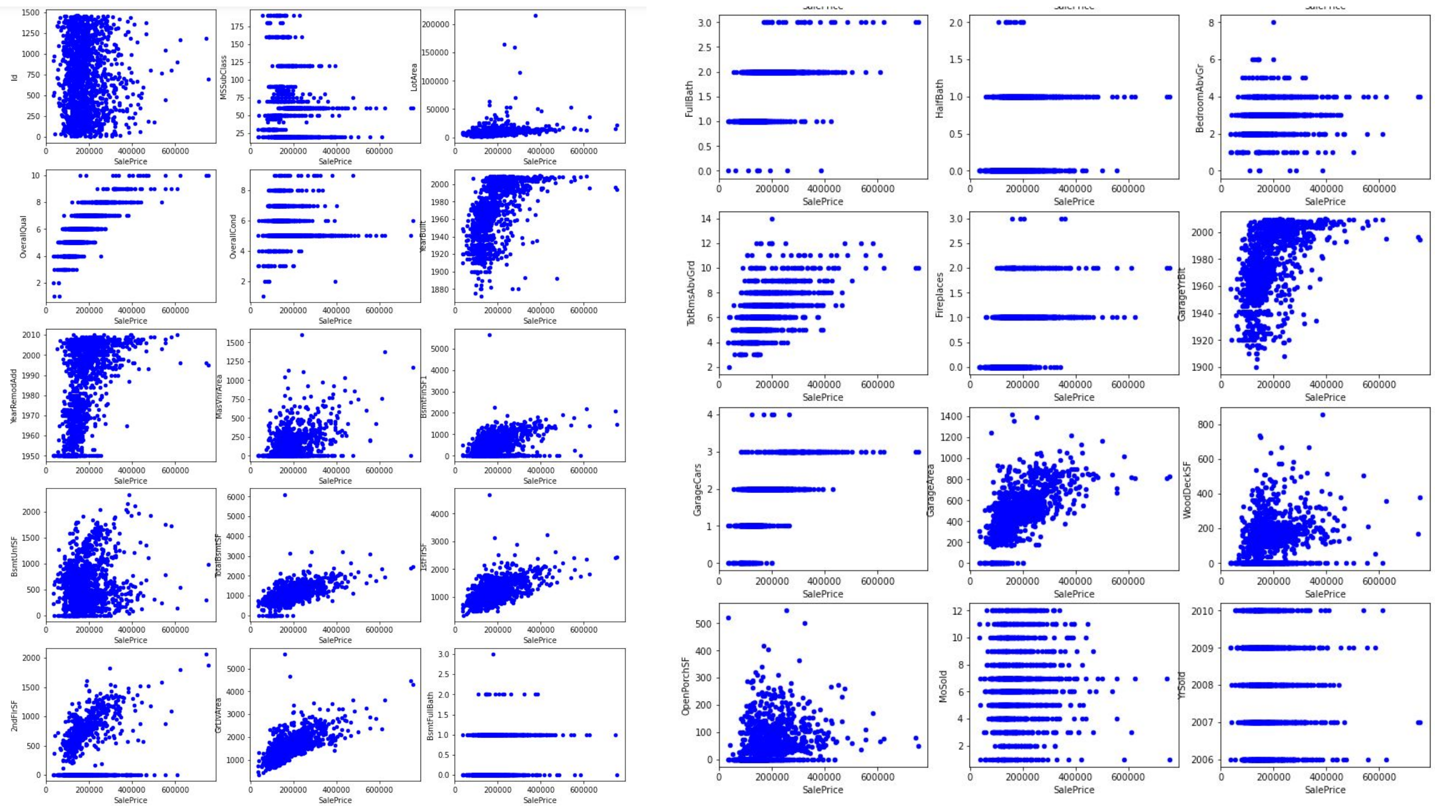
The features having a correlation coefficient of equal to or more than 0.5, can be safely assumed to be highly correlated with the Sale Price

```
In [31]: correlation = train_numerical.corr()['SalePrice'].sort_values().dropna()
         correlation

Out[31]: MSSubClass     -0.084284
         OverallCond    -0.077856
         YrSold         -0.028923
         Id             -0.021917
         MoSold          0.046432
         BedroomAbvGr    0.168213
         BsmtUnfSF       0.214479
         BsmtFullBath    0.227122
         LotArea         0.263843
         HalfBath        0.284108
         OpenPorchSF     0.315856
         2ndFlrSF        0.319334
         WoodDeckSF      0.324413
         BsmtFinSF1      0.386420
         Fireplaces      0.466929
         MasVnrArea      0.477493
         GarageYrBlt     0.486362
         YearRemodAdd    0.507101
         YearBuilt       0.522897
         TotRmsAbvGrd    0.533723
         FullBath        0.560664
         1stFlrSF        0.605852
         TotalBsmtSF     0.613581
         GarageArea      0.623431
         GarageCars      0.640409
         GrLivArea       0.708624
         OverallQual     0.790982
         SalePrice       1.000000
         Name: SalePrice, dtype: float64
```

Data Story

In [72]:
```
fig, axes = plt.subplots(9, 3, figsize=(14, 35))
axe = axes.ravel()
for i, col in enumerate(train_numerical.columns.values[:-1]):
    train_numerical.plot(x=('SalePrice'),y=(col),ax=axe[i], kind = 'scatter', color='blue')
plt.show()
```



Now let's visually verify the correlation pattern among various features using scatter plots.

Data Story

Following features seem to have a linear relationship with the Sale Price, while the rest of the features do not exhibit a linear relationship with Sale Price and hence do not seem to have much influence on the sale price. Lot Area: Lot size in square feet

1. **Lot Area**: Lot size in square feet
2. **Year Built**: Original construction date
3. **YearRemoAdd**: Remodel date
4. **MasVnrArea**: Masonry veneer area in square feet
5. **BsmtFinSF1**: Type 1 finished square feet
6. **BsmtUnfSF**: Unfinished square feet of basement area
7. **TotalBsmtSF**: Total square feet of basement area
8. **1stFlrSF**: First Floor square feet
9. **2ndFlrSF**: Second floor square feet
10. **GrLivArea**: Above grade (ground) living area square feet
11. **GarageArea**: Size of garage in square feet
12. **WoodDeckSF**: Wood deck area in square feet
13. **OpenPorchSF**: Open porch area in square feet

Data Story

- Let’s identify features which have too many rows. We may need to remove these features if most of the rows have jusy one or two values attributes associated with them.
- Essentially it is assumed that these attributes do not have much role in influencing the sale price of the house.
- Following features are identified

In [34]:
```
num_rows = len(train_categorical.index)
low_information_cols = []

for col in train_categorical.columns:
    cnts = train_categorical[col].value_counts(dropna=False)
    top_pct = (cnts/num_rows).iloc[0]

    if top_pct > 0.85:
        low_information_cols.append(col)
        print('{0}: {1:.5f}%'.format(col, top_pct*100))
        print(cnts)
        print()
```

- 'Street','LandContour','Utilities','LandSlope','Condition1','Condition2','RoofMatl', 'ExterCond','BsmtCond','BsmtFinType2','Heating','CentralAir', 'Electrical','Functional', 'GarageQual','GarageCond','PavedDrive', 'SaleType'

Data Story

In [35]:
```
train_categorical = train_categorical.drop(columns = ['Street','LandContour',
              'Utilities','LandSlope','Condition1','Condition2','RoofMatl',
              'ExterCond','BsmtCond','BsmtFinType2','Heating','CentralAir',
              'Electrical','Functional', 'GarageQual','GarageCond','PavedDrive',
              'SaleType'])
train_categorical
```

Out[35]:

	MSZoning	LotShape	LotConfig	Neighborhood	BldgType	HouseStyle	RoofStyle	Exterior1st	Exterior2nd	MasVnrType	ExterQual	Foundation	BsmtQua
0	RL	Reg	Inside	CollgCr	1Fam	2Story	Gable	VinylSd	VinylSd	BrkFace	Gd	PConc	G
1	RL	Reg	FR2	Veenker	1Fam	1Story	Gable	MetalSd	MetalSd	None	TA	CBlock	G
2	RL	IR1	Inside	CollgCr	1Fam	2Story	Gable	VinylSd	VinylSd	BrkFace	Gd	PConc	G
3	RL	IR1	Corner	Crawfor	1Fam	2Story	Gable	Wd Sdng	Wd Shng	None	TA	BrkTil	T
4	RL	IR1	FR2	NoRidge	1Fam	2Story	Gable	VinylSd	VinylSd	BrkFace	Gd	PConc	G
...	...	...	...	...	...	...	...	...	...	...	...	...	..
1455	RL	Reg	Inside	Gilbert	1Fam	2Story	Gable	VinylSd	VinylSd	None	TA	PConc	G
1456	RL	Reg	Inside	NWAmes	1Fam	1Story	Gable	Plywood	Plywood	Stone	TA	CBlock	G
1457	RL	Reg	Inside	Crawfor	1Fam	2Story	Gable	CemntBd	CmentBd	None	Ex	Stone	T
1458	RL	Reg	Inside	NAmes	1Fam	1Story	Hip	MetalSd	MetalSd	None	TA	CBlock	T
1459	RL	Reg	Inside	Edwards	1Fam	1Story	Gable	HdBoard	HdBoard	None	Gd	CBlock	T

1460 rows × 20 columns

Data Story

In [36]:
```
#Let's repeat this extercise for Test dataset
test_categorical = test3.select_dtypes(exclude = [np.number])
test_categorical
```

Out[36]:

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	...	Electrical	KitchenQual	Functional
0	RH	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Feedr	Norm	...	SBrkr	TA	Typ
1	RL	Pave	IR1	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	...	SBrkr	Gd	Typ
2	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	...	SBrkr	TA	Typ
3	RL	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	...	SBrkr	Gd	Typ
4	RL	Pave	IR1	HLS	AllPub	Inside	Gtl	StoneBr	Norm	Norm	...	SBrkr	Gd	Typ
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1454	RM	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	Norm	Norm	...	SBrkr	TA	Typ
1455	RM	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	Norm	Norm	...	SBrkr	TA	Typ
1456	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	Norm	Norm	...	SBrkr	TA	Typ
1457	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	Norm	Norm	...	SBrkr	TA	Typ
1458	RL	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	Norm	Norm	...	SBrkr	TA	Typ

1459 rows × 38 columns

Data Story

In [37]:
```
test_categorical = test_categorical.drop(columns = ['Street','LandContour',
            'Utilities','LandSlope','Condition1','Condition2','RoofMatl',
            'ExterCond','BsmtCond','BsmtFinType2','Heating','CentralAir',
            'Electrical','Functional', 'GarageQual','GarageCond','PavedDrive',
            'SaleType'])
test_categorical
```

Out[37]:

	MSZoning	LotShape	LotConfig	Neighborhood	BldgType	HouseStyle	RoofStyle	Exterior1st	Exterior2nd	MasVnrType	ExterQual	Foundation	BsmtQua
0	RH	Reg	Inside	NAmes	1Fam	1Story	Gable	VinylSd	VinylSd	None	TA	CBlock	T/
1	RL	IR1	Corner	NAmes	1Fam	1Story	Hip	Wd Sdng	Wd Sdng	BrkFace	TA	CBlock	T/
2	RL	IR1	Inside	Gilbert	1Fam	2Story	Gable	VinylSd	VinylSd	None	TA	PConc	G(
3	RL	IR1	Inside	Gilbert	1Fam	2Story	Gable	VinylSd	VinylSd	BrkFace	TA	PConc	T/
4	RL	IR1	Inside	StoneBr	TwnhsE	1Story	Gable	HdBoard	HdBoard	None	Gd	PConc	G(
...	...	...	...	...	...	...	...	...	...	...	...	...	..
1454	RM	Reg	Inside	MeadowV	Twnhs	2Story	Gable	CemntBd	CmentBd	None	TA	CBlock	T/
1455	RM	Reg	Inside	MeadowV	TwnhsE	2Story	Gable	CemntBd	CmentBd	None	TA	CBlock	T/
1456	RL	Reg	Inside	Mitchel	1Fam	1Story	Gable	VinylSd	VinylSd	None	TA	CBlock	T/
1457	RL	Reg	Inside	Mitchel	1Fam	SFoyer	Gable	HdBoard	Wd Shng	None	TA	PConc	G(
1458	RL	Reg	Inside	Mitchel	1Fam	2Story	Gable	HdBoard	HdBoard	BrkFace	TA	PConc	G(

1459 rows × 20 columns

In-Depth Analysis Using ML Techniques

In-Depth Analysis Using ML Techniques

Please refer to the iPython Notebook for following steps and data transformations

- Step 1: Log Transformation of numeric variables
- Step 2: Bringing Train and Test datasets in alignment for the number of unique values for each of the features
- Step 3: Applying One Hot Encoder to the dataset to normalize the data
- Step 4: Conatinating Train and Test Data

In-Depth Analysis Using ML Techniques

Step 6: Let's split consolidated data in to Training and Testing sets

```
In [49]: train5 = Consolidated1.iloc[:1459, :]
         test5 = Consolidated1.iloc[1459:, :]
         print(train5.shape)
         print(test5.shape)

(1459, 165)
(1459, 165)

In [50]: # dropping sale price column from test set which has null values
         test6 =test5.drop(['SalePrice'], axis = 1)
         test6.shape

Out[50]: (1459, 164)

In [51]: X_train = train5.drop(['SalePrice', 'Id'], axis = 1)
         y_train = np.array(train5['SalePrice']).reshape((-1,1))
         X_test = test6.drop(['Id'], axis = 1)

         print(X_train.shape)
         print(y_train.shape)
         print(X_test.shape)

(1459, 163)
(1459, 1)
(1459, 163)
```

In-Depth Analysis Using ML Techniques

Step 7: We need to use imputer to replace missing/NaN values

```
In [52]: imputer = SimpleImputer(missing_values=nan, strategy='median')
         X_train = imputer.fit_transform(X_train)
         print('Missing: %d' % isnan(X_train).sum())

Missing: 0

In [53]: y_train = imputer.fit_transform(y_train)
         print('Missing: %d' % isnan(y_train).sum())

Missing: 0

In [54]: X_test = imputer.fit_transform(X_test)
         print('Missing: %d' % isnan(X_test).sum())

Missing: 0

In [55]: # Make sure all values are finite
         print(np.where(~np.isfinite(X_train)))
         print(np.where(~np.isfinite(y_train)))
         print(np.where(~np.isfinite(X_test)))

(array([], dtype=int64), array([], dtype=int64))
(array([], dtype=int64), array([], dtype=int64))
(array([], dtype=int64), array([], dtype=int64))
```

In-Depth Analysis Using ML Techniques

Step 8: Let's Evaluate a Few Models

We will compare five different machine learning models using the great Scikit-Learn library:

1. Support Vector Machine Regression
2. Random Forest Regression
3. Gradient Boosting Regression
4. K-Nearest Neighbors Regression
5. Boosting Regressor

```
In [56]: def RMSE(y_train, y_pred):
             return mean_squared_error(y_train, y_pred,squared = False)

In [57]: def fit_and_evaluate (model):
             model.fit(X_train, y_train.ravel())
             model_pred = model.predict(X_test)
             model_RMSE = RMSE(y_train, model_pred)

             return model_RMSE

In [58]: # Support Vector Regressor
         svr = SVR(C=1000, gamma = 0.1)
         svr_RMSE = fit_and_evaluate(svr)
         svr_RMSE

Out[58]: 0.47132559184049727

In [59]: # Gradient Boosting Regressor
         gradient_boosted = GradientBoostingRegressor(learning_rate = 0.1,random_state=60)
         gradient_boosted_RMSE = fit_and_evaluate(gradient_boosted)
         gradient_boosted_RMSE

Out[59]: 0.5504183214580101

In [60]: # Bagging Regressor
         bagging = BaggingRegressor()
         bagging_RMSE = fit_and_evaluate(bagging)
         bagging_RMSE

Out[60]: 0.5418501200330114

In [61]: # Random Forest Regressor
         random_forest = RandomForestRegressor(random_state=60)
         random_forest_RMSE = fit_and_evaluate(random_forest)
         random_forest_RMSE

Out[61]: 0.5404380271000233

In [62]: # KNeighbors Regressor
         knn = KNeighborsRegressor(n_neighbors=10)
         knn_RMSE = fit_and_evaluate(knn)
         knn_RMSE

Out[62]: 0.5103854276512614
```

In-Depth Analysis Using ML Techniques

Step 9: Since SVR method has the lowest RMSE we will go ahead with SVR method for further analysis

```
In [63]: SVM = SVR().fit(X_train, y_train.ravel())
         score = SVM.score(X_train, y_train)
         print('R_squared:', score)
         print('RMSE:', svr_RMSE)

         R_squared: 0.8873301230904915
         RMSE: 0.47132559184049727

In [64]: HousePrice_Prediction_with_log = SVM.predict(X_test)
         HousePrice_Prediction_with_log

Out[64]: array([11.67331478, 12.02302764, 12.10646058, ..., 11.96611778,
                11.65541846, 12.27776084])

In [65]: # Converting log values back to normal values
         HousePrice_Prediction_without_log = np.exp(HousePrice_Prediction_with_log)
         HousePrice_Prediction_without_log

Out[65]: array([117396.78215015, 166546.13492393, 181037.69987283, ...,
                157332.67375508, 115314.50045709, 214864.06293624])
```

In-Depth Analysis Using ML Techniques

Step 10: Create Data Frame for Results

In [66]:
```
#Create a Dataframe for the results
results = pd.DataFrame()
results['Id'] = test.Id
results['SalePrice'] = HousePrice_Prediction_without_log
results
```

Out[66]:

	Id	SalePrice
0	1461	117396.782150
1	1462	166546.134924
2	1463	181037.699873
3	1464	200325.377181
4	1465	186982.473085
...	...	...
1454	2915	85261.463708
1455	2916	87854.190275
1456	2917	157332.673755
1457	2918	115314.500457
1458	2919	214864.062936

1459 rows × 2 columns

Closing Thoughts, Conclusions

Closing Thoughts, Conclusions

- Out of the various models used, the Support Vector Model Regression method turned out to be the most effective method for predicting price. Although, it is difficult to determine why it was giving better results than other methods. Also, it is possible that for other datasets, this may not hold and some other model could turn out to be more effective. That is why it is important to evaluate many models to determine which model works the best.
- Key Findings Related to Data Set & Features
    - Although the data set had a lot of attributes, the attributes which had real/significant impact are just a few.
    - Following are the key attributes which have impact on Sale Price
        - Lot Area, Year Built, YearRemoAdd, MasVnrArea, BsmtFinSF1, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, GrLivArea, GarageArea, WoodDeckSF, OpenPorchSF

References

References

Ames Housing Dataset: http://www.amstat.org/publications/jse/v19n3/decock.pdf

Kaggle Link: https://www.kaggle.com/c/house-prices-advanced-regression-techniques

Thank You