HOUSING PRICE PREDICTION

Submitted by-:
NITISH KUMAR SHARMA

ACKNOWLEDGEMENT

It is great pleasure for me to undertake this project. I feel overwhelmed doing this project entitled – “Housing Price Prediction“.

Some of the resources that helped me to complete this project are as follows:

- Internet/web
- Stack overflow
- https://www.homestratosphere.com/types-exterior-siding-home
- Analytics Vidhya
- Coding Ninjas
- Articles published in Medium.com
- Wikipedia

INTRODUCTION

BUSINESS PROBLEM FRAMING

This project includes the real time problem for US-based housing company named Surprise Housing that has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price.

This industry makes money by developing a property for reselling at a higher charge. Prediction house prices are expected to help people who plan to buy a house so that they can know the price range in the future, then they can plan their finance well. In addition, house price predictions are also beneficial for property investors to know the trend of housing prices in a certain location.

Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

Housing sales price are determined by numerous factors such as area of the property, location of the house, material used for construction, age of the property, number of bedrooms and garages and so on.
Generally the property values rise with respect to time and its appraised value need to be calculated.

Now, Property prices depend on various parameters in the economy and society. However, previous analyses show that house prices are strongly dependent on the size of the house and its geographical location .We have also considered various intrinsic parameters (such as number of bedrooms, living area and construction material) and also external parameters (such as location, proximity, upcoming projects, etc.

Houses have a variant number of features that may not have the same cost due to its location. For instance, a big house may have a higher price if it is located in desirable rich area than being placed in a poor neighbourhood.

REVIEW OF LITERATURE

The real estate market is one of the most competitive in terms of pricing and the same tends to vary significantly based on a lot of factors, hence it becomes one of the prime fields to apply the concepts of machine learning to optimize and predict the prices with high accuracy.

The client has collected dataset from the sale of houses in Australia .The company is looking at prospective properties to buy houses to enter the market. The client wants predictions for the actual value of the prospective properties and decide whether to invest in them or not.

We have used different machine learning models to predict the above. Since we have continous target data so regression model algorithms has been used.

We will start our project with the train and test dataset which contains Saleprice as target variable along with other independent variables. We will observe all the variables with following goals in mind:

- Relevance of the variables
- Distribution of the variables
- Data Cleaning of the variables
- Visualization of the variables
- Visualization of the variables as per Saleprice for data analysis

After having gone through all the variables and cleaning the dataset, we will move on to machine learning regression modelling:

- Pre-processing of the dataset for models
- Testing multiple algorithms with multiple evaluation metrics
- Select evaluation metric as per our specific business application
- Doing hyper-parameter tuning using GridSearchCV for the best model
- Finally saving the best model
- Loading and predicting with the loaded model

MOTIVATION FOR THE PROBLEM UNDERTAKEN

This project deals in real estate business which is a hot market. Here a company is investing in billions to get profitable returns. So I am very much excited and thrilled to deal with real time data provided by the client so that through my analysis and observation I could predict whether to invest or not in certain properties. Hence solving their business problem.

With more and more analysis on the data, it will improve my knowledge about this domain. It will be very productive for me to know how the features/variables present in the data are affecting the target variable, so that we can evaluate price of houses exact based on the information provided.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL / ANALYTICAL MODELING OF THE PROBLEM

The dataset contains two csv files (Train & Test dataset). Train data has 81 attributes (80 variables and 1 target variable). The target (Saleprice) variable is a continous data. Thus it's a regression problem. The other key variables are MSSubClass (type of dwelling), Neighborhood, OverallQual (overall quality of house), garage, RoofMatl (Roof material), Foundation, Exterior1st (exterior covering on house),saletype, 1stfloorSF, basement, sale condition, etc.

To know the overview/stats of the dataset , we will be using df.describe() function which gives informations like count, min, max, mean, standard deviation values of features . By plotting of heat map we can correlate features if they are highly inter-correlated or not. So we can drop columns if problem of multicollinearity arises (if vif >10) when we observe through variance inflation factor.

From an initial statistical overview of the dataset, we infer that most variables have standard deviation value greater than mean value which shows that the data is messed up.

DATA SOURCES AND THEIR FORMATS

The train and test data that I received was in CSV format (Comma Separated Values). After reading the train data by using function df=pd.read_csv (‘ ---file path --- ’) in jupyter notebook there were 1168 rows and 81 columns. For Test data there were 292 rows and 80 columns.

Train dataset datatypes are as follow :

```
df1.info() # to know datatype of each columns in train data
```

```
<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
 32  BsmtExposure   1137 non-null   object
 33  BsmtFinType1   1138 non-null   object
 34  BsmtFinSF1     1168 non-null   int64
 35  BsmtFinType2   1137 non-null   object
 36  BsmtFinSF2     1168 non-null   int64
 37  BsmtUnfSF      1168 non-null   int64
 38  TotalBsmtSF    1168 non-null   int64
 39  Heating        1168 non-null   object
 40  HeatingQC      1168 non-null   object
 41  CentralAir     1168 non-null   object
 42  Electrical     1168 non-null   object
 43  1stFlrSF       1168 non-null   int64
 44  2ndFlrSF       1168 non-null   int64
 45  LowQualFinSF   1168 non-null   int64
 46  GrLivArea      1168 non-null   int64
 47  BsmtFullBath   1168 non-null   int64
 48  BsmtHalfBath   1168 non-null   int64
 49  FullBath       1168 non-null   int64
 50  HalfBath       1168 non-null   int64
 51  BedroomAbvGr   1168 non-null   int64
 52  KitchenAbvGr   1168 non-null   int64
 53  KitchenQual    1168 non-null   object
 54  TotRmsAbvGrd   1168 non-null   int64
 55  Functional     1168 non-null   object
 56  Fireplaces     1168 non-null   int64
 57  FireplaceQu    617 non-null    object
 58  GarageType     1104 non-null   object
 59  GarageYrBlt    1104 non-null   float64
 60  GarageFinish   1104 non-null   object
 61  GarageCars     1168 non-null   int64
 62  GarageArea     1168 non-null   int64
 63  GarageQual     1104 non-null   object
 64  GarageCond     1104 non-null   object
 65  PavedDrive     1168 non-null   object
 66  WoodDeckSF     1168 non-null   int64
 67  OpenPorchSF    1168 non-null   int64
 68  EnclosedPorch  1168 non-null   int64
 69  3SsnPorch      1168 non-null   int64
 70  ScreenPorch    1168 non-null   int64
 71  PoolArea       1168 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          237 non-null    object
 74  MiscFeature    44 non-null     object
 75  MiscVal        1168 non-null   int64
 76  MoSold         1168 non-null   int64
 77  YrSold         1168 non-null   int64
 78  SaleType       1168 non-null   object
 79  SaleCondition  1168 non-null   object
 80  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 739.2+ KB
```

Test dataset datatypes are as follow :

```
df2.info() # to know datatype of each columns in test data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292 entries, 0 to 291
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             292 non-null    int64
 1   MSSubClass     292 non-null    int64
 2   MSZoning       292 non-null    object
 3   LotFrontage    247 non-null    float64
 4   LotArea        292 non-null    int64
 5   Street         292 non-null    object
 6   Alley          14 non-null     object
 7   LotShape       292 non-null    object
 8   LandContour    292 non-null    object
 9   Utilities      292 non-null    object
 10  LotConfig      292 non-null    object
 11  LandSlope      292 non-null    object
 12  Neighborhood   292 non-null    object
 13  Condition1     292 non-null    object
 14  Condition2     292 non-null    object
 15  BldgType       292 non-null    object
 16  HouseStyle     292 non-null    object
 17  OverallQual    292 non-null    int64
 18  OverallCond    292 non-null    int64
 19  YearBuilt      292 non-null    int64
 20  YearRemodAdd   292 non-null    int64
 21  RoofStyle      292 non-null    object
 22  RoofMatl       292 non-null    object
 23  Exterior1st    292 non-null    object
 24  Exterior2nd    292 non-null    object
 25  MasVnrType     291 non-null    object
 26  MasVnrArea     291 non-null    float64
 27  ExterQual      292 non-null    object
 28  ExterCond      292 non-null    object
 29  Foundation     292 non-null    object
 30  BsmtQual       285 non-null    object
 31  BsmtCond       285 non-null    object
 32  BsmtExposure   285 non-null    object
 33  BsmtFinType1   285 non-null    object
 34  BsmtFinSF1     292 non-null    int64
 35  BsmtFinType2   285 non-null    object
 36  BsmtFinSF2     292 non-null    int64
 37  BsmtUnfSF      292 non-null    int64
 38  TotalBsmtSF    292 non-null    int64
 39  Heating        292 non-null    object
 40  HeatingQC      292 non-null    object
 41  CentralAir     292 non-null    object
 42  Electrical     291 non-null    object
 43  1stFlrSF       292 non-null    int64
 44  2ndFlrSF       292 non-null    int64
 45  LowQualFinSF   292 non-null    int64
 46  GrLivArea      292 non-null    int64
 47  BsmtFullBath   292 non-null    int64
 48  BsmtHalfBath   292 non-null    int64
 49  FullBath       292 non-null    int64
 50  HalfBath       292 non-null    int64
 51  BedroomAbvGr   292 non-null    int64
 52  KitchenAbvGr   292 non-null    int64
 53  KitchenQual    292 non-null    object
 54  TotRmsAbvGrd   292 non-null    int64
 55  Functional     292 non-null    object
 56  Fireplaces     292 non-null    int64
 57  FireplaceQu    153 non-null    object
 58  GarageType     275 non-null    object
 59  GarageYrBlt    275 non-null    float64
 60  GarageFinish   275 non-null    object
 61  GarageCars     292 non-null    int64
 62  GarageArea     292 non-null    int64
 63  GarageQual     275 non-null    object
 64  GarageCond     275 non-null    object
 65  PavedDrive     292 non-null    object
 66  WoodDeckSF     292 non-null    int64
 67  OpenPorchSF    292 non-null    int64
 68  EnclosedPorch  292 non-null    int64
 69  3SsnPorch      292 non-null    int64
 70  ScreenPorch    292 non-null    int64
 71  PoolArea       292 non-null    int64
 72  PoolQC         0 non-null      float64
 73  Fence          44 non-null     object
 74  MiscFeature    10 non-null     object
 75  MiscVal        292 non-null    int64
 76  MoSold         292 non-null    int64
 77  YrSold         292 non-null    int64
 78  SaleType       292 non-null    object
 79  SaleCondition  292 non-null    object
dtypes: float64(4), int64(34), object(42)
memory usage: 182.6+ KB
```

Categorical & Continous features in the train dataset are as follows :

```
# to list out categorical features from train dataset
cat_features=[i for i in df1.columns if df1.dtypes[i]=='object']
cat_features
```

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

```
# listing out continous features from train data
con_features1=[i for i in df1.columns if df1.dtypes[i]=='int64' or df1.dtypes[i]=='float64']
con_features1
```

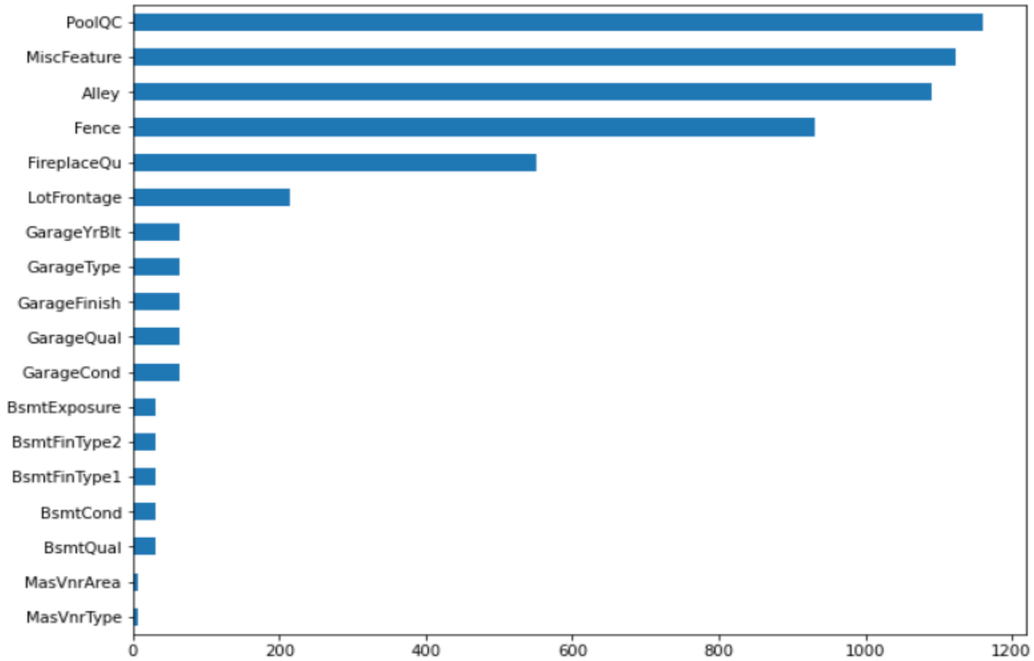
```
['MSSubClass',
 'LotFrontage',
 'LotArea',
 'OverallQual',
 'OverallCond',
 'YearBuilt',
 'YearRemodAdd',
 'MasVnrArea',
 'BsmtFinSF1',
 'BsmtFinSF2',
 'BsmtUnfSF',
 'TotalBsmtSF',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
 'GrLivArea',
 'BsmtFullBath',
 'BsmtHalfBath',
 'FullBath',
 'HalfBath',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

DATA PREPROCESSING DONE

- Variables from train and test data containing null values only was plotted horizontally to be observed

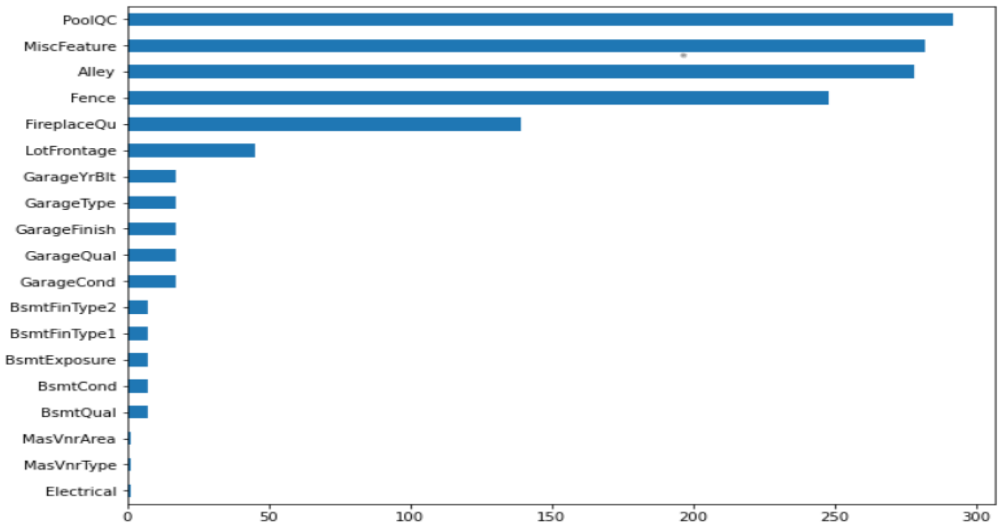
```
# to visualize variables having missing values from train data
fig, ax = plt.subplots(figsize=(10,8))
null = df1.isnull().sum()
null = null[null > 0]
null.sort_values(inplace=True)
null.plot.barh(ax=ax)
```

```
<AxesSubplot:>
```




```
# to visualize features having null values from test data
fig, ax = plt.subplots(figsize=(10,8))
null = df2.isnull().sum()
null = null[null > 0]
null.sort_values(inplace=True)
null.plot.barh(ax=ax)
```

```
<AxesSubplot:>
```



- Lots of variables containing null values from both train & test data.

```python
# dropping variables having maximum % of null values from train & test data
df1=df1.drop(columns=['PoolQC','Fence','MiscFeature','Alley'])
df2=df2.drop(columns=['PoolQC','Fence','MiscFeature','Alley'])
```

```python
# Dropping 'GarageQual' column : Garage Quality & Garage Condition column represents same thing
df1=df1.drop(columns='GarageQual')
df2=df2.drop(columns='GarageQual')
```

```python
# dropping columns as they are not useful for prediction
df1=df1.drop(columns=['Id','MiscVal','KitchenAbvGr','FireplaceQu'])
df2=df2.drop(columns=['Id','MiscVal','KitchenAbvGr','FireplaceQu'])
```

- Variables having null values between (75-100) % were removed.
- Out of two variables, one was removed that represented same thing.
- Variables containing numbers or identifiers were removed which were not useful.

```python
# dropping feature
df1=df1.drop(columns='Utilities')
df2=df2.drop(columns='Utilities')
```

```python
df1=df1.drop(columns=['Exterior2nd','BsmtFinType2'])
df2=df2.drop(columns=['Exterior2nd','BsmtFinType2'])
```

```python
# dropping 'Condition2' column : categories are same like 'condition1' column and also has less categorical values
df1=df1.drop(columns='Condition2')
df2=df2.drop(columns='Condition2')
```

```python
# treating null values for train data : Using mode for categorical variables and median for continous variables
df1['BsmtCond']=df1['BsmtCond'].fillna(df1['BsmtCond'].mode()[0])
df1['BsmtQual']=df1['BsmtQual'].fillna(df1['BsmtQual'].mode()[0])
df1['BsmtExposure']=df1['BsmtExposure'].fillna(df1['BsmtExposure'].mode()[0])
df1['BsmtFinType1']=df1['BsmtFinType1'].fillna(df1['BsmtFinType1'].mode()[0])
df1['GarageCond']=df1['GarageCond'].fillna(df1['GarageCond'].mode()[0])
df1['GarageFinish']=df1['GarageFinish'].fillna(df1['GarageFinish'].mode()[0])
df1['GarageType']=df1['GarageType'].fillna(df1['GarageType'].mode()[0])
df1['MasVnrType']=df1['MasVnrType'].fillna(df1['MasVnrType'].mode()[0])

df1['LotFrontage']=df1['LotFrontage'].fillna(df1['LotFrontage'].median())
df1['GarageYrBlt']=df1['GarageYrBlt'].fillna(df1['GarageYrBlt'].median())
df1['MasVnrArea']=df1['MasVnrArea'].fillna(df1['MasVnrArea'].median())
```

```python
# treating null values for test data : Using mode for categorical variables and median for continous variables
df2['BsmtCond']=df2['BsmtCond'].fillna(df2['BsmtCond'].mode()[0])
df2['BsmtQual']=df2['BsmtQual'].fillna(df2['BsmtQual'].mode()[0])
df2['BsmtExposure']=df2['BsmtExposure'].fillna(df2['BsmtExposure'].mode()[0])
df2['BsmtFinType1']=df2['BsmtFinType1'].fillna(df2['BsmtFinType1'].mode()[0])
df2['GarageCond']=df2['GarageCond'].fillna(df2['GarageCond'].mode()[0])
df2['GarageFinish']=df2['GarageFinish'].fillna(df2['GarageFinish'].mode()[0])
df2['GarageType']=df2['GarageType'].fillna(df2['GarageType'].mode()[0])
df2['MasVnrType']=df2['MasVnrType'].fillna(df2['MasVnrType'].mode()[0])
df2['Electrical']=df2['Electrical'].fillna(df2['Electrical'].mode()[0])

df2['LotFrontage']=df2['LotFrontage'].fillna(df2['LotFrontage'].median())
df2['GarageYrBlt']=df2['GarageYrBlt'].fillna(df2['GarageYrBlt'].median())
df2['MasVnrArea']=df2['MasVnrArea'].fillna(df2['MasVnrArea'].median())
```

- Filling null values with median and mode whose missing values were of acceptable range.

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

When visualizing the feature variables with the target (Saleprice) variable, It gave me insights that how these variables are so important for making houses price predictions.

Lot of variables given here explains that how price increases/decreases according to the location & density of population. Which type of foundation and exterior covering done has maximum or minimum Saleprice. What type of dwelling is now in trend with its expected price range.

With increase in overall material and finish of the house (OverallQual), the saleprice increases. If the built year of properties/houses is less then saleprice of houses will be high.

STATE THE SET OF ASSUMPTIONS (IF ANY) RELATED TO THE PROBLEM UNDER CONSIDERATION

```
df1.describe().T # to get high understanding of dataset or to get overview/stats of the train dataset
```

	count	mean	std	min	25%	50%	75%	max
Id	1168.0	724.136130	416.159877	1.0	360.50	714.5	1079.5	1460.0
MSSubClass	1168.0	56.767979	41.940650	20.0	20.00	50.0	70.0	190.0
LotFrontage	954.0	70.988470	24.828750	21.0	60.00	70.0	80.0	313.0
LotArea	1168.0	10484.749144	8957.442311	1300.0	7621.50	9522.5	11515.5	164660.0
OverallQual	1168.0	6.104452	1.390153	1.0	5.00	6.0	7.0	10.0
OverallCond	1168.0	5.595890	1.124343	1.0	5.00	5.0	6.0	9.0
YearBuilt	1168.0	1970.930651	30.145255	1875.0	1954.00	1972.0	2000.0	2010.0
YearRemodAdd	1168.0	1984.758562	20.785185	1950.0	1966.00	1993.0	2004.0	2010.0
MasVnrArea	1161.0	102.310078	182.595606	0.0	0.00	0.0	160.0	1600.0
BsmtFinSF1	1168.0	444.726027	462.664785	0.0	0.00	385.5	714.5	5644.0
BsmtFinSF2	1168.0	46.647260	163.520016	0.0	0.00	0.0	0.0	1474.0
BsmtUnfSF	1168.0	569.721747	449.375525	0.0	216.00	474.0	816.0	2336.0
TotalBsmtSF	1168.0	1061.095034	442.272249	0.0	799.00	1005.5	1291.5	6110.0
1stFlrSF	1168.0	1169.860445	391.161983	334.0	892.00	1096.5	1392.0	4692.0
2ndFlrSF	1168.0	348.826199	439.696370	0.0	0.00	0.0	729.0	2065.0
LowQualFinSF	1168.0	6.380137	50.892844	0.0	0.00	0.0	0.0	572.0
GrLivArea	1168.0	1525.066781	528.042957	334.0	1143.25	1468.5	1795.0	5642.0
BsmtFullBath	1168.0	0.425514	0.521615	0.0	0.00	0.0	1.0	3.0
BsmtHalfBath	1168.0	0.055651	0.236699	0.0	0.00	0.0	0.0	2.0
FullBath	1168.0	1.562500	0.551882	0.0	1.00	2.0	2.0	3.0
HalfBath	1168.0	0.388699	0.504929	0.0	0.00	0.0	1.0	2.0
BedroomAbvGr	1168.0	2.884418	0.817229	0.0	2.00	3.0	3.0	8.0
KitchenAbvGr	1168.0	1.045377	0.216292	0.0	1.00	1.0	1.0	3.0
TotRmsAbvGrd	1168.0	6.542808	1.598484	2.0	5.00	6.0	7.0	14.0
Fireplaces	1168.0	0.617295	0.650575	0.0	0.00	1.0	1.0	3.0
GarageYrBlt	1104.0	1978.193841	24.890704	1900.0	1961.00	1980.0	2002.0	2010.0
GarageCars	1168.0	1.776541	0.745554	0.0	1.00	2.0	2.0	4.0
GarageArea	1168.0	476.860445	214.466769	0.0	338.00	480.0	576.0	1418.0
WoodDeckSF	1168.0	96.206336	126.158988	0.0	0.00	0.0	171.0	857.0
OpenPorchSF	1168.0	46.559932	66.381023	0.0	0.00	24.0	70.0	547.0
EnclosedPorch	1168.0	23.015411	63.191089	0.0	0.00	0.0	0.0	552.0
3SsnPorch	1168.0	3.639555	29.088867	0.0	0.00	0.0	0.0	508.0
ScreenPorch	1168.0	15.051370	55.080816	0.0	0.00	0.0	0.0	480.0
PoolArea	1168.0	3.448630	44.896939	0.0	0.00	0.0	0.0	738.0
MiscVal	1168.0	47.315068	543.264432	0.0	0.00	0.0	0.0	15500.0
MoSold	1168.0	6.344178	2.686352	1.0	5.00	6.0	8.0	12.0
YrSold	1168.0	2007.804795	1.329738	2006.0	2007.00	2008.0	2009.0	2010.0
SalePrice	1168.0	181477.005993	79105.586863	34900.0	130375.00	163995.0	215000.0	755000.0

- From the above statistical summary we can observe that lots of variables data is messed up as their standard deviation value is greater than their mean value
- Count is not same for all variables so there will be missing values in the train data as well as for test data when observed.

HARDWARE / SOFTWARE REQUIREMENTS AND TOOLS USED

- Anaconda Navigator 1.10.0
- Jupyter Notebook 6.1.4 , Python 3

Libraries

```
import pandas as pd # for handling dataset
import numpy as np  # for mathematical computation

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score

from sklearn.preprocessing import LabelEncoder
from scipy.stats import skew

# for visualization
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import style
import seaborn as sns

# for saving & loading model
import pickle

import warnings
warnings.filterwarnings('ignore')
```

```
# importing libraries for model building
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, plot_roc_curve, roc_auc_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
#Label Encoding for object to numeric conversion for train data
lab_enc = LabelEncoder()
objList = df1.select_dtypes(include = "object").columns
for var in objList:
    df1[var] = lab_enc.fit_transform(df1[var].astype(str))
```

Libraries and Packages used:

- Pickle for saving & loading machine learning model.
- GridSearchCV for Hyper-parameter tuning.
- Cross validation score to cross check if the model is overfitting or not.
- Label Encoder to convert objects into integers for train and test categorical variables.
- Seaborn and matplotlib for visualization.

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

As we have to predict saleprice of houses with variables containing both categorical and continuous values but our target variable was of continuous data. Thus it’s a regression problem and we have to apply regression algorithms and build a model giving good performance.

We have to drop variables which were containing maximum percentage of null values and also which is having one value throughout its column. Removed some duplicate variables that represented same thing, thereby minimizing runtime of system.

Plotted heatmap to visualize which variables were showing good positive or negative correlation with the target variable. So that it could help us in further analysis.

Checked for skewness, outliers and handled them with 1.5 IQR method. Dropped variables that were highly correlated with each other to remove multicollinearity.

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

Algorithms used:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- KNN Regressor
- Gradient Boosting Regressor
- XGB Regressor

1. Logistic Regression

```
# Split data into train and test. Model will be built on training data and tested on test data
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size = 0.25, random_state = 51)
```

```
y_train.head()
```

```
753    181000
19     106000
65     350000
54      84900
221    115000
Name: SalePrice, dtype: int64
```

```
regression = LinearRegression()
regression.fit(x_train,y_train)
```

```
LinearRegression()
```

```
# lets check how well model fits the test data
regression.score(x_test,y_test)
```

Test Result : 85.13%

2. Decision Tree Regressor

```
dt_reg = DecisionTreeRegressor()
dt_reg.fit(x_train,y_train)

print("\n==================Train Result=====================")

print(f"Accuracy Score: {dt_reg.score(x_train,y_train) * 100:.2f}%")
print("_______________________________________________")

#*********************** Test Score ************************

y_pred = dt_reg.predict(x_test)
print("\n==================Test Result=====================")
print(f"Accuracy Score: {dt_reg.score(x_test,y_test) * 100:.2f}%")
print("_______________________________________________")
```

Training Result : 100%   Test Result : 76.42%

3. Random Forest Regressor

```
rand_reg = RandomForestRegressor(n_estimators = 100,random_state=51)
rand_reg.fit(x_train,y_train)

print("\n==================Train Result=====================")

print(f"Accuracy Score: {rand_reg.score(x_train,y_train) * 100:.2f}%")
print("_______________________________________________")

#*********************** Test Score ************************

y_pred = rand_reg.predict(x_test)
print("\n==================Test Result=====================")
print(f"Accuracy Score: {rand_reg.score(x_test,y_test) * 100:.2f}%")
print("_______________________________________________")
```

Training Result : 97.73%   Test Result : 87.08%

4. KNN Regressor

```
knn=KNeighborsRegressor(n_neighbors=5)
knn.fit(x_train,y_train)

print("\n=================Train Result====================")

print(f"Accuracy Score: {knn.score(x_train,y_train) * 100:.2f}%")
print("_______________________________________________")

#*********************** Test Score ************************

y_pred = knn.predict(x_test)
print("\n=================Test Result====================")
print(f"Accuracy Score: {knn.score(x_test,y_test) * 100:.2f}%")
print("_______________________________________________")
```

Training Result : 82.88% Test Result : 74.15%

5. Gradient Boosting Regressor

```
from sklearn.ensemble import GradientBoostingRegressor
np.random.seed(42)

gbr = GradientBoostingRegressor()
gbr.fit(x_train, y_train)

y_preds = gbr.predict(x_test)
y_preds

gbr.score(x_test, y_test)
```

Test Result : 88.72%

6. XGB Regressor

```
from xgboost.sklearn import XGBRegressor
np.random.seed(42)
xgb = XGBRegressor()
xgb.fit(x_train, y_train)
predictions = xgb.predict(x_test)

xgb.score(x_test,y_test)
```

Test Result : 87.04%

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

Precision: It is the ratio between the True Positives and all the Positives. It can be seen as a measure of quality, higher precision means that an algorithm returns more relevant results than irrelevant ones

Recall : It is the measure of our model correctly identifying True Positives. It is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

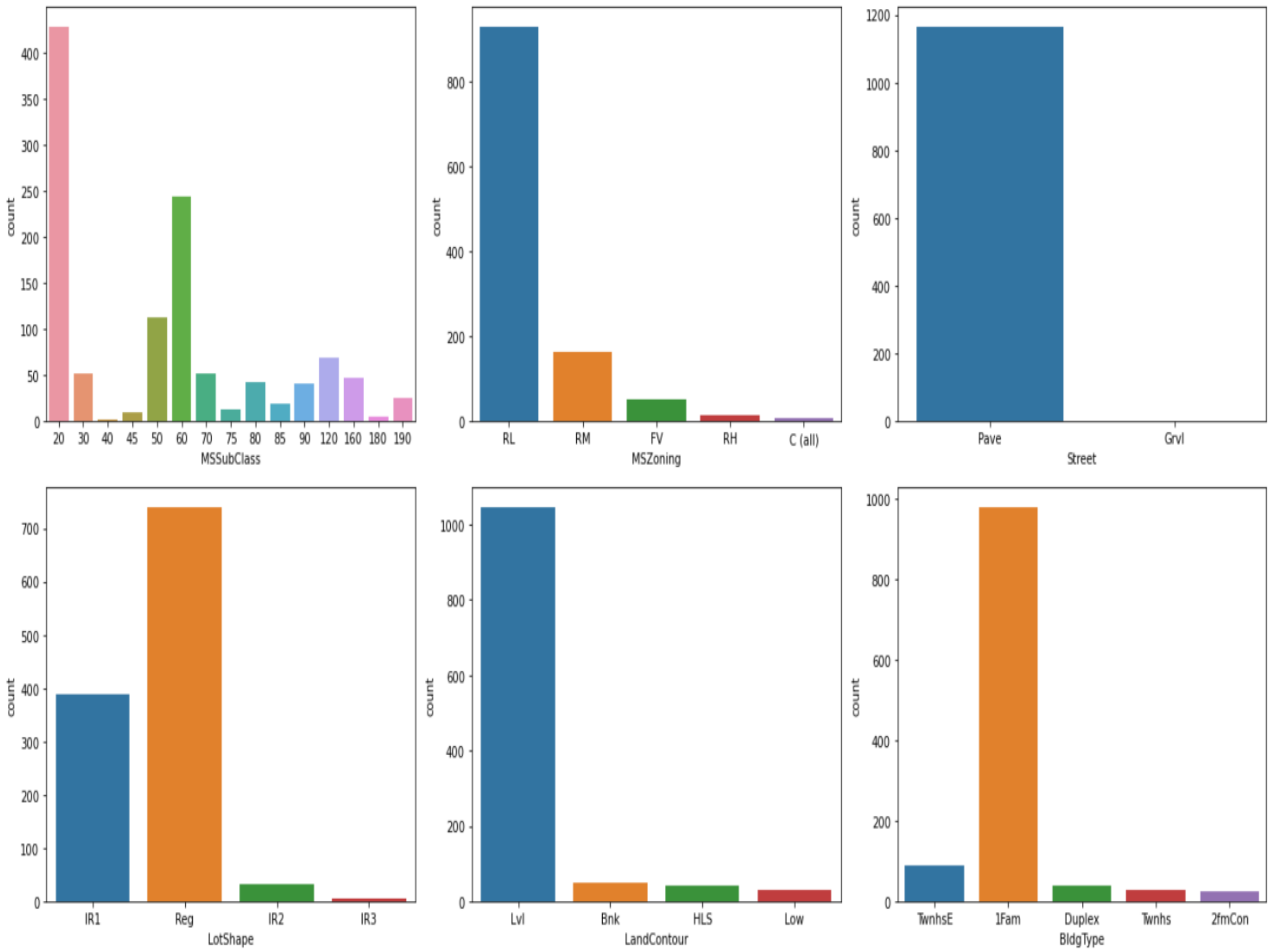
Accuracy score : It is the ratio of the total number of correct predictions and the total number of predictions. It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar

F1-score : It is used when the False Negatives and False Positives are crucial. Hence F1-score is a better metric when there are imbalanced classes.

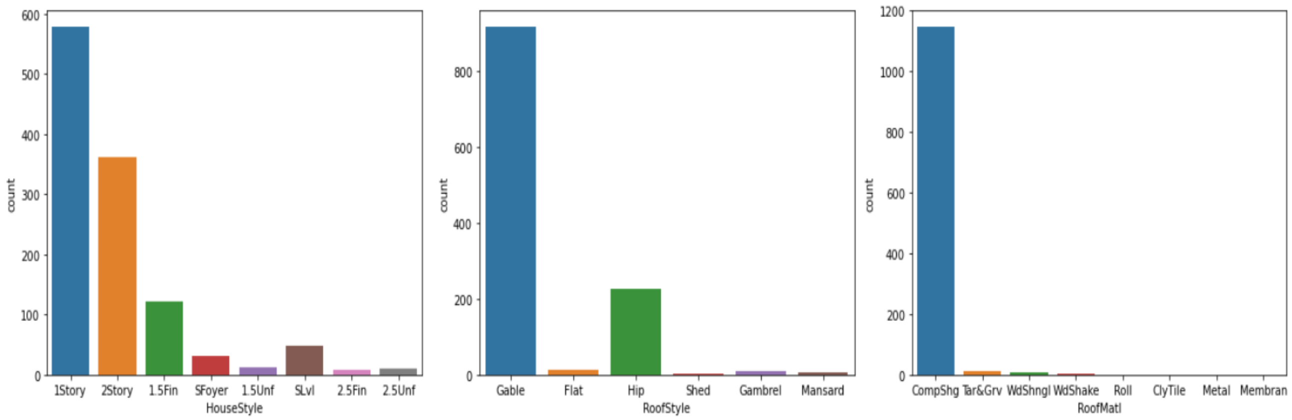
Cross_val_score : To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross- validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

roc _auc _score : ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

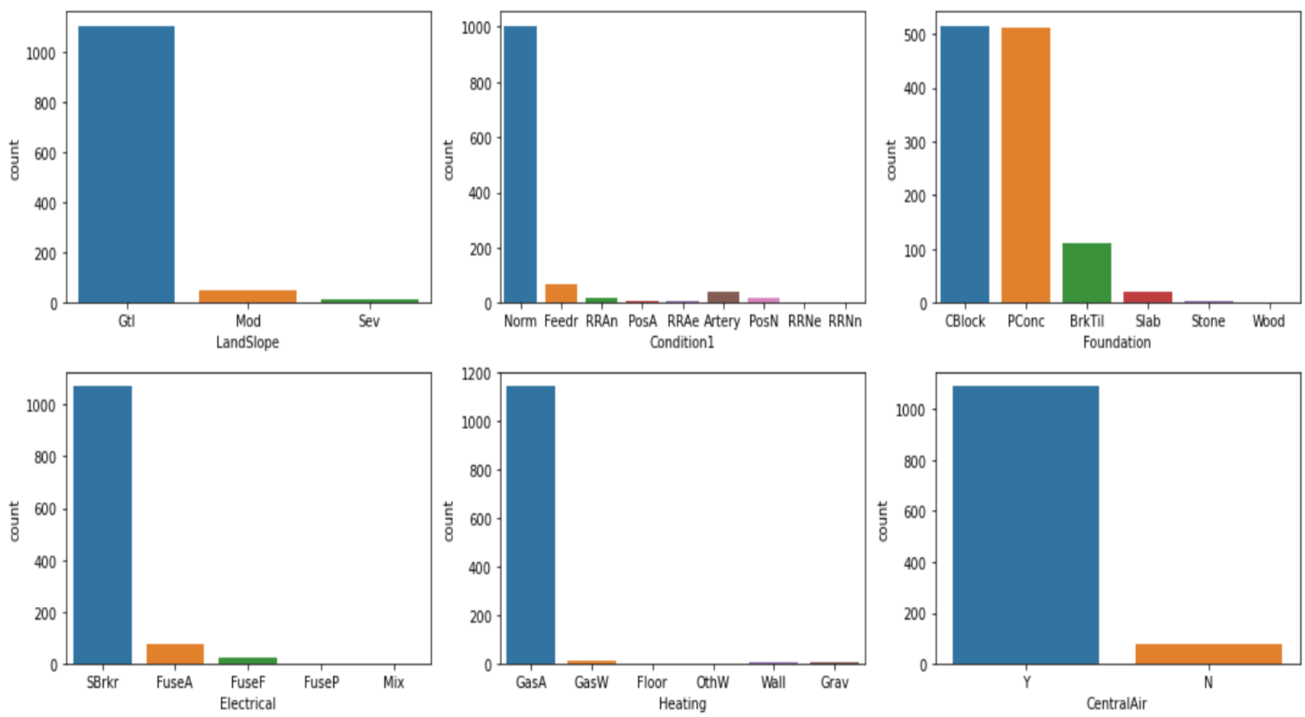
VISUALIZATION



- Mostly 1-STORY 1946 & NEWER ALL STYLES type of dwelling is involved in the sale.
- Approx 80% properties are in residential low density area.
- For road access to property mostly its a paved road surface.
- Around 63% shape of properties is regular.
- Approx 90% land contour of properties is flat/level.
- Mostly properties is of single-family home building type

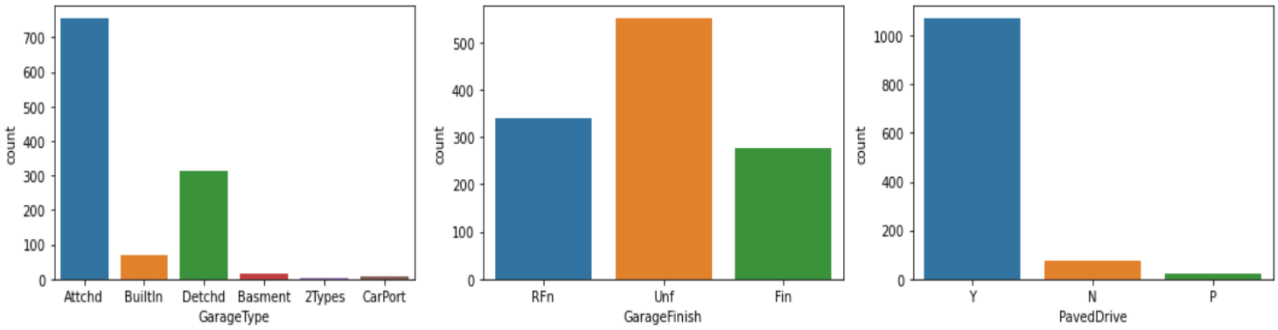


- Approx 50% properties is a ground storey house(single-storey).
- Mostly houses have around 78% gable type of roof.
- Almost all propertes have Standard (Composite) Shingle type of roof material installed.



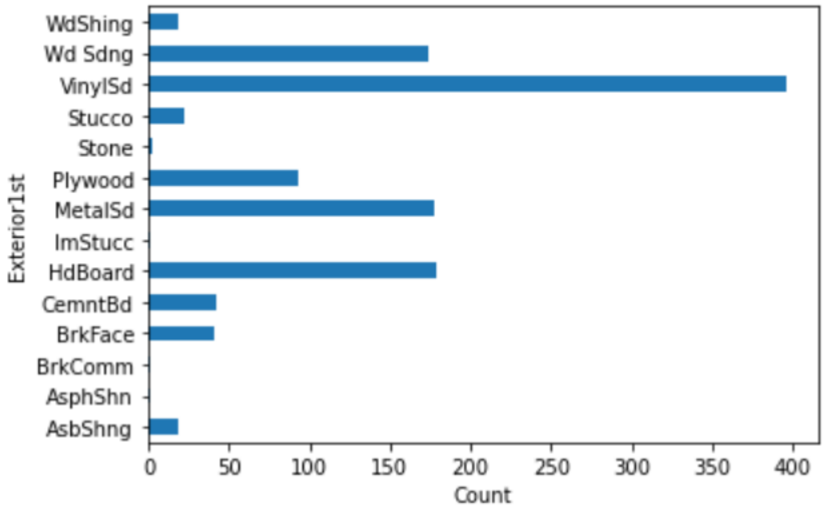
- Approx 95% properties is located on gentle slope.
- Mostly properties are adjusted to normal condition.
- Mostly foundation of properties is done by cinder block & poured concrete.

- Standard Circuit Breakers & Romex electrical system is widely used for houses.
- Almost all have GasA (Gas forced warm air furnace) type of heating in their houses.
- Approx 93% houses have central air conditioning setup.



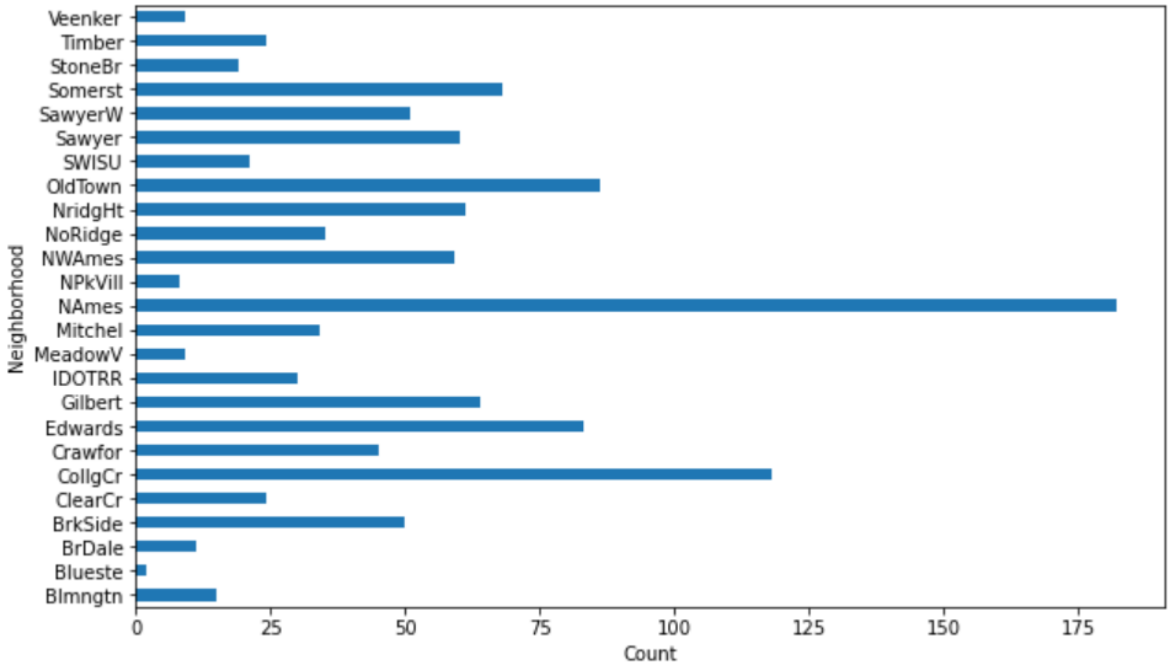
- Approx 65% houses have garage attached to their houses.
- Mostly houses garage interior finishing is not done.
- Approx 92% houses have paved driveway to their mansion.

```
df1.groupby('Exterior1st')['Exterior1st'].count().plot(kind='barh')
plt.xlabel('Count')
plt.show()
```

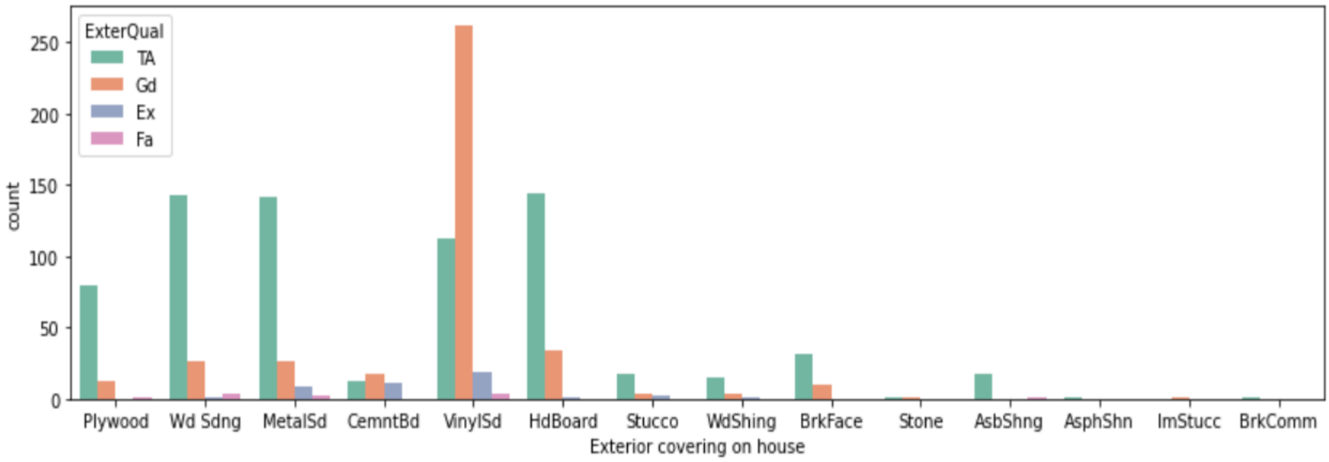


- Mostly Vinyl Siding have been installed to cover exterior of their houses.

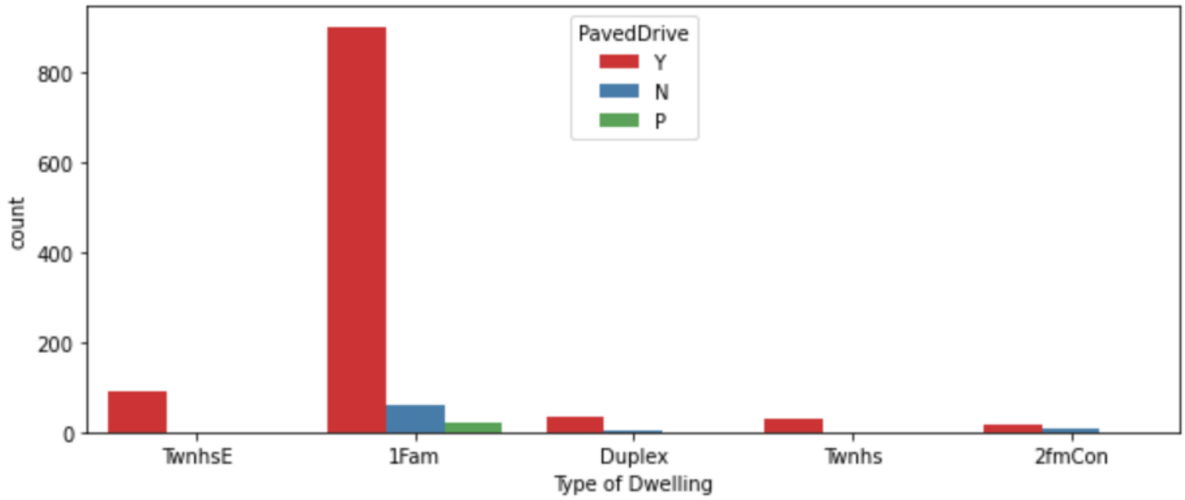
```
plt.subplots(figsize=(10,6))
df1.groupby('Neighborhood')['Neighborhood'].count().plot(kind='barh')
plt.xlabel('Count')
plt.show()
```



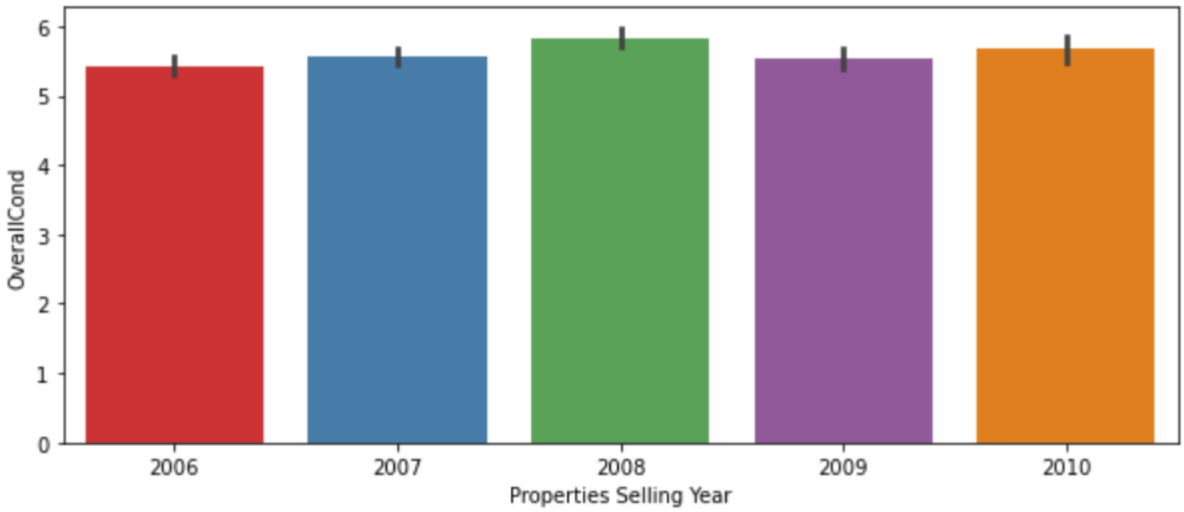
- Mostly properties are located nearby Northwest Ames.



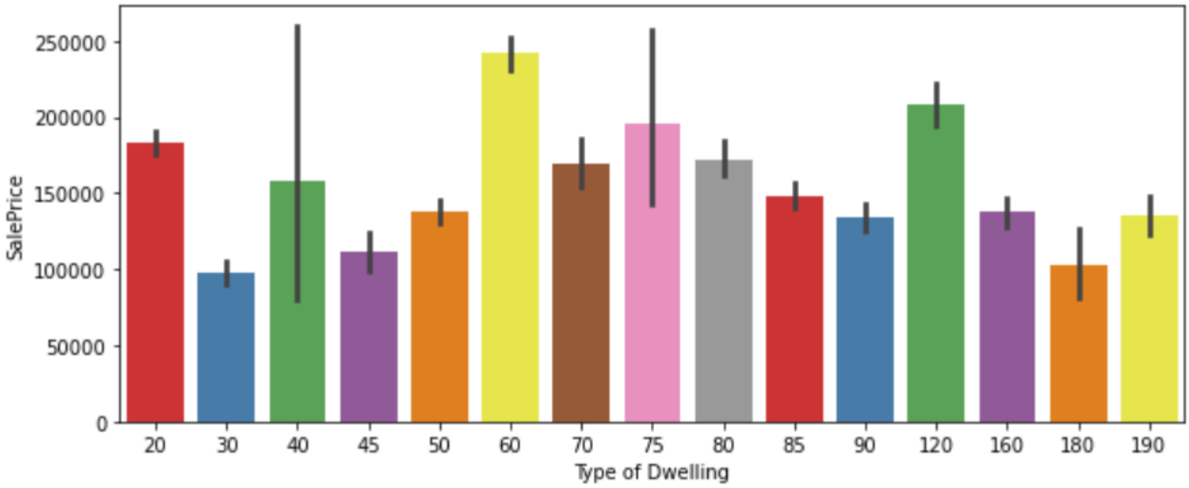
- In terms of quality Vinyl Siding is the best among all materials used for exterior covering of houses.



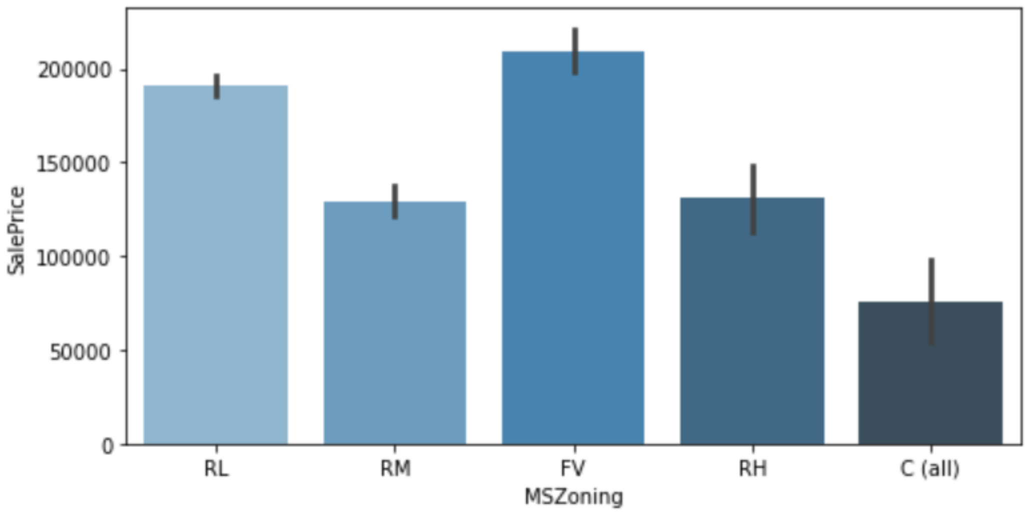
- Mostly all type of houses have their driveway paved only.



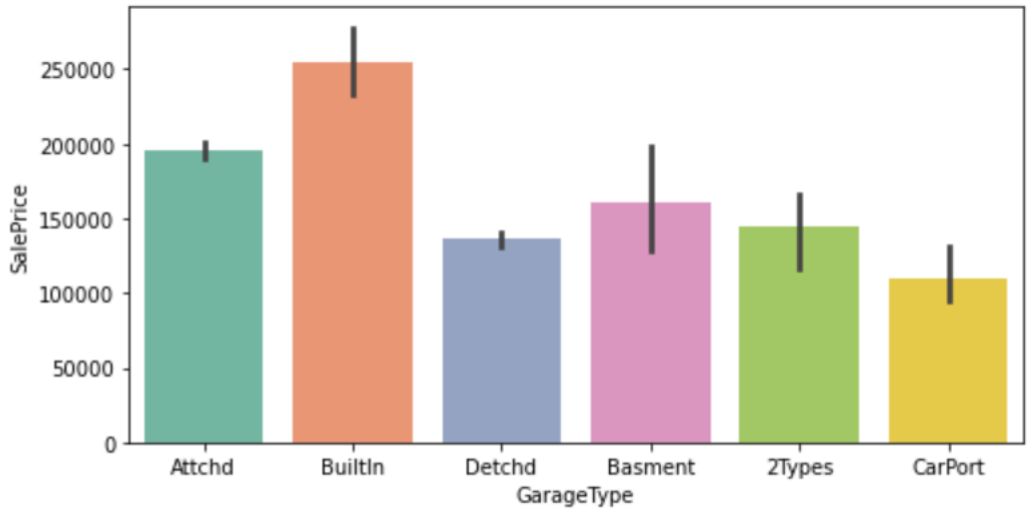
- The overall condition of properties when sold was mostly average.



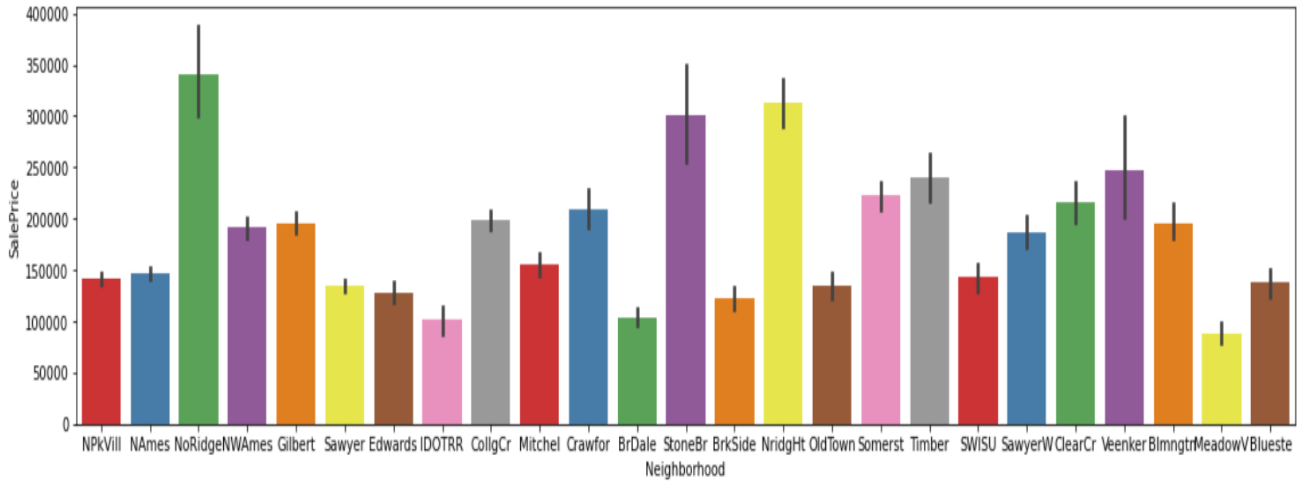
- 2-STORY 1946 & NEWER type of houses saleprice is maximum than others.



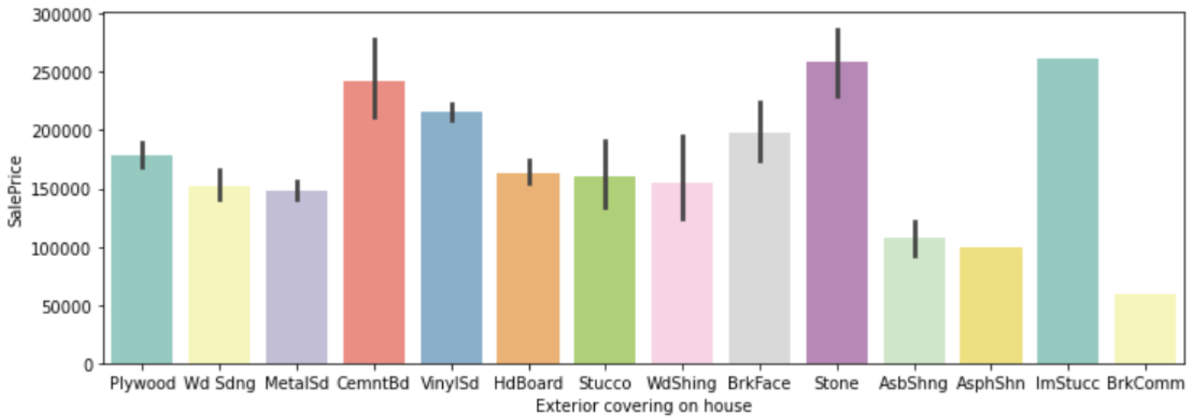
- Floating Village & Low Density Residential houses saleprice is most than others.



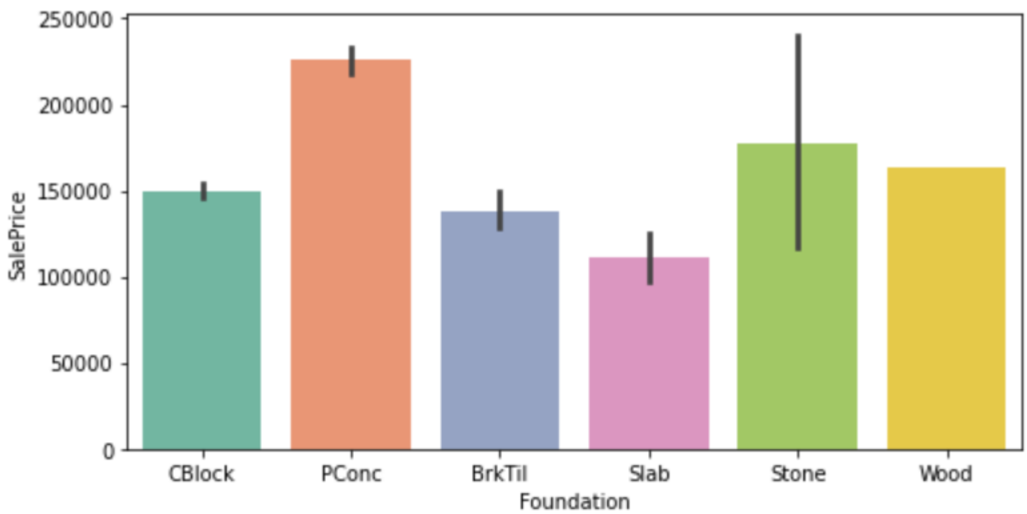
- Saleprice of those houses are high which has Built-In (Garage part of house - typically has room above garage).



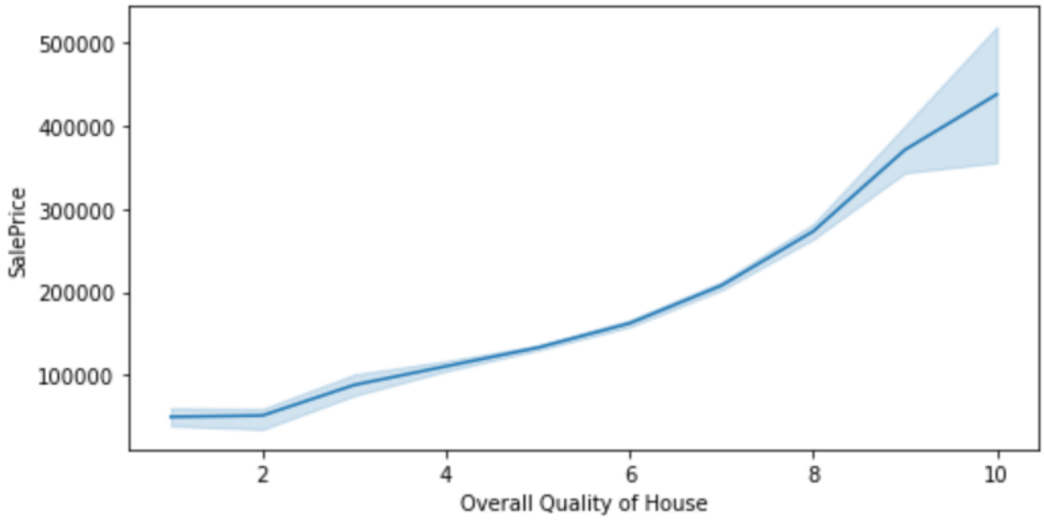
- Saleprice of properties located nearby North Ridge is very high.
- Saleprice of properties located nearby Meadow Village is least.



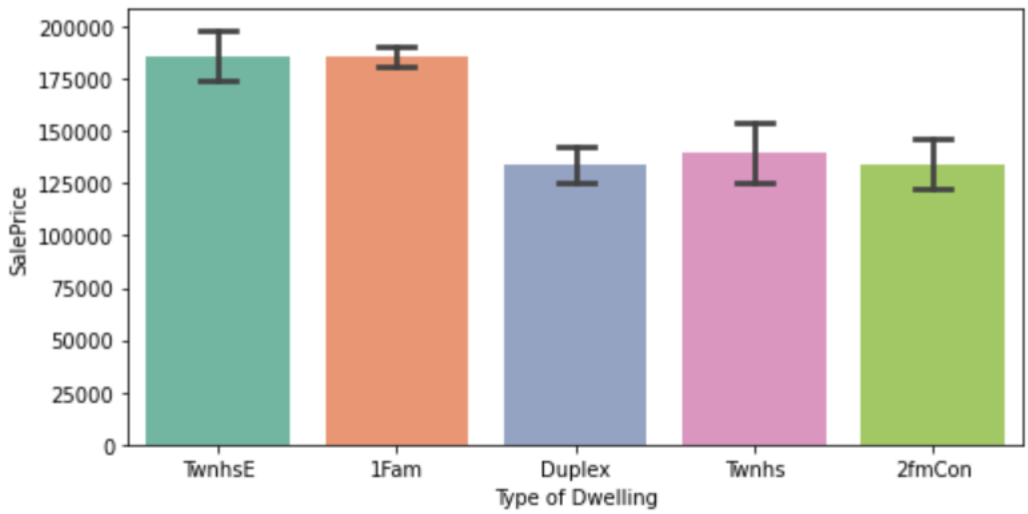
- Saleprice of properties is high if its exterior covering is done by stone and cement board.



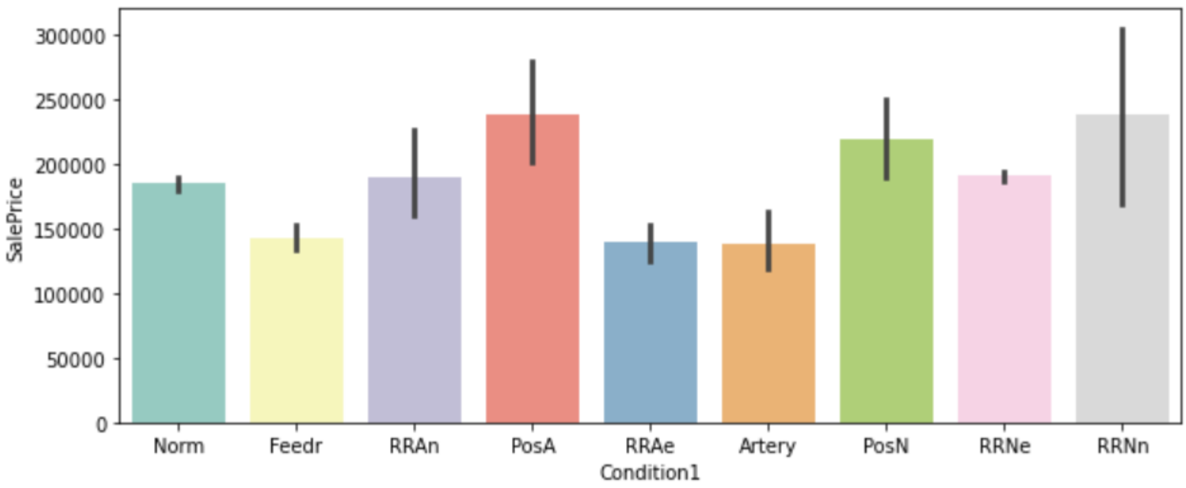
- Those houses whose foundation is built by poured concrete has maximum saleprice than others.



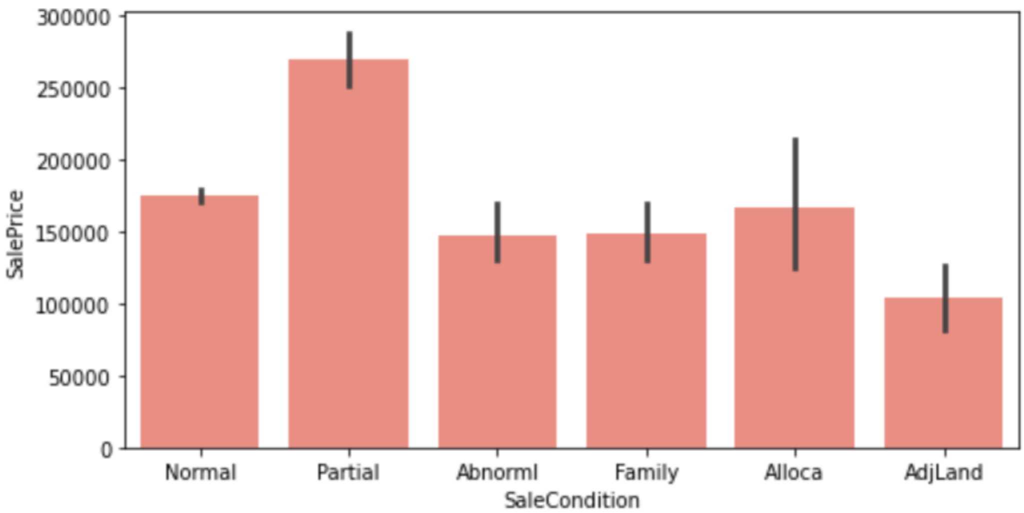
- Saleprice of house increases with increase in overall rating for material and finish of the house.



- Single family house & Townhouse End Unit types of properties saleprice is much higher than other type of dwellings.



- Saleprice of properties is high if located Within 200' of North-South Railroad and also Adjacent or near to postive off-site feature like park,greenbelt etc.



- The saleprice of Partial houses which was not completed when last assessed (associated with New Homes) is maximum than others.

INTERPRETATION OF RESULTS

From visualization it was observed that vinyl siding was used mostly for exterior covering of houses to withstand weather and resist moisture damage. Mostly driveway of properties were paved. Price range of properties increases if its exterior covering is done by stone & cement board. Foundation of houses built by concrete has highest saleprice. Overall quality variable was having the highest positive correlation with target variable. Properties saleprice increases if it is located to places where parks, malls, highway etc is nearby.

From preprocessing it was observed that it’s a regression problem as we have to predict the saleprice of properties with train and test dataset. Where we have to build models on train data and predict on test data. Both train and test dataset contained null values which was treated accordingly. Doing label encoding to convert categorical variables into machine language for both train and test dataset. Through variance inflation factor removed variables to deal with multicollinearity problems.
Splitting the train dataset into two, keeping 25% for testing and rest 75% for training to build ML models.

From modelling it was observed that XGB Regressor was our best model because the difference between its accuracy and CV score was least among all models. Then applied hyper parameter tuning on our best model and the accuracy increased by 2.03%.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

From the whole project evaluation these are the inferences that I could draw from the visualization of data.

- The analysis showed that OverallQual, TotalBsmtSF, GarageCars, GarageArea and GrLivArea variables were important determinants of target (Saleprice) variable.
- OverallQual variable plays a huge role in deciding the price of a house.
- People do not ignore house foundation material when purchasing a house. High quality house, based on its foundation material, also affects pricing.
- Location of properties & accessibility to highway, shopping malls, park, market, hospitals etc plays an important role in affecting house prices.
- The newer the house the higher is the price.
- Properties size, appeal and usable space is an important element to consider.
- Remodelling of houses can boost its value overtime.
- If garage, central air conditioning & pool are available then these will have a significant impact on the sale price of houses.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

Through data visualization lot of variables depicted a lot of story telling when visualization was done for all of them. Visualization gives meaning to a raw data which helps drawing inference from it.

Challenges faced was that lot of variables were having null values, also variables having same categorical values with slightly different variable name.

To handle outliers 1.5 IQR method was used and also kept in mind to not lose 7-8% of data.

Lasso and Ridge regression gave same r2 score which means our linear regression model has been well trained over the training data and there is no overfitting

XGB Regressor was the best model to be deployed because the difference between its accuracy and CV value was least among all models.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

The dataset provided was less. If large dataset was given then building ML models with good accuracy would have been a bit challenging and exciting to do.

The dataset provided contained almost all features/variables that were required for prediction of price of properties but some variables were having large number of missing values which can be important for futher analysis.

If I could add some features it might be lawn not large but small or medium will do at the front side.

Directions of house could improve results means whether the house is east facing where sun rises early morning will be good strategy to flip those kind of houses in higher prices by talking about their advantages for its direction.

Houses with proper vaastu is a must nowadays so before investing in a house we have to totally go through each and every point that will make our buyer believe that this will be perfect for them.