NAME OF THE PROJECT

HOUSE PRICE PREDICTION

Submitted by:

RANJEET JANAGOUDA

ACKNOWLEDGMENT

First and foremost, I would like to thank Flip Robo Technologies to provide me a chance to work on this project. It was a great experience to work on this project under your guidance.

I would like to present my gratitude to the following websites:

- Zendesk
- Kaggle
- Datatrained Notes
- Sklearn.org
- Crazyegg

These websites were of great help and due to this, I was able to complete my project effectively and efficiently.

INTRODUCTION

- Business Problem Framing

You are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

- Conceptual Background of the Domain Problem

Basic EDA concepts and regression algorithms must be known to work on this project. One should know what is Housing Price and how it is going to affect the real estate business. Why predicting the house prices is important and how can it is going to help the company?

- Review of Literature

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

A US-based housing company named **Surprise Housing** 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. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

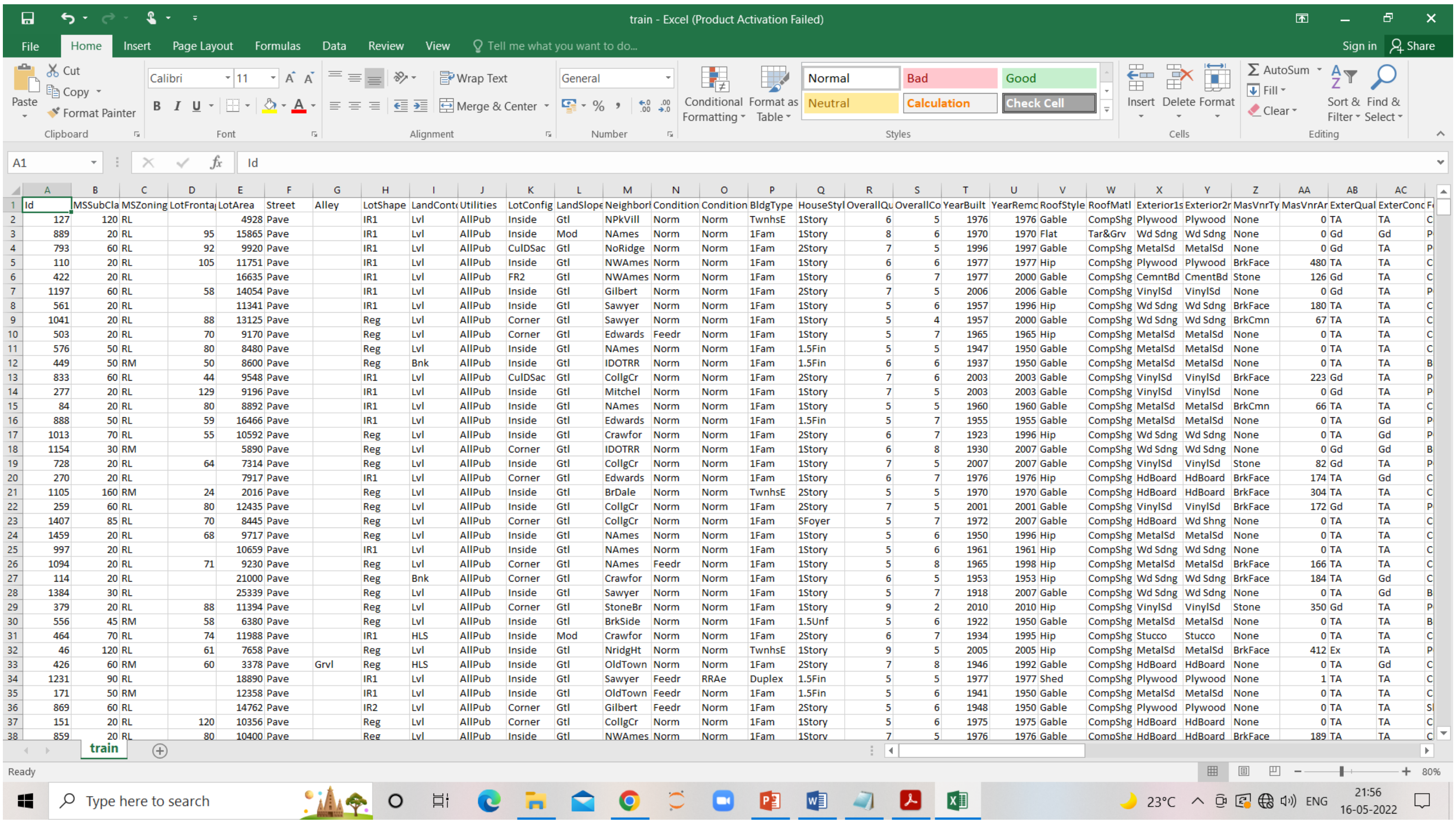
The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

Analytical Problem Framing

- ## Data Sources and their formats

The dataset is provided the internship organization in an excel format which contains the data in both in code sheet and categorical data. It contains 81 columns and 1168 rows. There are so many factors which can be used for the prediction of sale price of a house. It contains the factors on which the sale price of a house can depend. Dataset contain both numerical as well as categorical data.



- ## Libraries Used

I am using different libraries to explore the datatset.

1. Pandas – It is used to load and store the dataset. We can discuss the dataset with the pandas different attributes like .info, .columns, .shape
2. Seaborn – It is used to plot the different types of plots like catplot, lineplot, countplot and more to have a better visualization of the dataset.
3. Matplotlib.pyplot – It helps to give a proper description to the plotted graph by seaborn and make our graph more informative.
4. Numpy – It is the library to perform the numerical analysis to the dataset

Load the Dataset

Importing the training dataset

In [2]:
```
pd.set_option('display.max_rows',None)  #setting the display option to max
pd.set_option('display.max_columns',None)
train=pd.read_csv(r'F:\Internship - Data Science\Project-Housing--2---1-\Project-Housing_splitted\train.csv')
train.head()
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Cond
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	

Importing the test dataset

In [3]:
```
test=pd.read_csv(r'F:\Internship - Data Science\Project-Housing--2---1-\Project-Housing_splitted\test.csv')
test.head()
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Con
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr	

We have successfully load our both the dataset, test & train for our further processes.

Checking the Attributes

- First & last five rows of both the dataset
- Shape of the datasets
- Columns present in the datasets
- Brief info about the datasets
- Null values present in both the dataset

Shape of both the train & test dataset

In [6]:
```
print('Shape of training dataset: ',train.shape)
print('Shape of testing dataset: ',test.shape)
```

```
Shape of training dataset:  (1168, 81)
Shape of testing dataset:  (292, 80)
```

Columns of the train & test dataset

In [7]:
```
train.columns #contains the sale price column
```

Out[7]:
```
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')
```

In [8]:
```
test.columns #the sale price is to be predicted
```

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',
```

A brief info about the train & test dataset columns

In [9]:
```
train.info()
```

```
 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
```

In [10]:
```
test.info()
```

```
<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
```

Checking the Null Values

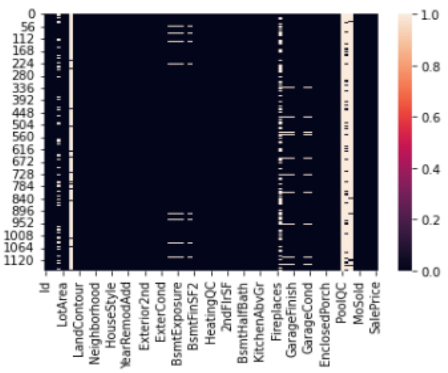
In [11]:
```
train.isnull().sum()
```

```
LotFrontage      214
LotArea            0
Street             0
Alley           1091
LotShape           0
LandContour        0
Utilities          0
LotConfig          0
LandSlope          0
Neighborhood       0
Condition1         0
Condition2         0
BldgType           0
HouseStyle         0
OverallQual        0
OverallCond        0
YearBuilt          0
YearRemodAdd       0
RoofStyle          0
RoofMatl           0
```

We have so many columns with null values that has to be handled

In [12]:
```
sns.heatmap(train.isnull())  #null values using the heatmap
```

Out[12]: <AxesSubplot:>



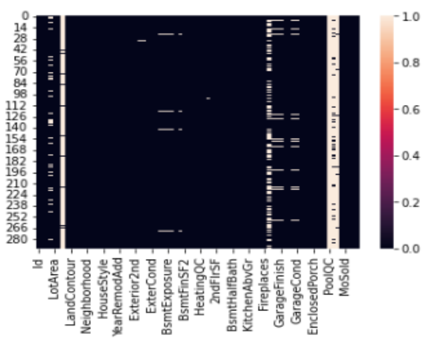
In [13]:
```
test.isnull().sum()
```

Out[13]:
```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage      45
LotArea           0
Street            0
Alley           278
LotShape          0
LandContour       0
Utilities         0
LotConfig         0
LandSlope         0
Neighborhood      0
Condition1        0
Condition2        0
BldgType          0
HouseStyle        0
OverallQual       0
OverallCond       0
```

Test dataset also have null values which is to be handled separately

In [14]:
```
sns.heatmap(test.isnull())  #null values using heatmap
```

Out[14]: <AxesSubplot:>



Now we have checked the attributes for the dataset and get a rough idea about the dataset like the no of rows & columns, datatype & null values in the dataset.

Dealing with the Null Values

In both the dataset null values are present, so we have to handled them for better model learning. As we have categorical & numerical data so we have to handled them accordingly. We also drop those rows who are having more than 50&% null values.

Dropping the columns which have more than 50% null values from both the test & train dataset

```
In [15]: train.drop(["Alley","PoolQC","Fence","MiscFeature"],axis=1,inplace=True)
```

```
In [16]: test.drop(["Alley","PoolQC","Fence","MiscFeature"],axis=1,inplace=True)
```

```
In [17]: train.shape
Out[17]: (1168, 77)
```

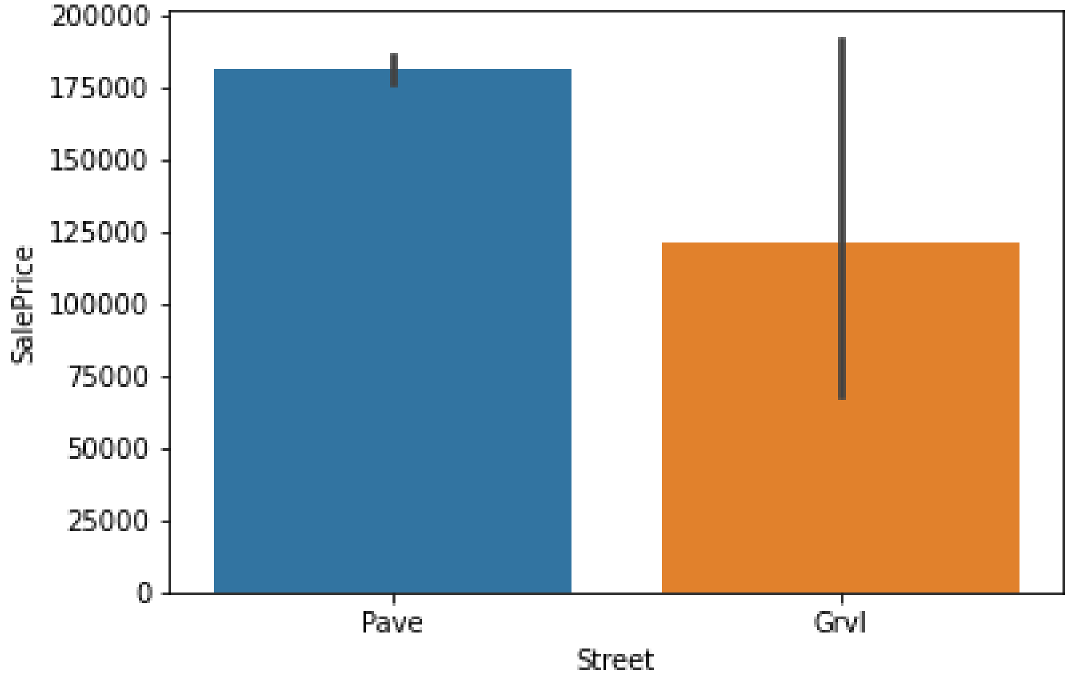
Dealing with the Null Values

```
In [19]: train['LotFrontage']=train['LotFrontage'].fillna(train['LotFrontage'].mean())
train['MasVnrType']=train['MasVnrType'].fillna('None')
train['MasVnrArea']=train['MasVnrArea'].fillna(train['MasVnrArea'].mean())
train['BsmtQual']=train['BsmtQual'].fillna('TA')
train['BsmtCond']=train['BsmtCond'].fillna('TA')
train['BsmtExposure']=train['BsmtExposure'].fillna('No')
train['BsmtFinType1']=train['BsmtFinType1'].fillna('Unf')
train['BsmtFinType2']=train['BsmtFinType2'].fillna('Unf')
train['FireplaceQu']=train['FireplaceQu'].fillna('Gd')
train['GarageType']=train['GarageType'].fillna('Attached')
train['GarageYrBlt']=train['GarageYrBlt'].fillna(2006.6)
train['GarageFinish']=train['GarageFinish'].fillna('Unf')
train['GarageQual']=train['GarageQual'].fillna('TA')
train['GarageCond']=train['GarageCond'].fillna('TA')
```

```
In [20]: test['LotFrontage']=test['LotFrontage'].fillna(test['LotFrontage'].mean())
test['MasVnrType']=test['MasVnrType'].fillna('None')
test['MasVnrArea']=test['MasVnrArea'].fillna(test['MasVnrArea'].mean())
test['BsmtQual']=test['BsmtQual'].fillna('TA')
test['BsmtCond']=test['BsmtCond'].fillna('TA')
test['BsmtExposure']=test['BsmtExposure'].fillna('No')
test['BsmtFinType1']=test['BsmtFinType1'].fillna('Unf')
test['BsmtFinType2']=test['BsmtFinType2'].fillna('Unf')
test['FireplaceQu']=test['FireplaceQu'].fillna('Gd')
test['GarageType']=test['GarageType'].fillna('Attached')
test['GarageYrBlt']=test['GarageYrBlt'].fillna(2006.6)
test['GarageFinish']=test['GarageFinish'].fillna('Unf')
test['GarageQual']=test['GarageQual'].fillna('TA')
test['GarageCond']=test['GarageCond'].fillna('TA')
test['Electrical']=test['Electrical'].fillna('SBrkr')
```

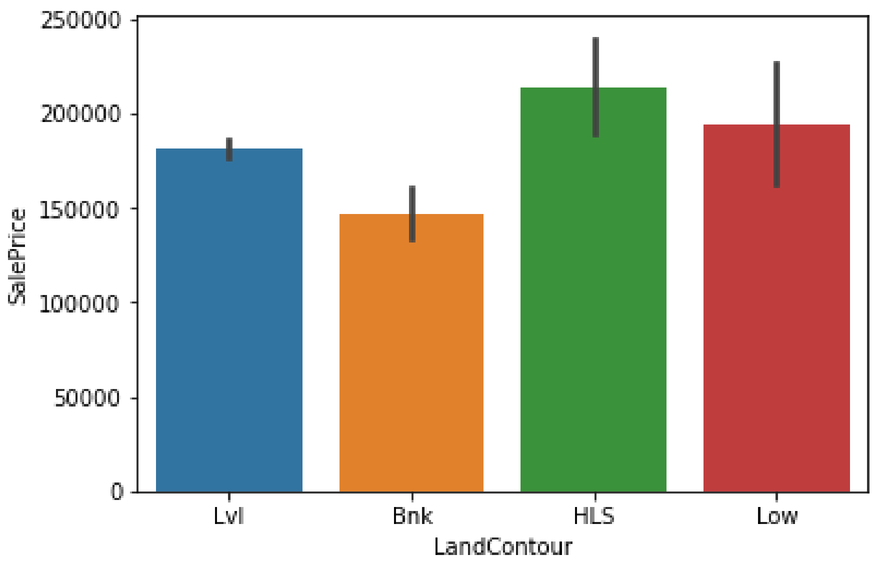
EXPLORATORY DATA ANALYSIS

Which street house has higher price?



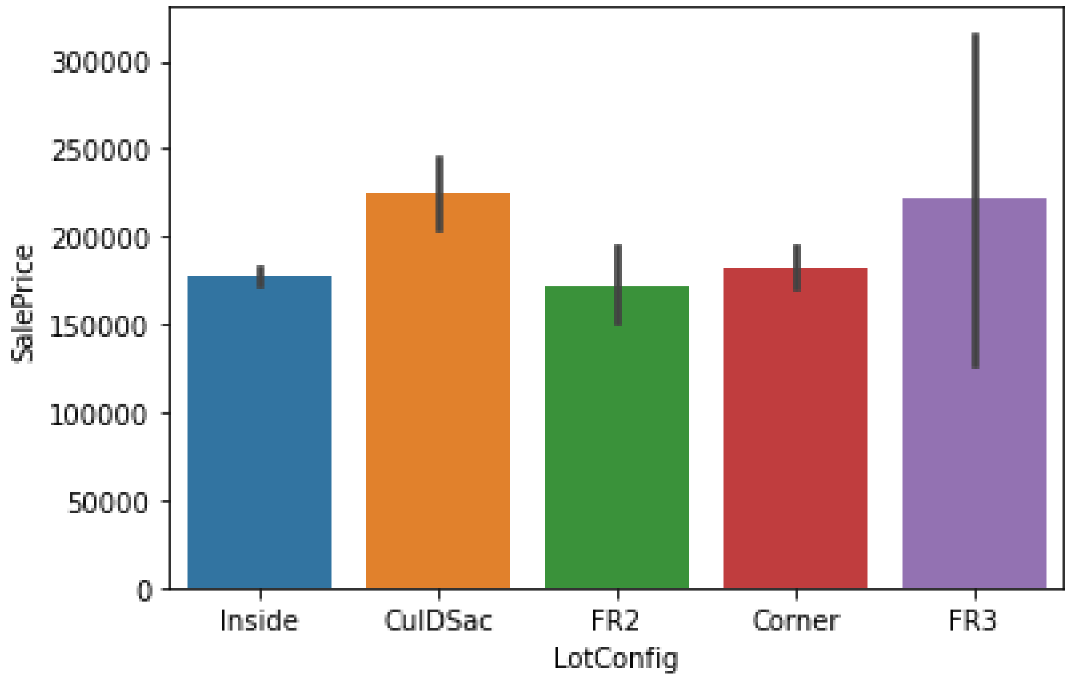
House in Pave street have higher sale price

What type of Land Contour has higher sale price?



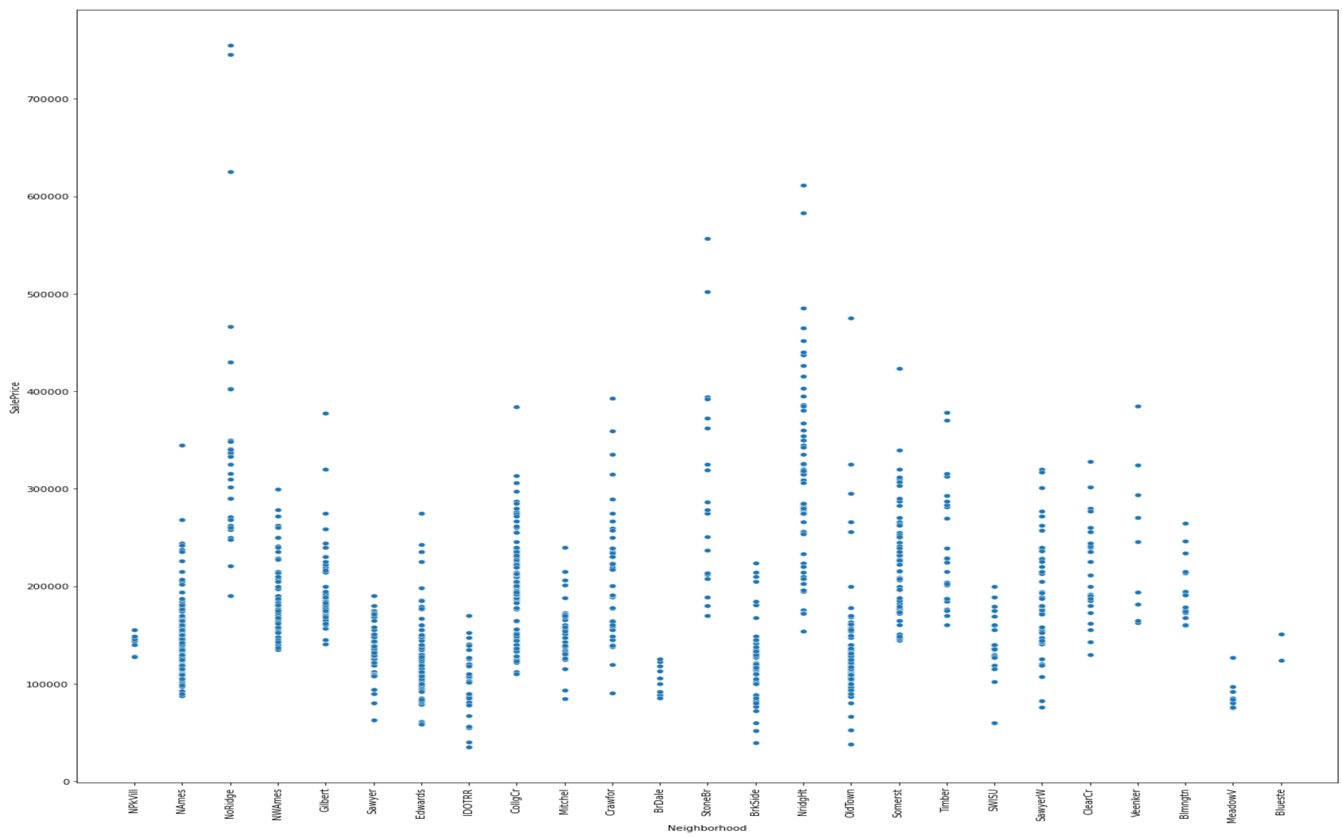
HLS type houses has higher sale price

What Lot configuration is in higher demand?



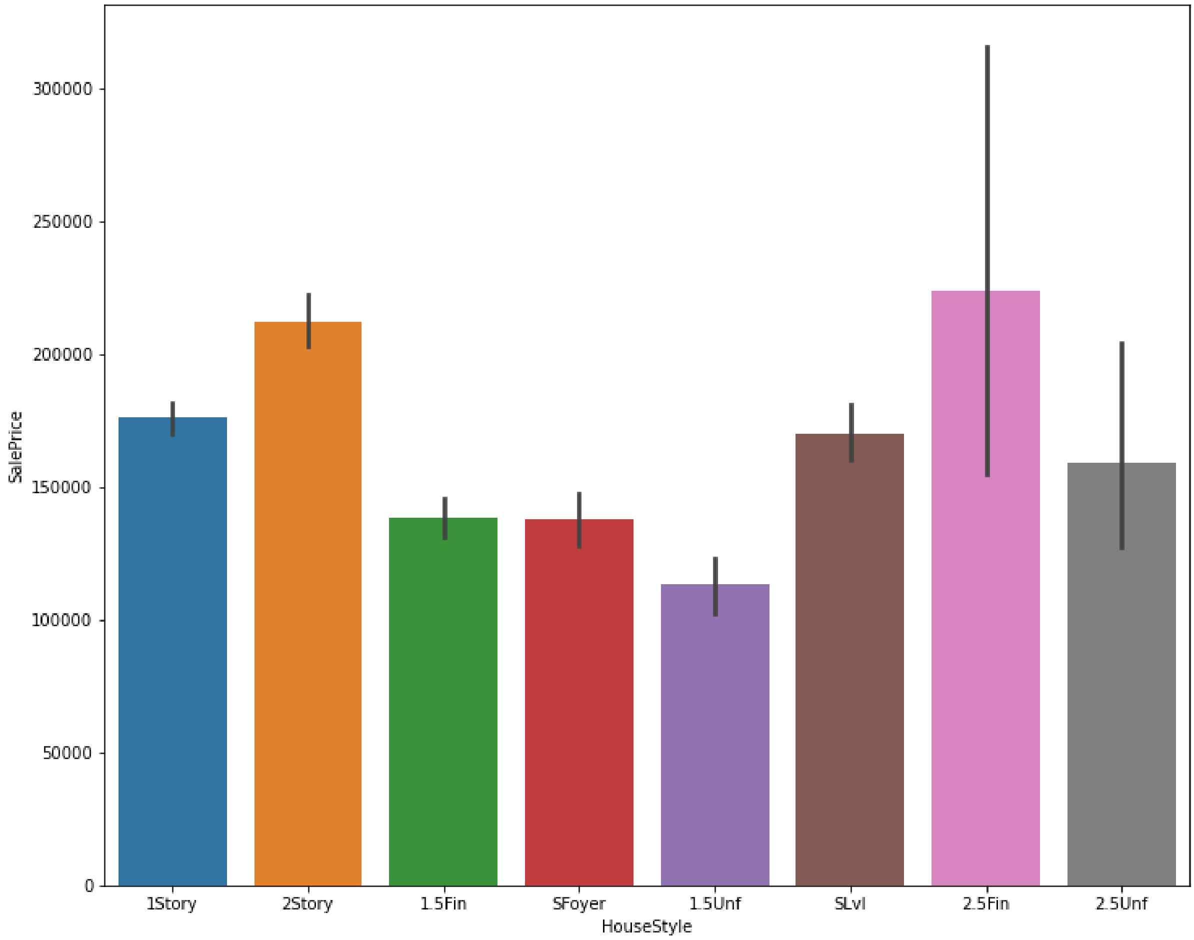
CulDSac followed by FR3 lot configuration are in higher demand.

Whose neighborhood increased the sale price?



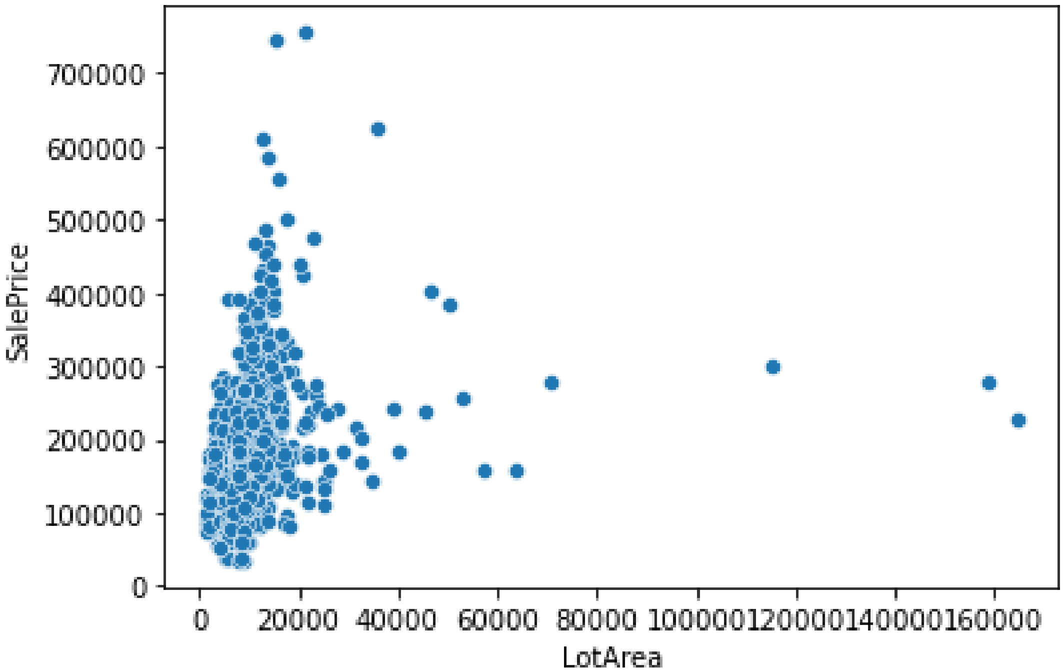
The one who has NoRidge & NridgHt in their neighbourhood has the high sale price. The one who has NPkVill & Bluestee in the neighbourhood are on the lower side

Which house style has high sale price?



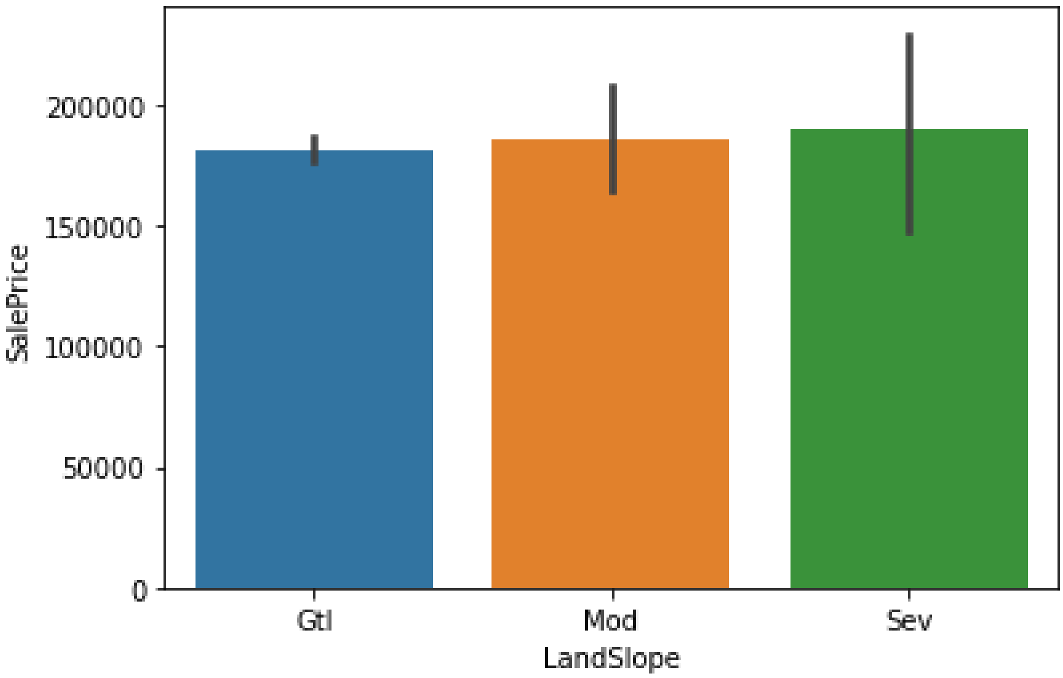
The 2.5Fin has the highest sale price followed by 2Story and 1.5Unf has the lowest sale price.

How lot area affects the price?



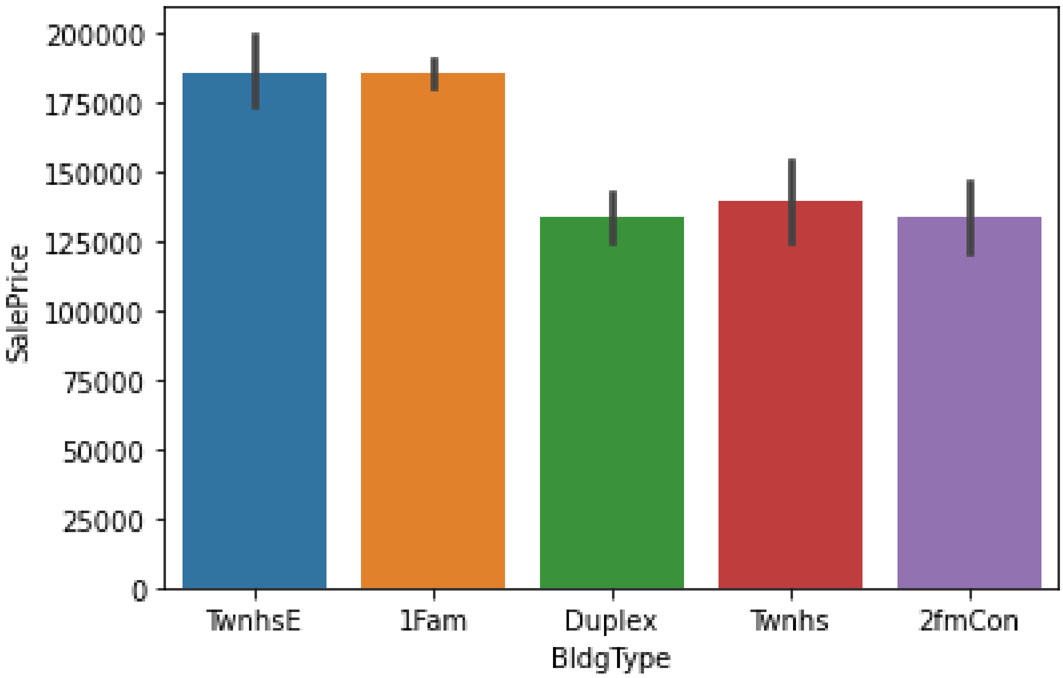
Most of the houses have low lot area, very little on the higher side & the price is very high for some of the houses

How land slope affects the price?



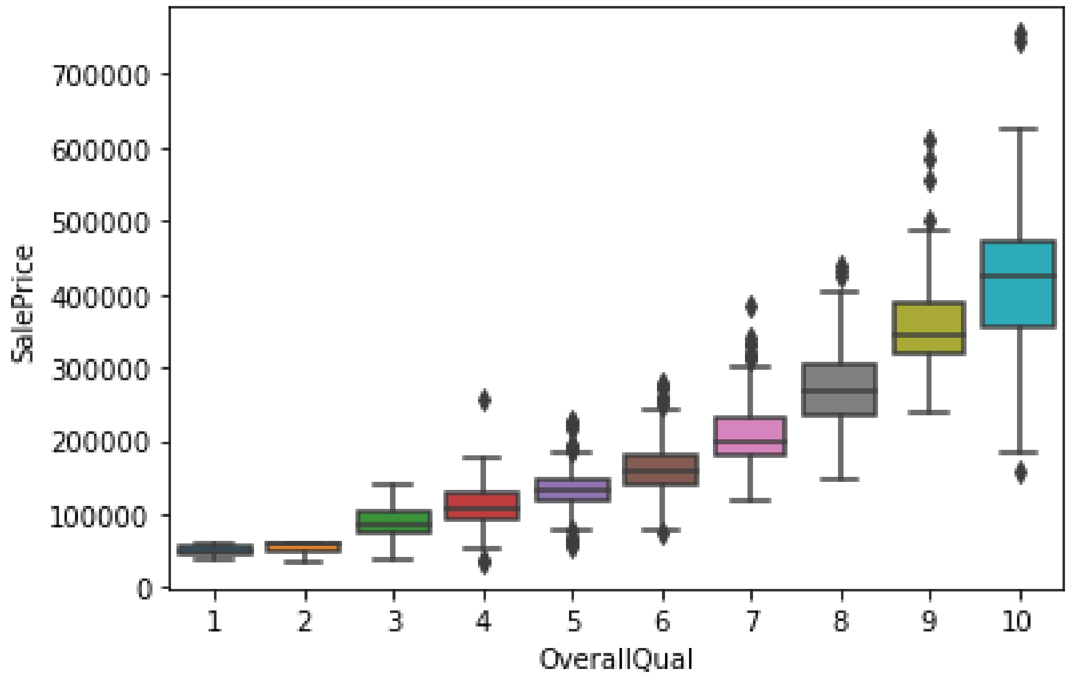
Land slope doesn't affect the price much more; it is same almost for every type.

What type of building has high sale price?



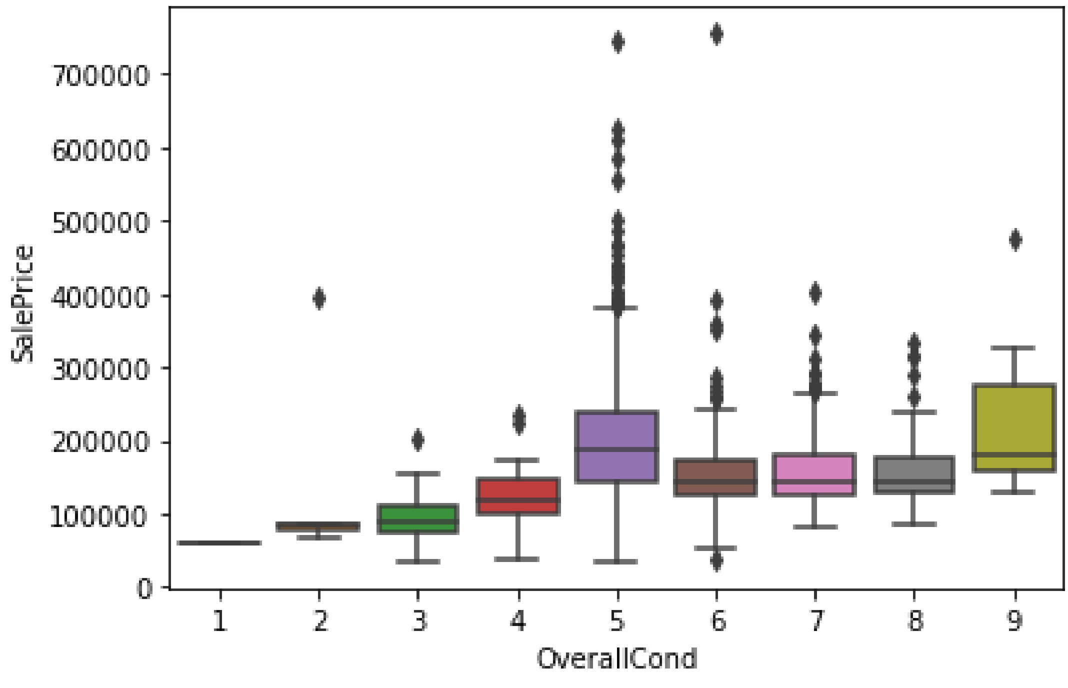
TwnhsE & 1Farm type buildings are on higher side.

OverallQual Vs Price



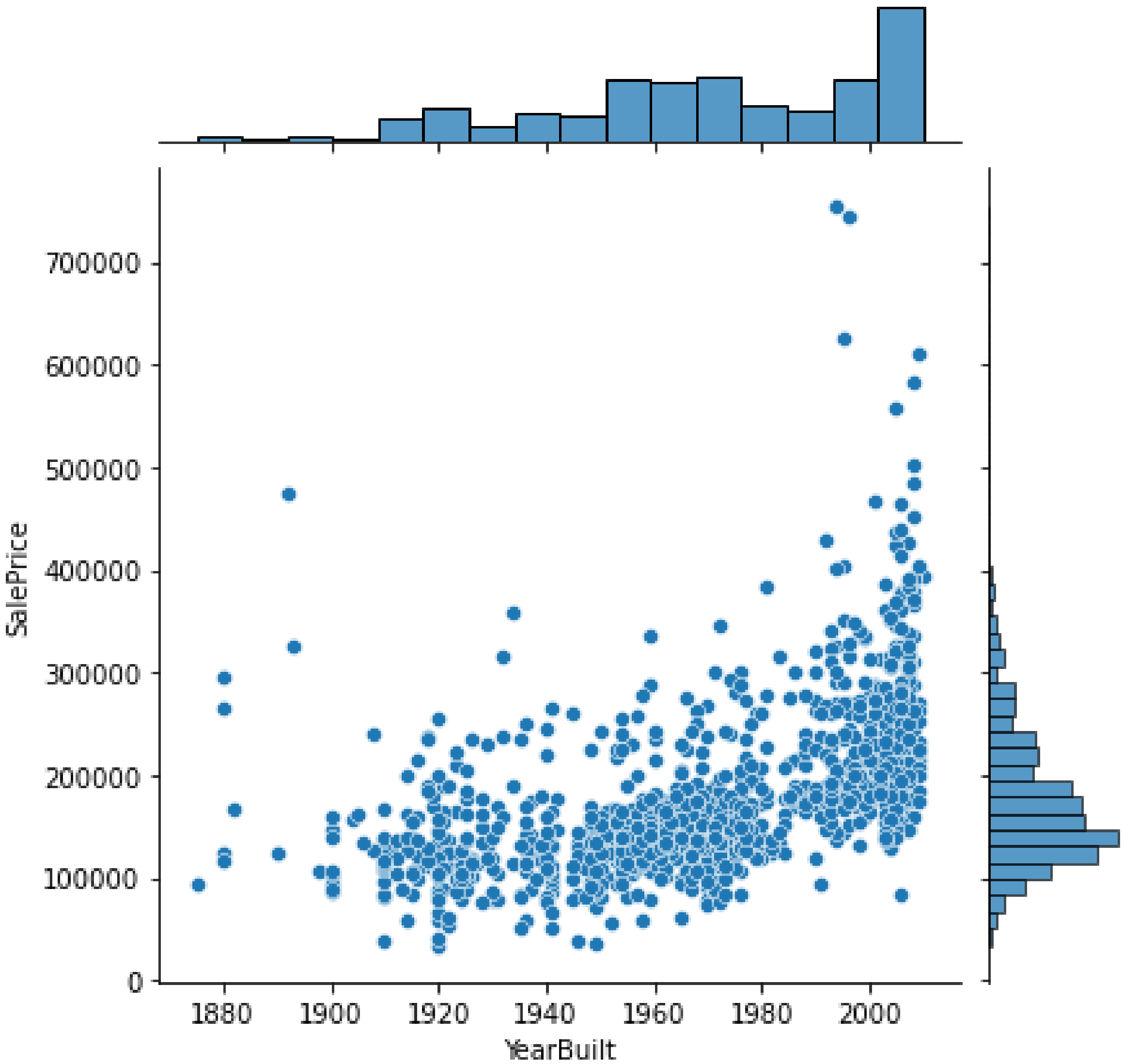
A higher overall quality grade means a higher sale price.

Overall Condition Vs Sale Price



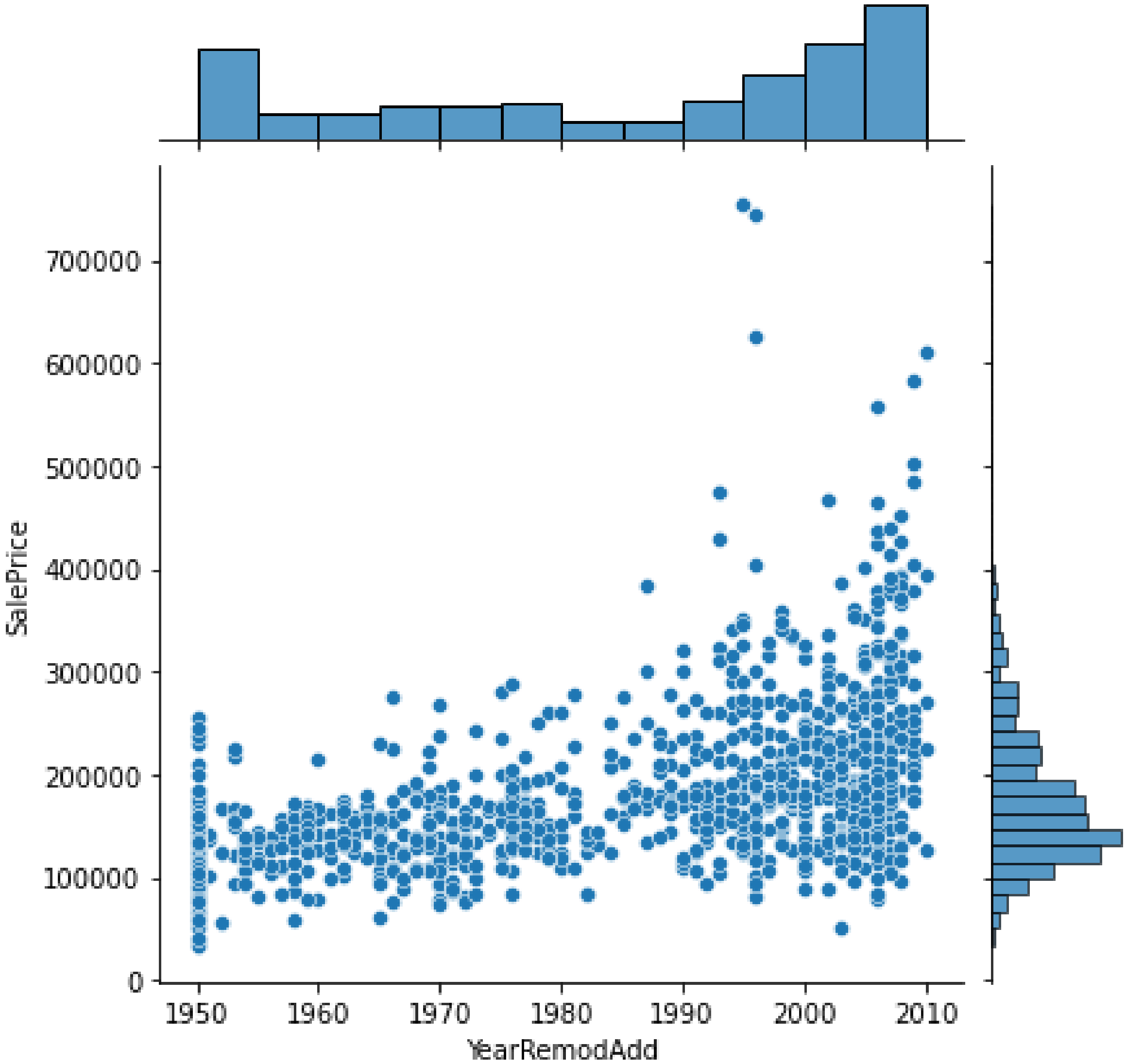
Whose overall condition is around 5 touches the higher side of price

Year built Vs Sale Price



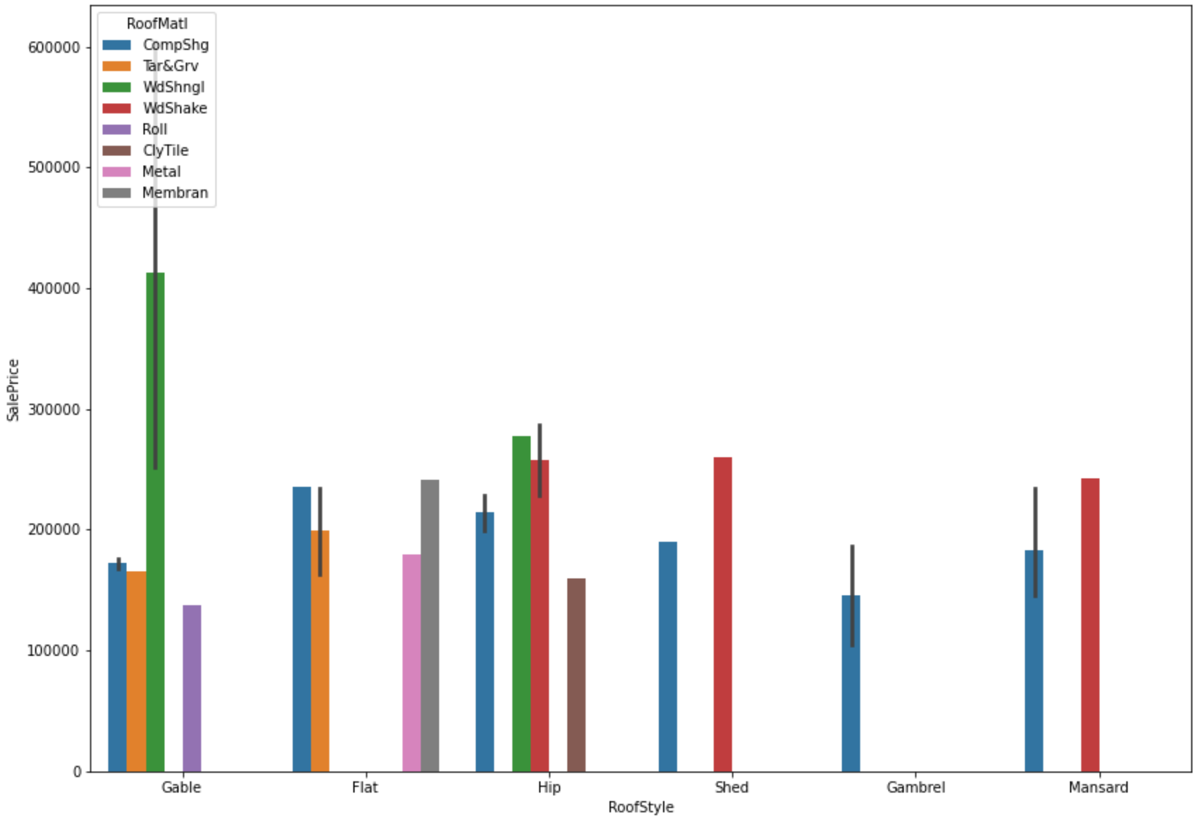
Newer houses sale price is high as compared to old houses but there are also some old houses whose sale price is high

YearRemodAdd Vs Sale Price



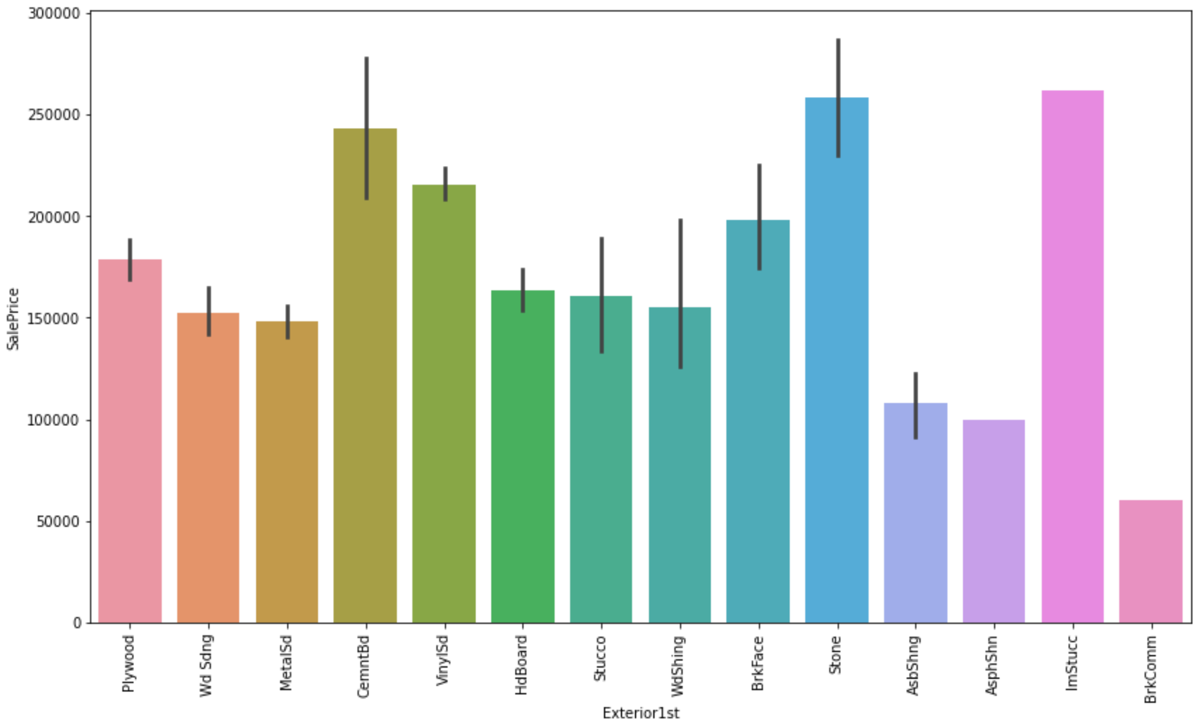
We almost have a distributed data in this but yes newer one has higher price than older one.

Which roof style & material increases the price of a house?

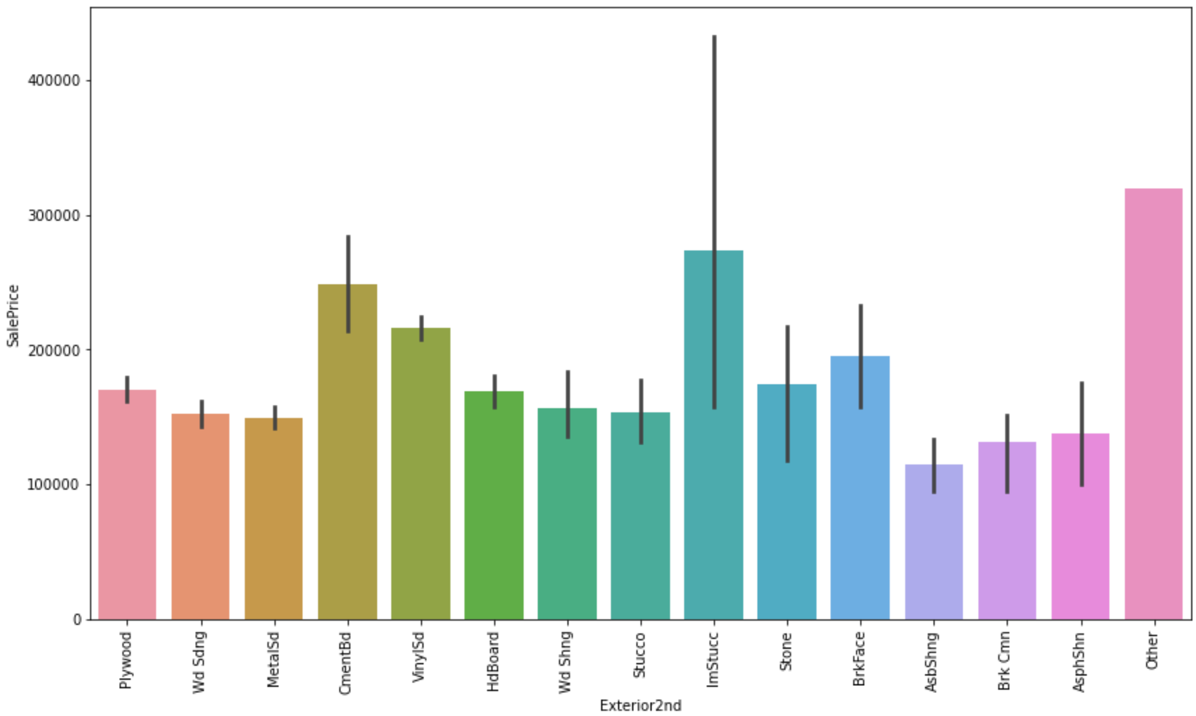


A house with Gable roof style and made of WdShngl shown up a with higher sale price

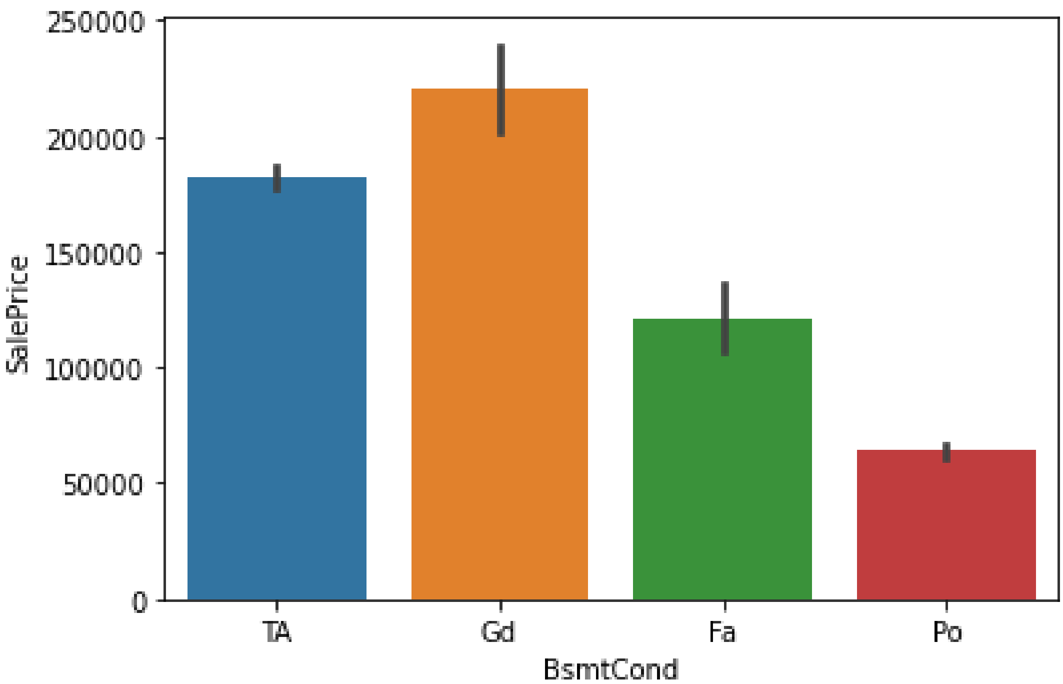
Which exterior contribute more towards sale price?



ImStucc & Stone followed by CementBd exterior sale price is high compare to others

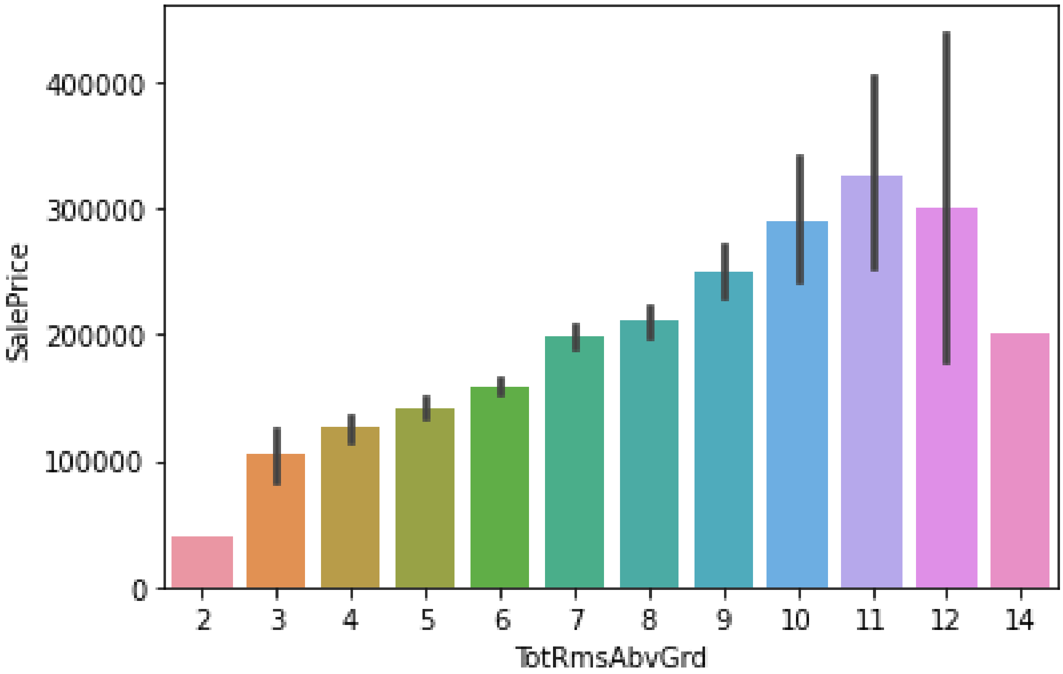


Basement condition Vs Price



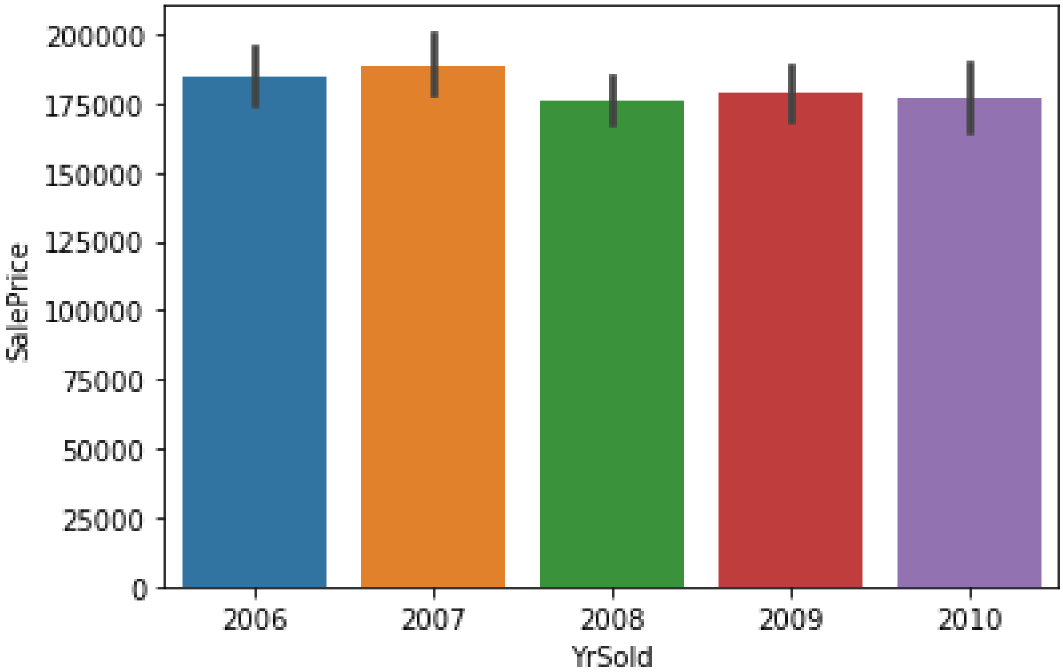
Whose condition is GD or TA then obviously getting the higher sale price

Total Rms above ground Vs Sale price



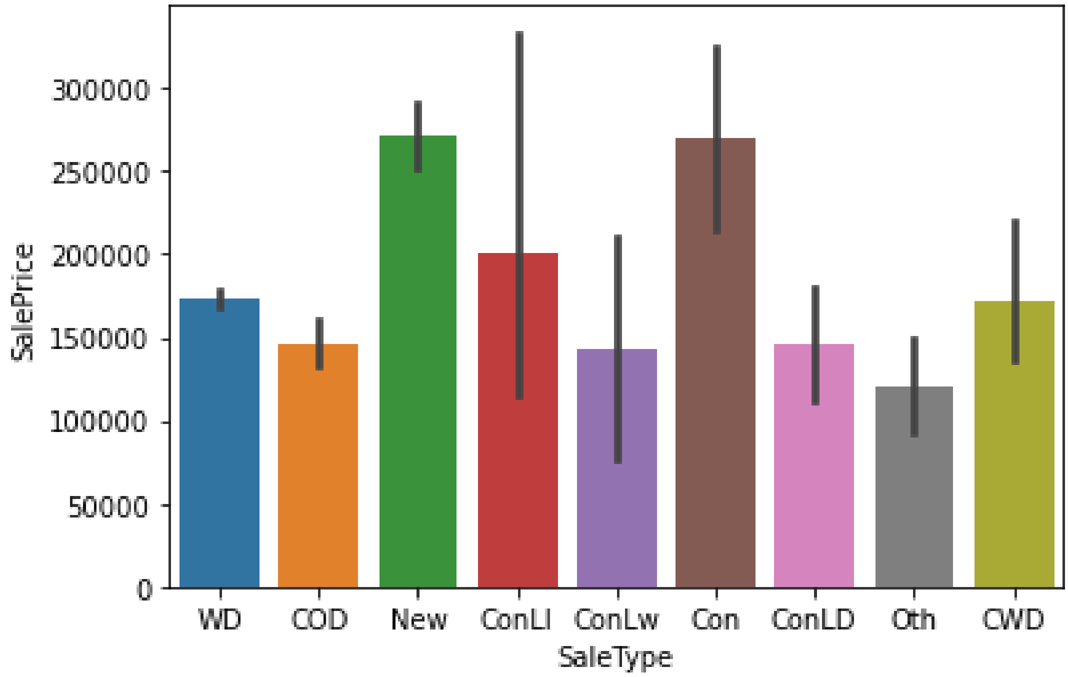
With Grade 9, 10 & 11 your sale price is going to be good

Year sold Vs Price



Whatever be the year sale price doesn't affect too much

Is saletype affect the sale price?



Yes, if your sale type is Con or New then definitely you are going to get a good price

Label Encoding & Correlation

As we have some categorical data we have to encoded those columns for machine learning model. We will use Label Encoder from sklearn.preprocessing.

We will describe the statistical summary of the dataset and find the correlation of each column.

In [52]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in test.columns:
    if test[i].dtypes=="object":
        test[i]=le.fit_transform(test[i].astype(str))
test.head()
```

Out[52]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condit
0	337	20	2	86.000000	14157	1	0	1	0	0	0	21	2	
1	1018	120	2	66.425101	5814	1	0	3	0	1	0	21	2	
2	929	20	2	66.425101	11838	1	3	3	0	4	0	4	2	
3	1148	70	2	75.000000	12000	1	3	0	0	4	0	5	2	
4	1227	60	2	86.000000	14598	1	0	3	0	1	0	20	1	

Statistical Summary

In [53]:
```python
train.describe()
```

Out[53]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Ne
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.0	1168.000000	1168.000000	1
mean	724.136130	56.767979	3.013699	70.988470	10484.749144	0.996575	1.938356	2.773973	0.0	3.004281	0.064212	
std	416.159877	41.940650	0.633120	22.437056	8957.442311	0.058445	1.412262	0.710027	0.0	1.642667	0.284088	
min	1.000000	20.000000	0.000000	21.000000	1300.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	
25%	360.500000	20.000000	3.000000	60.000000	7621.500000	1.000000	0.000000	3.000000	0.0	2.000000	0.000000	
50%	714.500000	50.000000	3.000000	70.988470	9522.500000	1.000000	3.000000	3.000000	0.0	4.000000	0.000000	
75%	1079.500000	70.000000	3.000000	79.250000	11515.500000	1.000000	3.000000	3.000000	0.0	4.000000	0.000000	
max	1460.000000	190.000000	4.000000	313.000000	164660.000000	1.000000	3.000000	3.000000	0.0	4.000000	2.000000	

Correlation

In [54]:
```python
corr=train.corr()
corr
```

GarageCond	-0.005130	-0.025595	-0.087375	0.043699	0.035657	-0.010973	-0.068449	0.005124	NaN	0.034690	-0.011606	0.(
PavedDrive	-0.009755	-0.068702	-0.077280	0.092551	0.021907	0.041318	-0.122756	0.111451	NaN	-0.034578	-0.012138	0.(
WoodDeckSF	-0.027498	-0.022609	-0.004509	0.088334	0.216720	-0.033142	-0.142202	-0.011580	NaN	-0.042424	0.089264	0.(

Removing the Outliers

In [59]:
```python
#seperating the dependent and independent varaibles
x=train.iloc[:,:-1]
y=train.iloc[:,-1]
```

Removing outliers

In [60]:
```python
from scipy.stats import zscore
z=np.abs(zscore(train))
threshold=3
print(np.where(z>3))
train_new=train[(z<3).all(axis=1)]
train=train_new
train.shape
```

```
(array([   1,    1,    1, ..., 1166, 1166, 1166], dtype=int64), array([ 9, 20, 34, ..., 39, 62, 63], dtype=int64))
```

Out[60]: (482, 76)

We have some outliers present in the dataset, so let’s handle them also. As the outliers in the dataset will affect our ML model. We need to remove all the outliers present in the dataset.

There is something called zscore which indicates how many standard deviations away an element is from the mean. We consider the points as outliers whose zscore is above 3 or less than -3. So we need to remove all such points from our dataset.

Using the threshold, we have removed all the points where the zscore is greater than 3. Now the total number of rows after removing the outliers are 721.

MODEL BUILDING

We will import important libraries for the building the ML model and defining the different models for our easiness.

Finding the best random state for the train test split.

Model Building

In [61]:
```
#importing the different machine learning models

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
```

In [62]:
```
# defining the different models

lg=LinearRegression()
rdr=RandomForestRegressor()
svr=SVR()
dtr=DecisionTreeRegressor()
knr=KNeighborsRegressor()
```

Finding the best random state

In [63]:
```
model=[lg,rdr,svr,dtr,knr]
maxAcc=0
maxRS=0
for i in range(40,60):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=.20)
    lg.fit(x_train,y_train)
    pred=lg.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print('Best Accuracy score is', maxAcc , 'on random state', maxRS)
```

```
Best Accuracy score is 0.878165525517784 on random state 49
```

In [64]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=49,test_size=.20)
```

Regression Algorithms

We have use five different regression algorithms to find the best model for our problem.

- **Linear Regression**
- ➢ from sklearn.linear_model import LinearRegression
- **Decision Tree Regressor**
- ➢ from sklearn.tree import DecisionTreeRegressor
- **Support Vector Regressor**
- ➢ from sklearn.svm import SVR
- **Kneighbor Regressor**
- ➢ from sklearn.neighbors import KNeighborsRegressor
- **Random Forest Regressor**
- ➢ from sklearn.ensemble import RandomForestRegressor

MODEL	ACCURACY
Linear Regression	0.878165525517784
Decision Tree Regressor	0.7819521438284093
Support Vector Regressor	-0.044480459746993
Kneighbor Regressor	0.6839964008801616
Random Forest Regressor	0.8815066126350098

Linear Regression

In [65]:
```
lg.fit(x_train,y_train)
pred1=lg.predict(x_test)
acc=r2_score(y_test,pred1)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.878165525517784
```

Decision Tree Regressor

In [66]:
```
dtr.fit(x_train,y_train)
pred2=dtr.predict(x_test)
acc=r2_score(y_test,pred2)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.7819521438284093
```

Support Vector Regressor

In [67]:
```
svr.fit(x_train,y_train)
pred3=svr.predict(x_test)
acc=r2_score(y_test,pred3)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  -0.04448045974699366
```

KNeighbor Regressor

In [68]:
```
knr.fit(x_train,y_train)
pred4=knr.predict(x_test)
acc=r2_score(y_test,pred4)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.6839964008801616
```

Random Forest Regressor

In [69]:
```
rdr.fit(x_train,y_train)
pred5=rdr.predict(x_test)
acc=r2_score(y_test,pred5)
print('Accuracy Score: ',acc)
```

```
Accuracy Score:  0.8815066126350098
```

Hence, we are getting the best accuracy score through the Random Forest Classifier Model. We will go ahead with this to find the cross val score and hypermeter tuning.

Cross Val Score & Hypermeter Tuning

Cross-validation provides information about how well a classifier generalizes, specifically the range of expected errors of the classifier. Cross Val Score tells how the model is generalized at a particular cross validation.

At CV=3 we get the best results i.e. the Random Forest Classifier more generalized at cv=3, so we calculate the hyper parameters at this value.

We will find which parameters of random forest classifier are the best foe our model. We will do this using Grid Search CV method & also calculate the accuracy score at those best parameters.

Cross Val Score

In [70]:
```
from sklearn.model_selection import cross_val_score
for i in range(3,7):
    cr=cross_val_score(rdr,x,y,cv=i)
    cr_mean=cr.mean()
    print("at cv= ", i)
    print('cross val score = ',cr_mean*100)
```

```
at cv=  3
cross val score =  85.23363749817996
at cv=  4
cross val score =  83.54462898149998
at cv=  5
cross val score =  84.56653146510307
at cv=  6
cross val score =  84.9353546893834
```

Hypermeter Tuning

In [71]:
```
from sklearn.model_selection import GridSearchCV
# creating parameters
para={'criterion':['squared_error','absolute_error','poisson'],
      'max_features':['sqrt','log2'],
      'max_depth':[1,2,3,4,5]}

GCV=GridSearchCV(rdr,para,cv=3,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

Out[71]:
```
{'criterion': 'squared_error', 'max_depth': 1, 'max_features': 'sqrt'}
```

In [72]:
```
GCV_pred=GCV.best_estimator_.predict(x_test)
r2_score(y_test,GCV_pred)
```

Out[72]:
```
0.5074682635895407
```

Saving the Model

Saving the best model – Random Forest Classifier in this case for future predictions. Let’s see what are the actual test data and what our model predicts.

Saving the model

In [74]:
```
import pickle
filename='house_price.pkl'
pickle.dump(rdr, open(filename,'wb'))
```

Conclusion

In [75]:
```
a=np.array(y_test)
pred=np.array(pred5)
Sale_Price=pd.DataFrame({'Actual':a,'Predicted':pred})
Sale_Price
```

17	136000	163126.48
18	485000	466702.81
19	108000	118654.50
20	143000	139013.80
21	253293	383934.62
22	155000	145269.50
23	227680	217334.05
24	102000	114376.51
25	212000	232302.71
26	163000	183058.18
27	116000	113933.02
28	269790	221657.25
29	135000	130609.74

Hence up to some good extensions our model predicted so well.

Now, what our model predict for test dataset?

With the best model that we have saved earlier, let's predict the sale price of the houses.

Loading the model for prediction

In [76]:
```
loaded_model = pickle.load(open(filename, 'rb'))
pred=loaded_model.predict(test)
pred
```

Out[76]:
```
array([365295.45, 226427.19, 247342.51, 168327.15, 198815.73,  82712.83,
       137777.39, 324962.73, 230147.25, 166520.48,  73989.08, 148594.92,
       120311.3 , 182975.13, 335520.44, 126949.83, 119323.27, 126761.74,
       165997.19, 196860.35, 162259.32, 147796.47, 147463.89,  73903.08,
       101088.94, 130068.27, 179357.17, 147927.41, 163529.97, 111576.92,
       150977.49, 180730.  , 232938.66, 161070.32, 104477.21, 166374.85,
       190619.78, 110942.76, 156020.31, 149300.32, 101158.08, 330364.56,
       197277.65, 184837.13, 127739.65, 133892.83, 122370.74,  92606.33,
       207494.62, 337328.65, 148877.81, 186679.  , 101795.  ,  94496.16,
       267731.36, 108796.5 , 146216.63, 184822.79, 108200.53, 255760.86,
        95328.  , 164240.28, 131616.26, 143728.33, 194539.87,  90089.33,
       150369.12, 202426.35, 133745.8 , 160733.24, 312643.52, 147338.65,
       183489.19, 155555.21, 140804.2 , 236595.04, 321680.95, 200872.1 ,
       295752.7 , 142658.5 , 215210.94, 140181.29, 142735.87, 155667.14,
       176494.84, 250359.07, 104706.73, 382913.98, 158011.89, 177961.13,
       237112.34, 126750.49, 140426.6 , 117649.61, 182513.3 , 159202.98,
       248993.37, 171123.98, 325389.76, 123384.25, 267595.62,  90476.5 ,
       110869.27, 148380.72, 197202.3 , 146011.48, 264506.47, 137876.55,
       182218.09, 200900.8 , 174209.77, 169235.  , 243074.96, 222422.89,
       126112.52, 110123.14, 131700.48, 194296.55, 140434.78, 104177.89,
        90329.66, 193834.46, 274589.68, 138277.93, 147515.63, 188305.9 ,
       124802.81, 164200.74,  84101.65, 110783.19, 138770.42, 222746.74,
       138469.55, 156398.39, 185228.7 , 290929.29, 200956.03, 118638.5 ,
       289402.43, 112749.26, 145221.08, 444623.04,  87546.76, 382672.11,
       181342.75, 235430.13, 176155.38, 128257.94, 103250.92, 197662.76,
       142113.84, 134627.  , 179813.32, 108390.15,  98459.26, 168691.  ,
       184141.5 , 173609.7 , 126865.71, 162305.71, 200646.89, 144286.74,
       194988.96, 113482.74, 113629.58, 237868.81, 201741.64, 182821.62,
       129918.98, 228808.41, 145210.57, 122357.19, 129057.81, 278459.88,
       134944.54, 368109.21, 135282.26, 111301.26, 146121.29, 146374.52,
       207439.3 , 151880.5 , 250929.12, 165543.81, 423278.72, 357188.84,
       215888.59,  95339.14, 168959.41, 153532.16, 111552.04, 223037.29,
       185439.5 ,  81655.15, 135241.22,  89355.03, 173852.22, 190537.7 ,
       120808.24, 164304.66, 143440.  , 101046.93, 256921.14, 297912.23,
       131024.67, 124577.  , 231932.2 , 152621.02, 130937.32, 157592.71,
       103831.18, 169129.  , 148496.5 , 193674.96, 100456.32, 259945.06,
       145057.33, 134707.07, 121448.83, 173718.48, 207706.74, 217139.38,
       248562.92, 131360.09, 177854.  , 334366.02, 223228.25,  95805.99,
       369782.26, 180656.28, 116106.82, 150128.  ,  81599.42, 133420.31,
       112298.71, 205239.25, 178174.37, 158876.29, 250356.74, 180383.05,
       223980.61, 147128.33, 374093.18,  94629.47, 138787.39, 235783.93,
       185753.54, 267542.08, 148799.89, 146066.27, 156711.62, 230277.21,
       168372.84, 116176.11, 101123.44, 202254.39, 127875.55, 381486.18,
       184229.94, 118437.91, 177909.4 , 186068.  , 101247.21, 132037.94,
       224418.3 , 163149.34, 103978.64, 175070.14, 264999.92, 228751.1 ,
       163643.76, 135959.3 , 330974.72, 264112.16, 301618.22, 198415.29,
```

CONCLUSION

Conclusion of the Study

The results of this study suggest following outputs which might be useful for the company to enter into the Australian Market:

- There are lot of things that is going to decide the sale price of a house. As we see above in our visualizations, a lot of things affect the price like neighbourhood, Quality condition, basement, living area, roof type, building type and many more. One needs to analyse every aspect to have good hands on the prediction of the price.
- With the machine learning it become easier to predict the price but yes it is not 100% accurate, it provides an idea and accordingly we can analyse the market and prepare the strategies to grab the opportunities.

- ## Learning Outcomes of the Study in respect of Data Science
  - I got to know the different factors required for the price prediction of a house.
  - It was fun to deal with this project and learn how we can use our saved model to predict the price for given dataset.
  - It was difficult to handle so much columns simultaneously but yes every difficulty learns the new things to us.