FLIP ROBO

HOUSING PRICE PREDICTION PROJECT

Submitted by:

JASMINE FERNANDES

ACKNOWLEDGMENT

Following are the acknowledgement sources:

My lecture notes and codes

You tube

INTRODUCTION

- ## Business Problem Framing

  Real estate is not a very straight forward industry, especially when entering a new market in a new country. The challenge is to purchase houses at a price below their actual values and resell them at a higher price. Before investing a large sum of capital, the aim is to perform data analysis on the collected data set from the sale of houses in Australia to make an informed decision related to the housing prices.

- ## Conceptual Background of the Domain Problem

  The company is finding perspective properties to enter the market. The aim is to predict the actual value of the prospective properties and decide whether to invest in them or not.

- ## Motivation for the Problem Undertaken

  The motive behind this analysis to pick the right properties for investing In order to do so, the objective is to create an effective machine learning model that is able to accurately estimate the price of the house based on given features given and use these insights for investing. The purpose of this the report is to find the best method to predict with real time factors that impact the housing industry.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

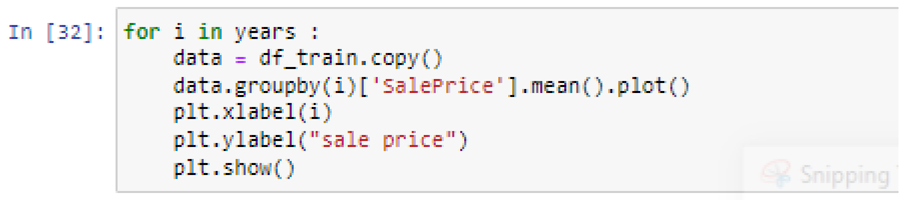
  Linear regression modelling is used as we are dealing with a continuous variable. Various techniques to build models are used: Random Forest, Linear regression, Decision Tee and Adaboost. The main aim of using Linear Regression model find the best fit linear line such that the error is minimized. Error is the difference between the actual value and Predicted value.

- Data Sources and their formats

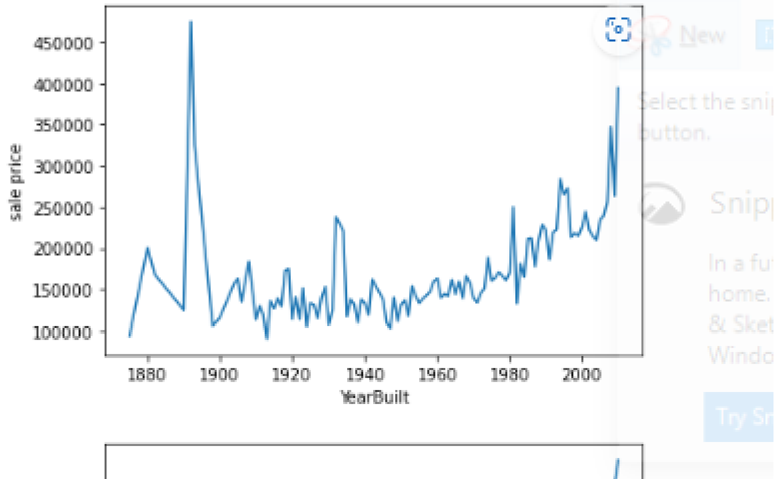
  Following formats are used:

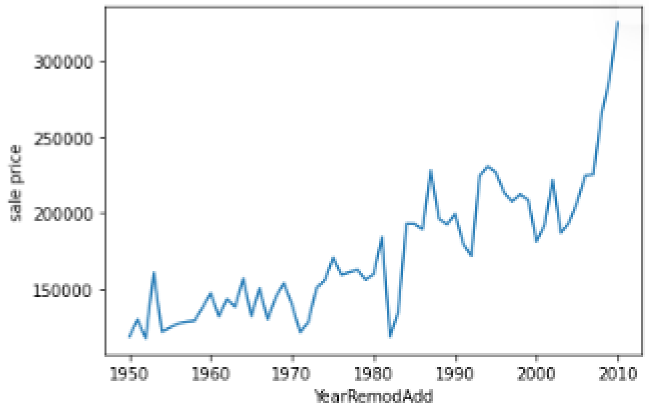
  ✔ Scatter plots
  ✔ Correlation plot
  ✔ Skewness table
  ✔ Used histogram to view distribution of all the numeric variables

  All data related to 'year' is separated to find their correlation with Sales price



```
In [32]: for i in years :
             data = df_train.copy()
             data.groupby(i)['SalePrice'].mean().plot()
             plt.xlabel(i)
             plt.ylabel("sale price")
             plt.show()
```





- Data Preprocessing Done

  Following steps are used for data cleaning:

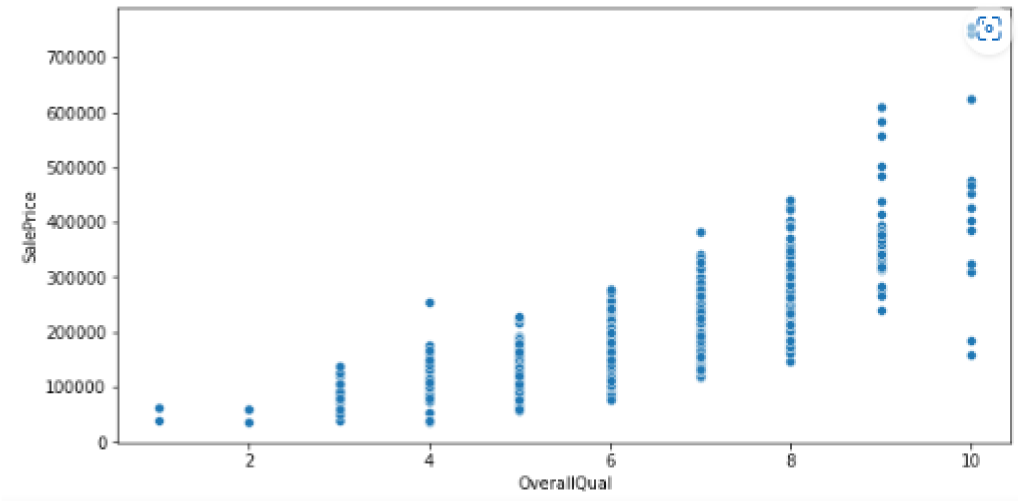
  1. Dropped the ‘cust id’ from the test data and train data as it does not impact the outcome. It is an unnecessary detail.
  2. Searched for Nan values with isna().sum() function. LotFrontage has imbalanced data
  3. Searched for duplicate columns, but few columns had many missing values and data would be lost(data set is as it is small). Hence other methods were used
  4. Finding if there are any unique values in object data types and separating object type columns

5. Did data cleaning with both, **train and test data** and balanced it to remove any nulls or missing data
6. Added numerical values to the object type columns for modelling
7. Viewing distribution with histogram for numerical columns
8. Separating float and int64 values and checking for any missing value
9. #GarageYrBlt , LotFrontage have missing values, we added mode and mean method
10. Checked for any missing data pattern
11. Segregated all year columns - 'YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold' and tried to vew their relationship with Sales price
12. Used correlation matrix to find correlations

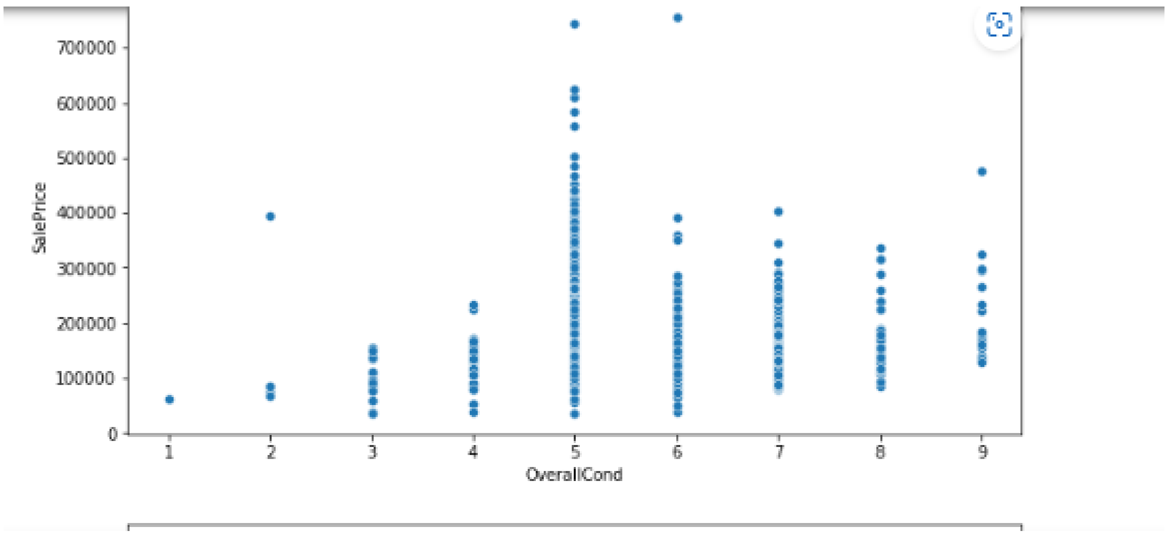
- ## Data Inputs- Logic- Output Relationships
- With the following code , I established relations between all object type data and target column – 'Sales Price':

```
for col in df_train.columns:
if df_train[col].dtype != 'object':
plt.figure(figsize=(12,6))
sns.scatterplot(x=col,y='SalePrice',data=df_train)
```

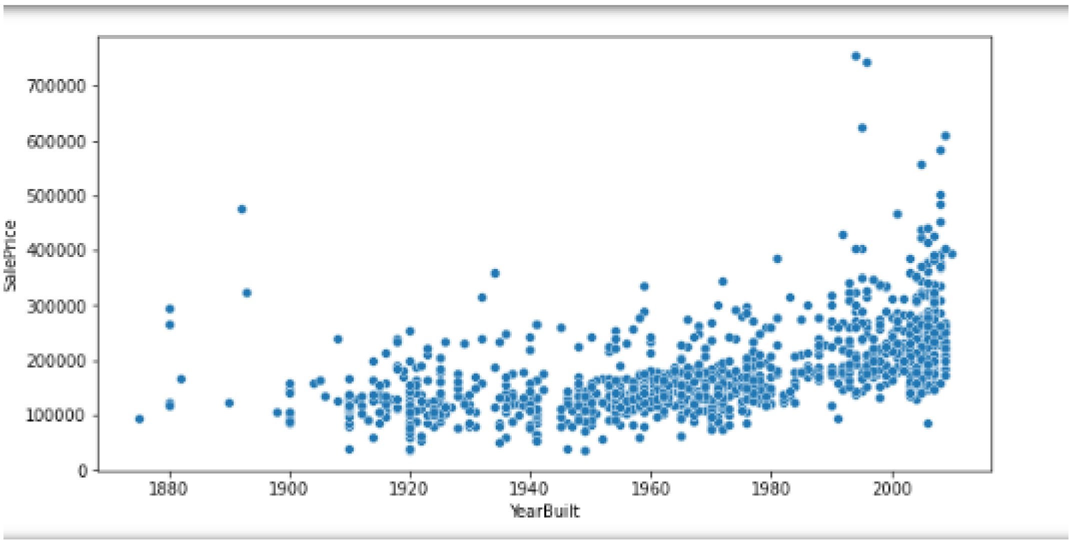
- Sales Price and OverallQual(Rates the overall material and finish of the house) : Sales price is highest for 9-10 (10 stands for excellent) –



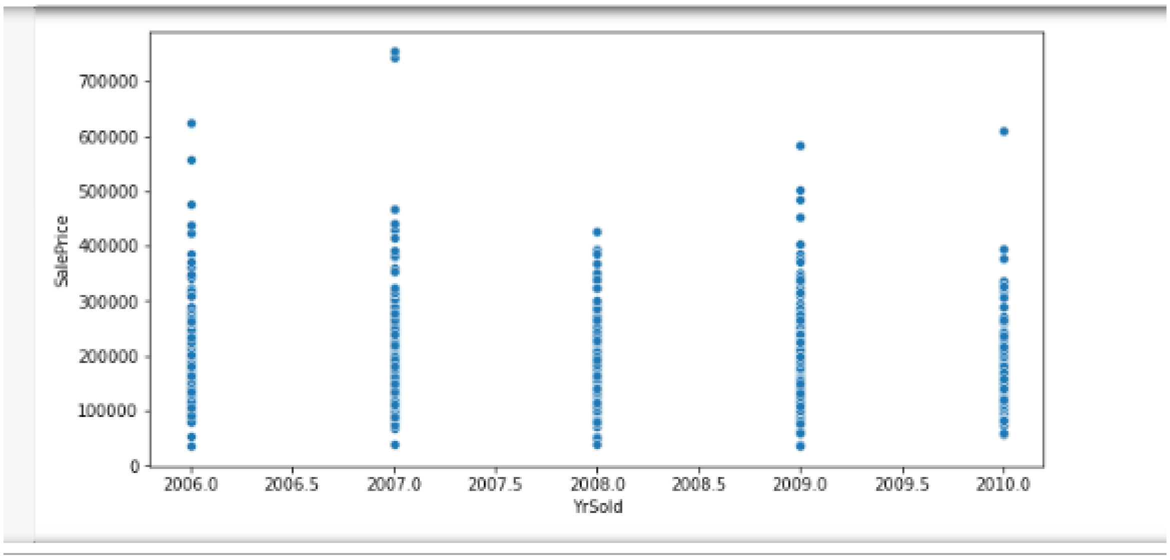
- Sales Price vs OverallCond(Rates the overall condition of the house): The sales price is highest for 5, which is average



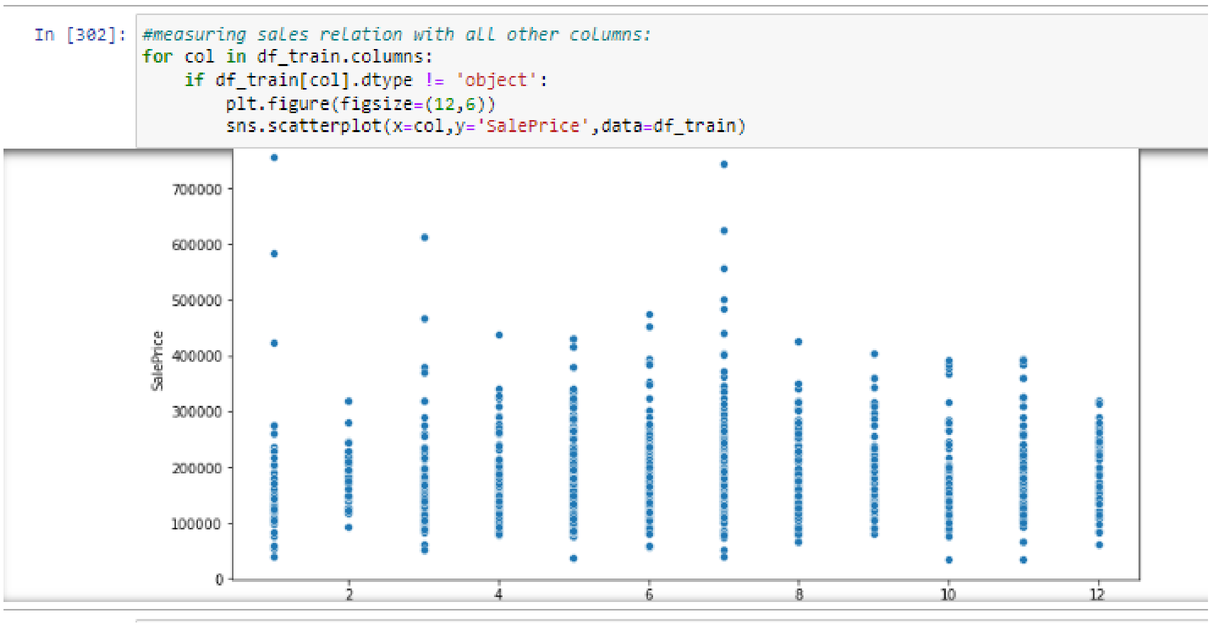
- Sales Price vs YearBuilt(Original construction date): The year 1997-98 has the highest sales price



- Sales Price vs YearRemodAdd: Remodel date (same as construction date if no remodeling or additions) - The year 1997-98 has the highest sales price
- Sales Price vs YrSold(Year Sold (YYYY): - The year 2007 has the highest sales price

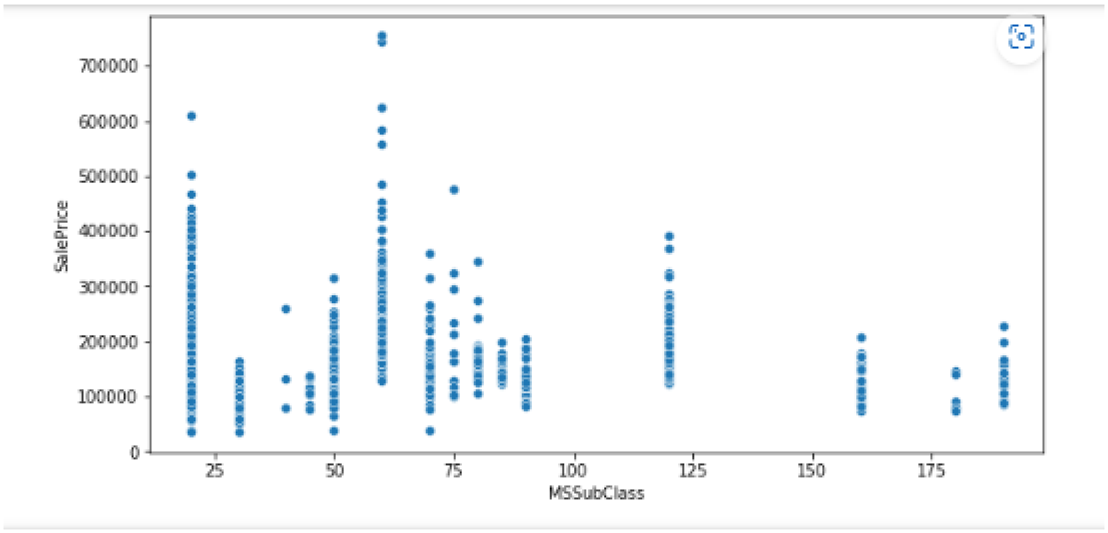


- Sales Price vs Month Sold (MM)): - The month July, 7th month has the highest sales price

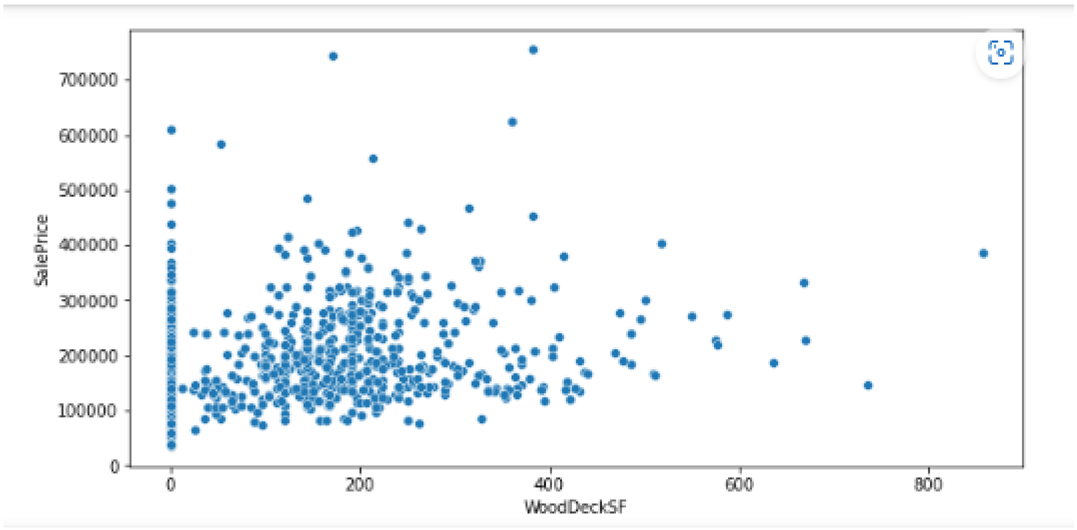


```
In [302]: #measuring sales relation with all other columns:
          for col in df_train.columns:
              if df_train[col].dtype != 'object':
                  plt.figure(figsize=(12,6))
                  sns.scatterplot(x=col,y='SalePrice',data=df_train)
```

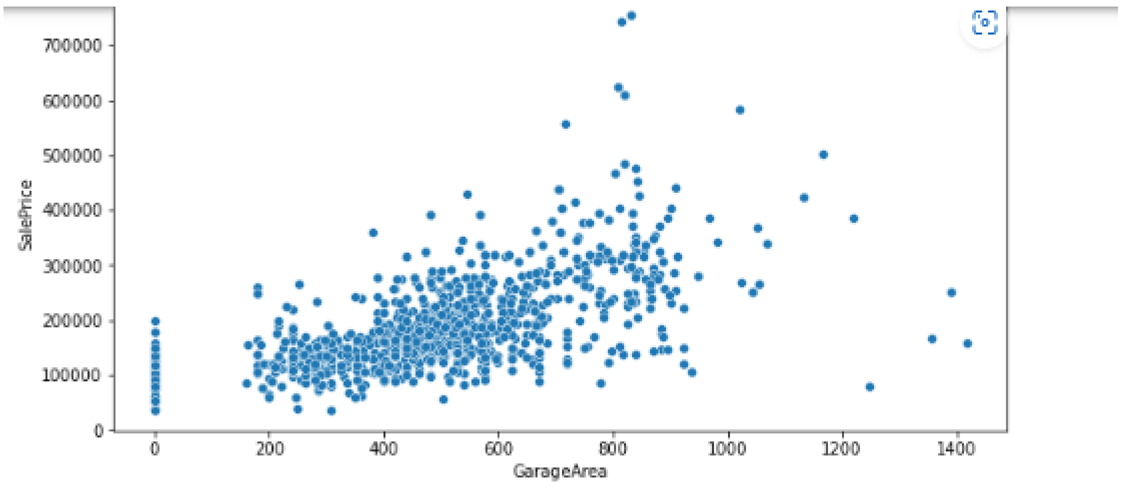
- Sales Price vs MSSubClass (Identifies the type of dwelling involved in the sale.)): - 60 has the highest sales price. 60 stands for 2-STORY 1946 & NEWER



- Sales Price vs **WoodDeckSF**(**Wood deck area in square feet)**:
  - 380-400 sq ft has the highest sales price

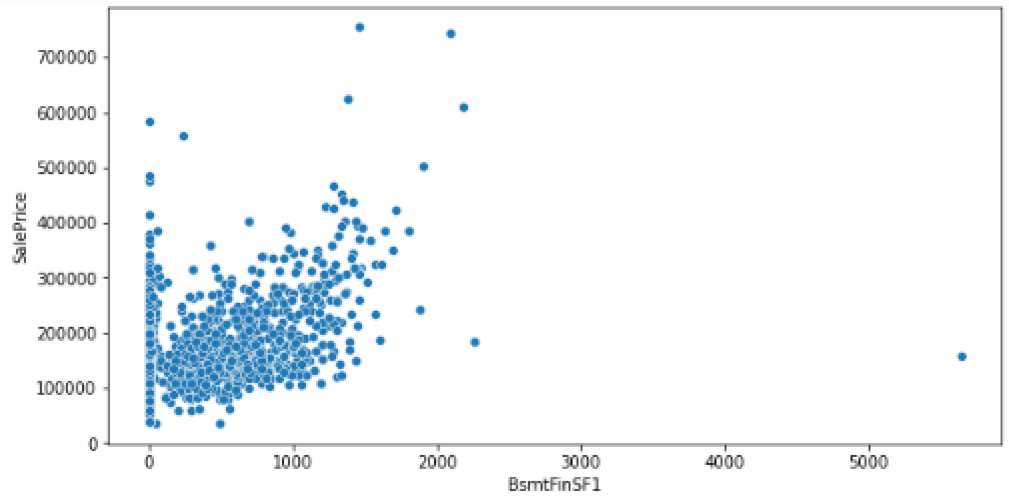


- Sales Price vs Garage area: 800-820 sq ft has the highest sales price

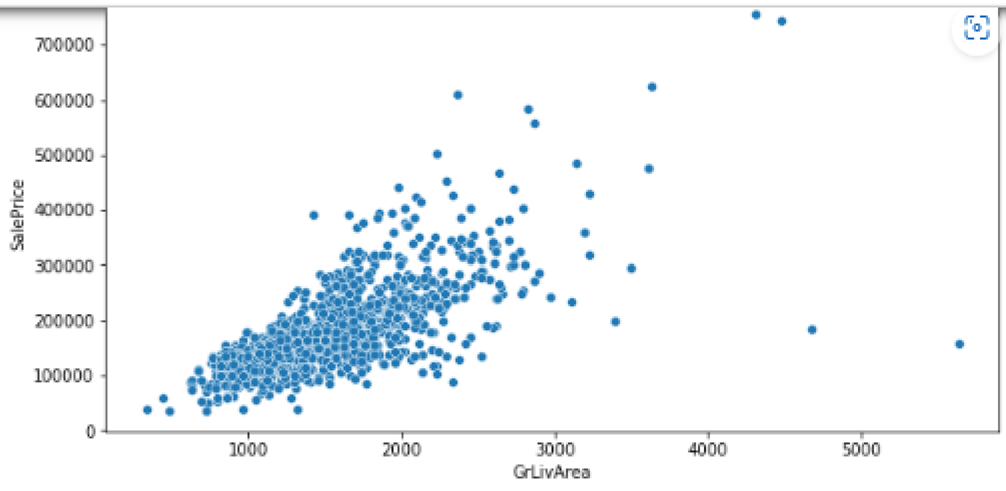


- Sales Price vs TotalBsmtSF-Total square feet of basement area: majority of the basement area available is between **0-1000 sqft**

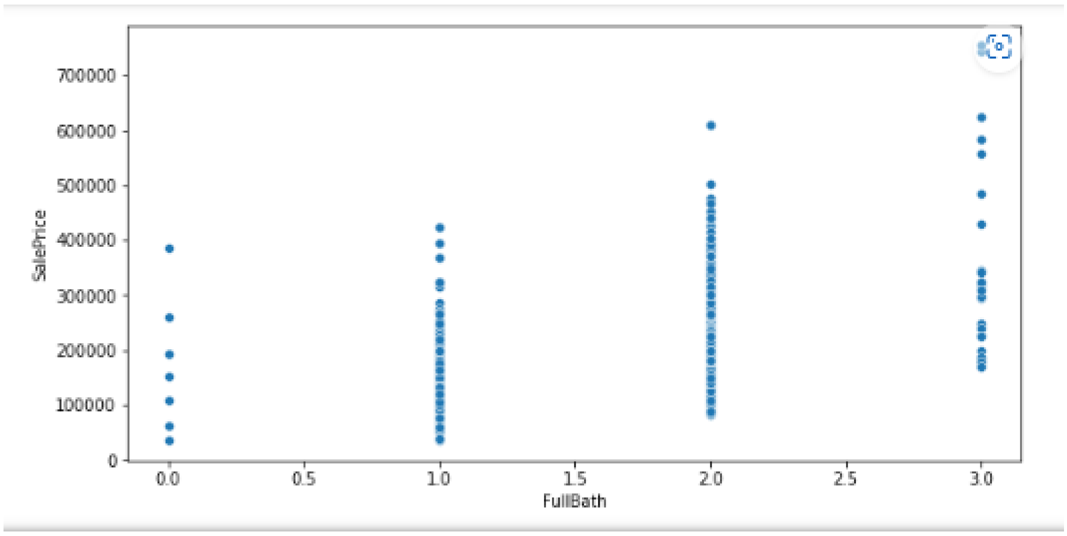
```
]:  #measuring sales relation with all other columns:
    for col in df_train.columns:
        if df_train[col].dtype != 'object':
            plt.figure(figsize=(10,5))
            sns.scatterplot(x=col,y='SalePrice',data=df_train)
```



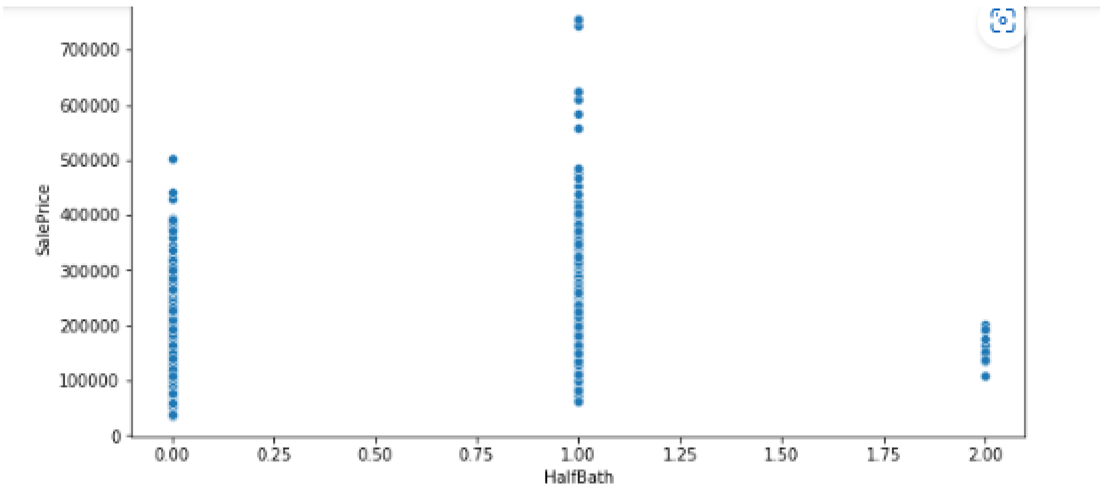
- Sales Price vs GrLivArea(Above grade (ground) living area square feet)-majority of the basement area available is between **1000-2000 sqft and costs between 100000-300000**



- Sales Price vs FullBath(Basement full bathrooms)-The category 3- full bath basement bathrooms is sold for the highest price

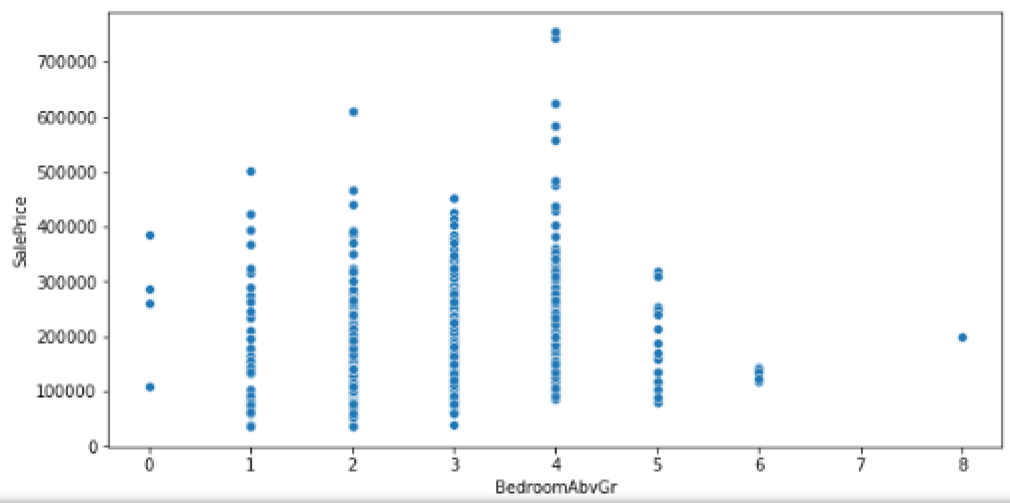


- Sales Price vs HalfBath(Basement half bathrooms)-The ‘1’ category half bath basement bathrooms is sold for the highest price
-



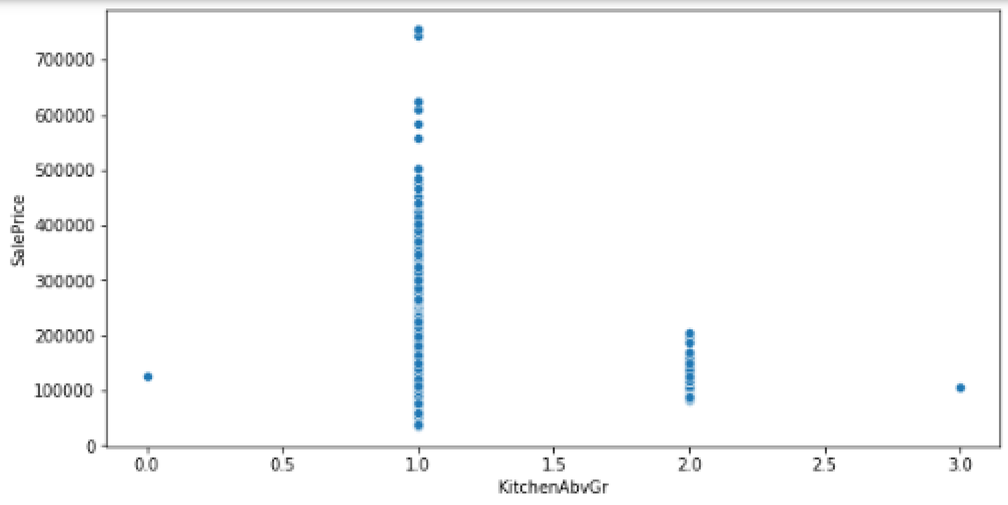
- Sales Price vs BedroomAbvGr:The ‘4’ category is sold for the highest price

```
#measuring sales relation with all other columns:
for col in df_train.columns:
    if df_train[col].dtype != 'object':
        plt.figure(figsize=(10,5))
        sns.scatterplot(x=col,y='SalePrice',data=df_train)
```

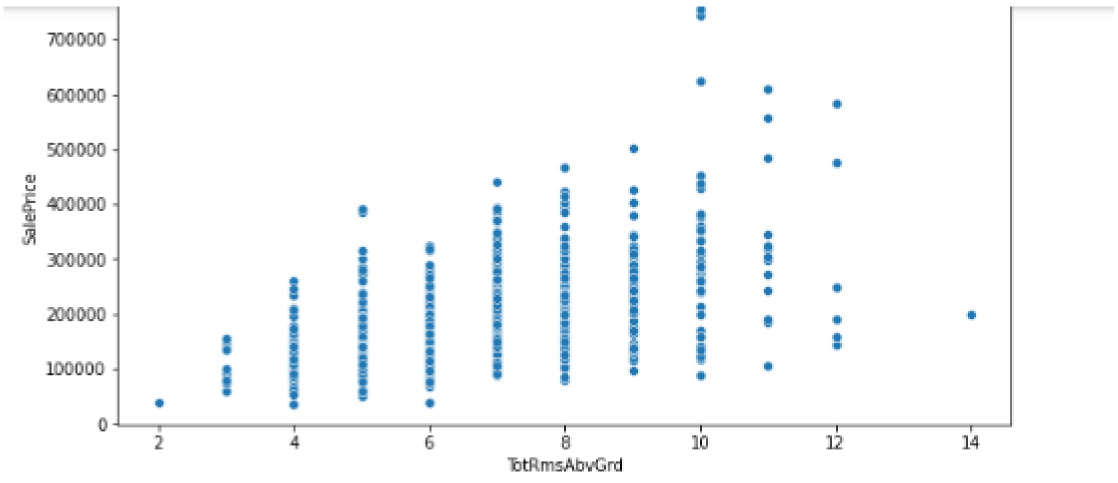


- Sales Price vs KitchenAbvGr:The ‘1’ category is sold for the highest sales price

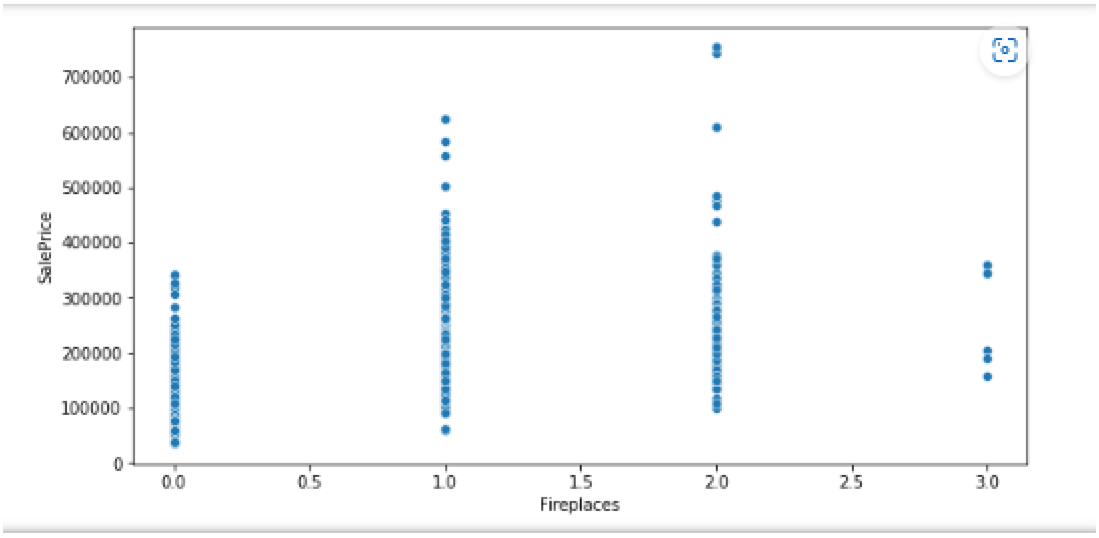
```
#measuring sales relation with all other columns:
for col in df_train.columns:
    if df_train[col].dtype != 'object':
        plt.figure(figsize=(10,5))
        sns.scatterplot(x=col,y='SalePrice',data=df_train)
```



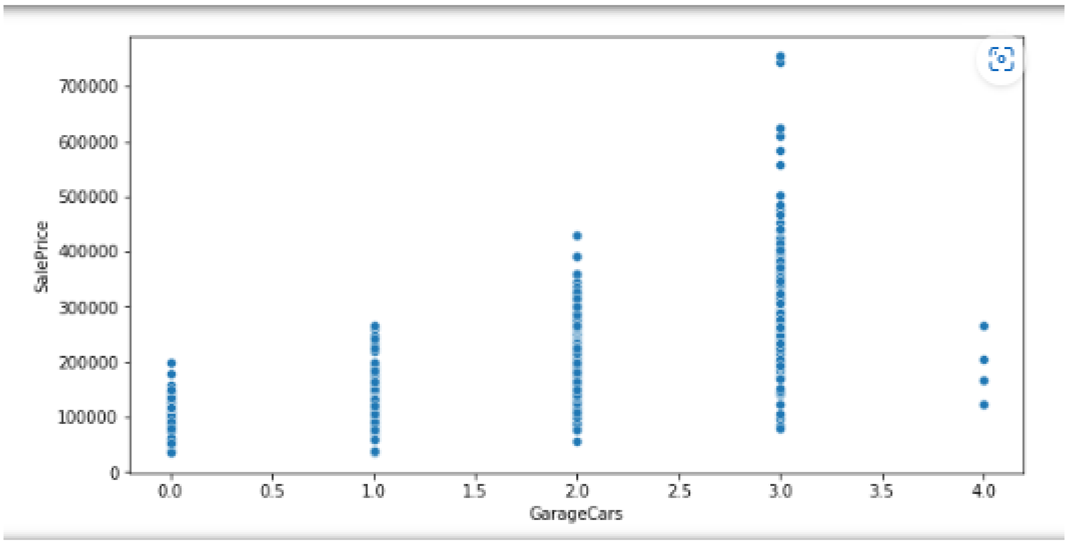
- Sales Price vs TotRmsAbvGr:The ‘10’ category is sold for the highest sales price



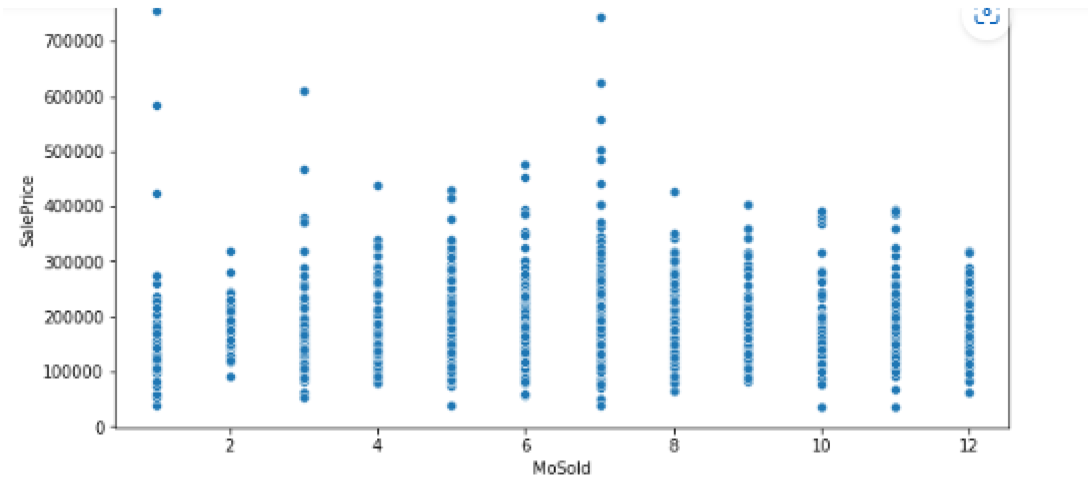
- Sales Price vs Fireplaces: The ‘2’ category is sold for the highest sales price



- Sales Price vs GarageCars: The ‘2’ category is sold for the highest sales price

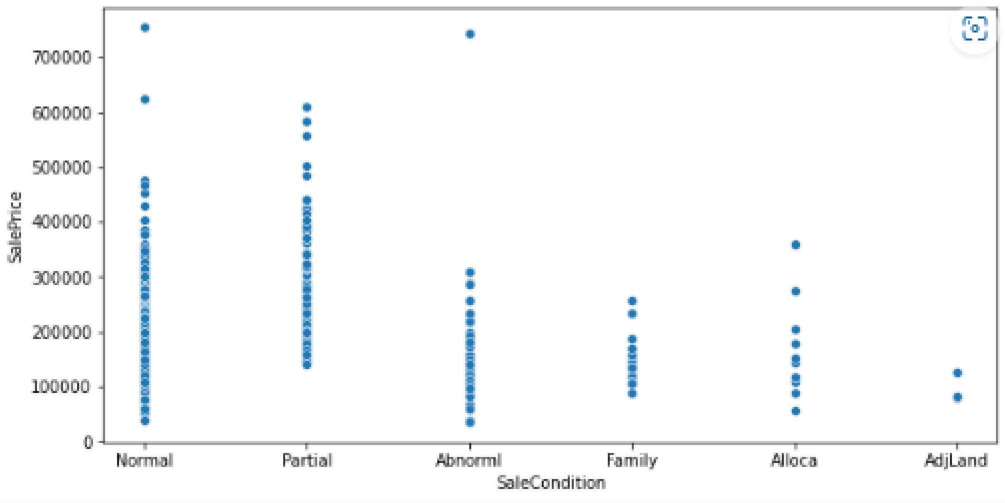


- Sales Price vs MoSold(month sold): The ‘7th ’ category month has the highest sales price

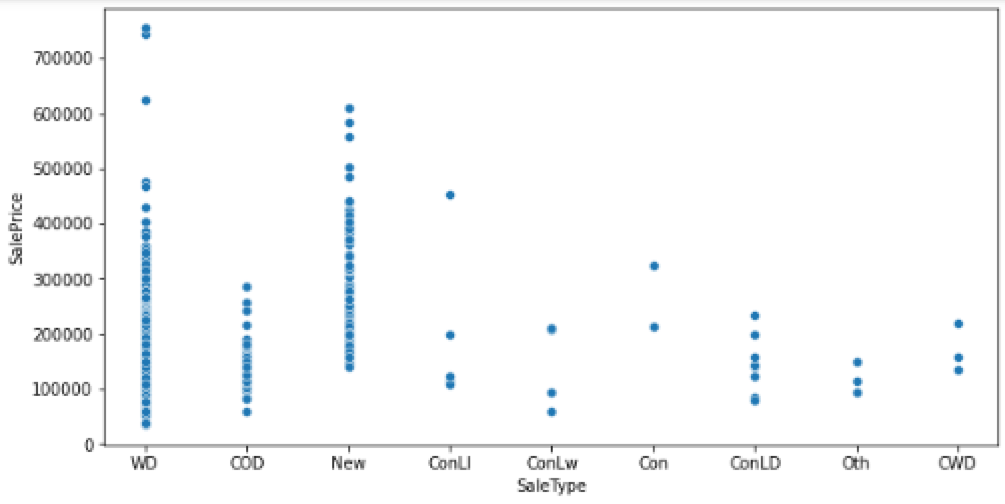


RELATIONS BETWEEN NUMERICAL COLUMNS & SALES PRICE

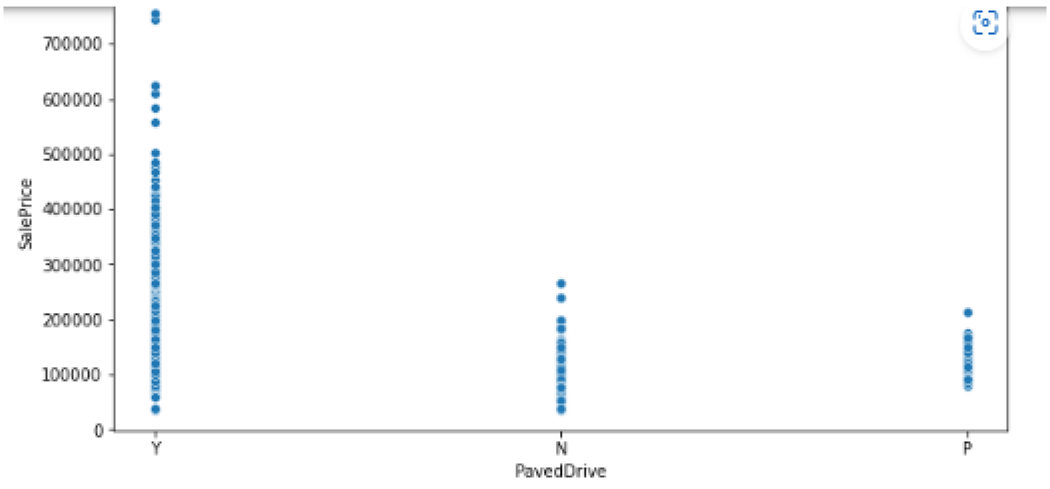
1. Sales Price vs Sales Condition: The normal and partial category fetches the highest sales price



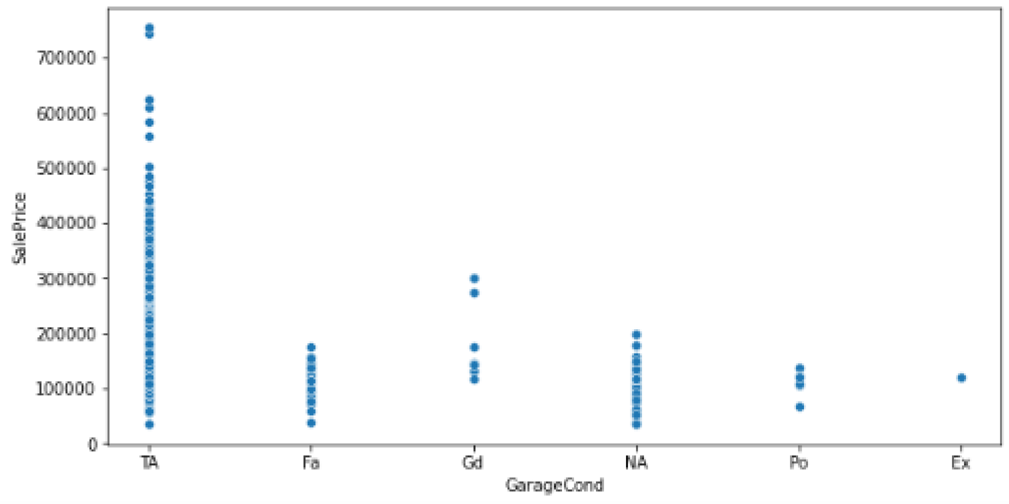
2. Sales Price vs Sales Type: WD(Warranty Deed – Conventional) fetches the highest sales price



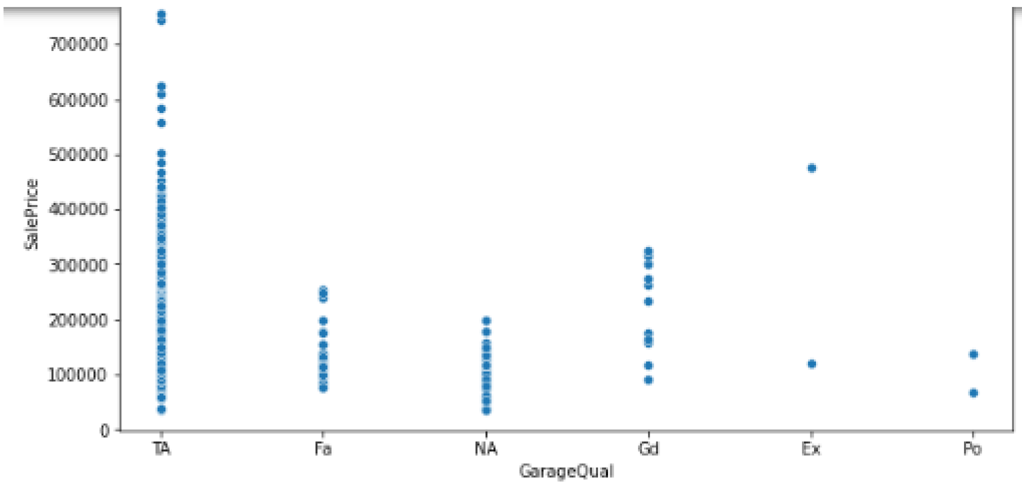
3. Sales Price vs Paved Drive: Paved fetches the highest sales price



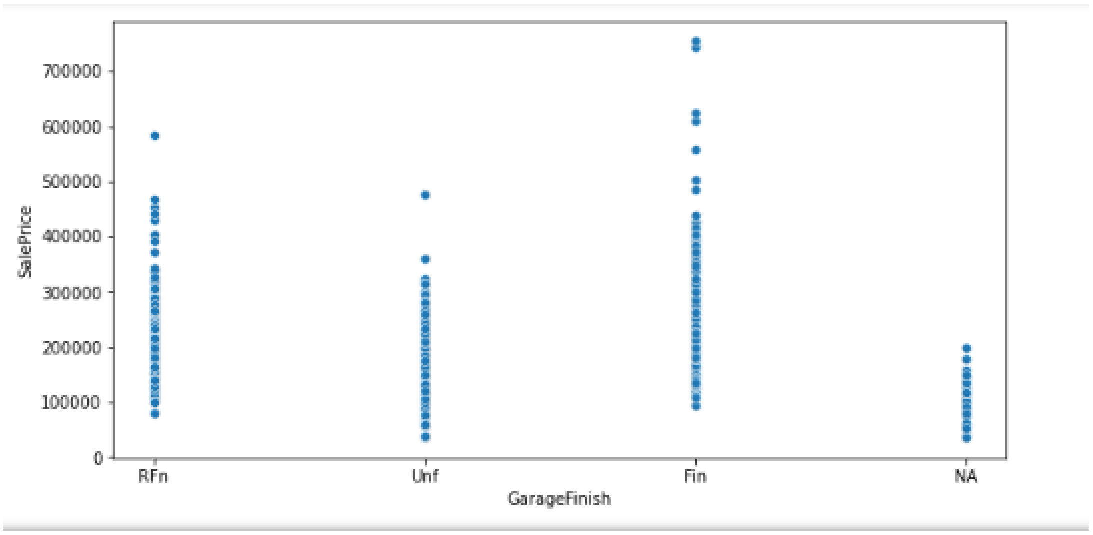
4. Sales Price vs GarageCond: TA fetches the highest sales price



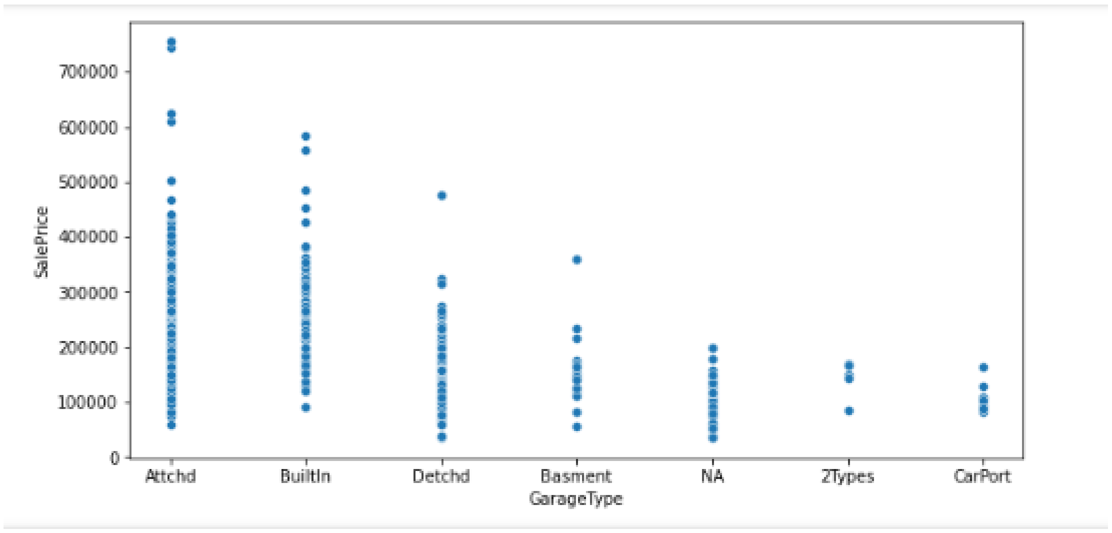
5. Sales Price vs GarageQual: TA fetches the highest sales price



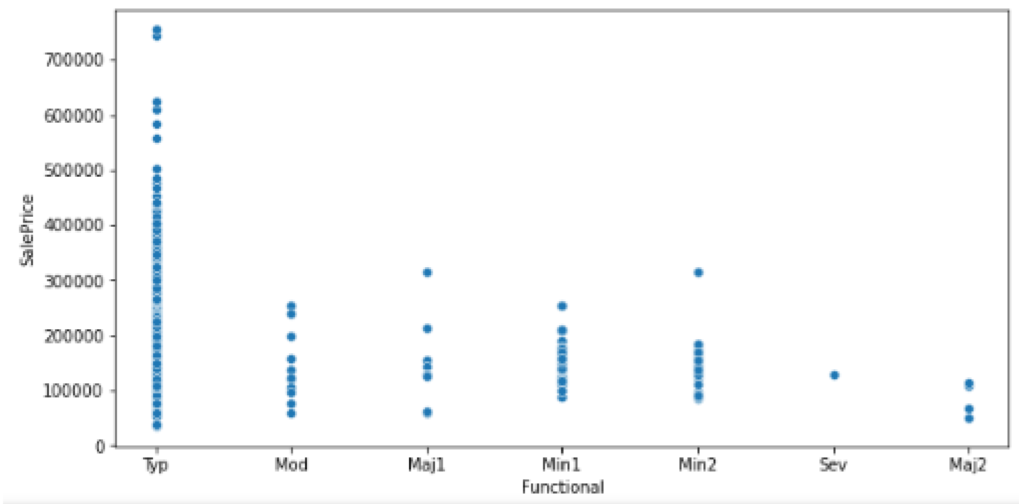
6. Sales Price vs GarageFinish: Fin fetches the highest sales price



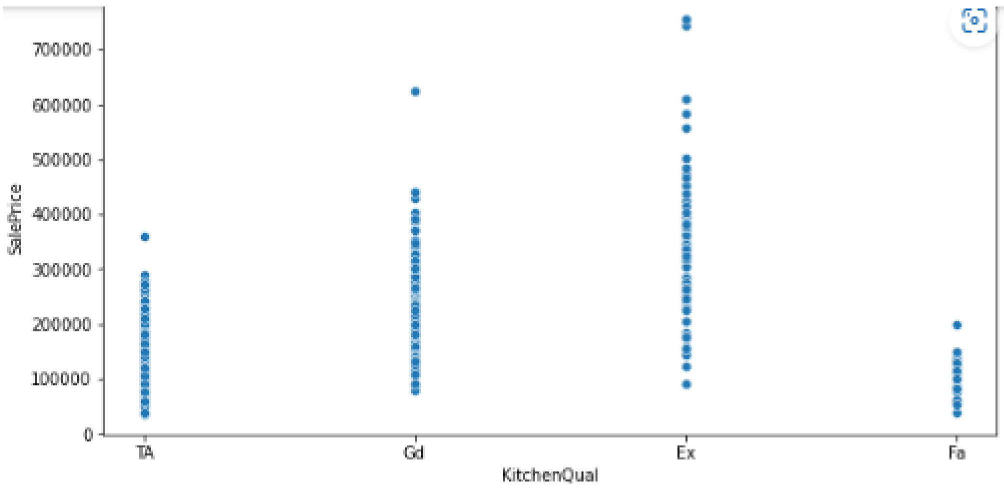
7. Sales Price vs GarageType: Attchd fetches the highest sales price



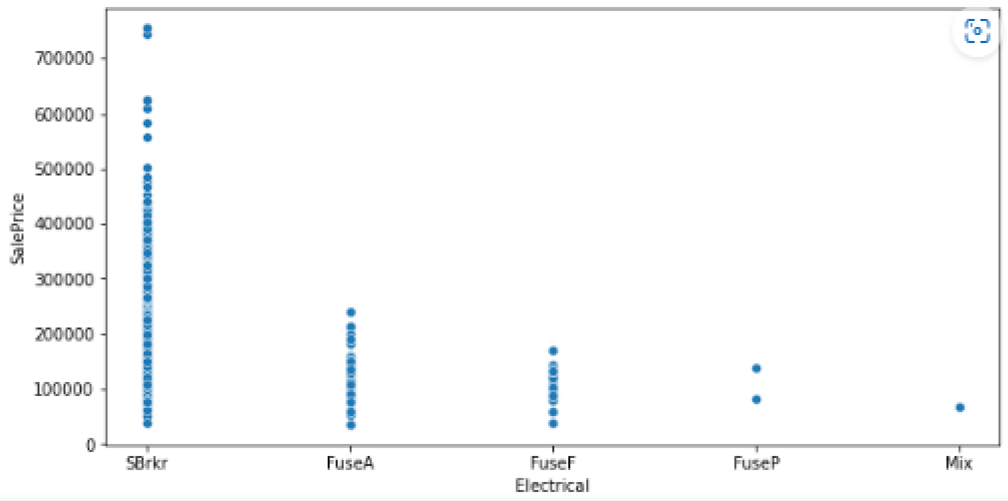
8. Sales Price vs Functional : Typ fetches the highest sales price



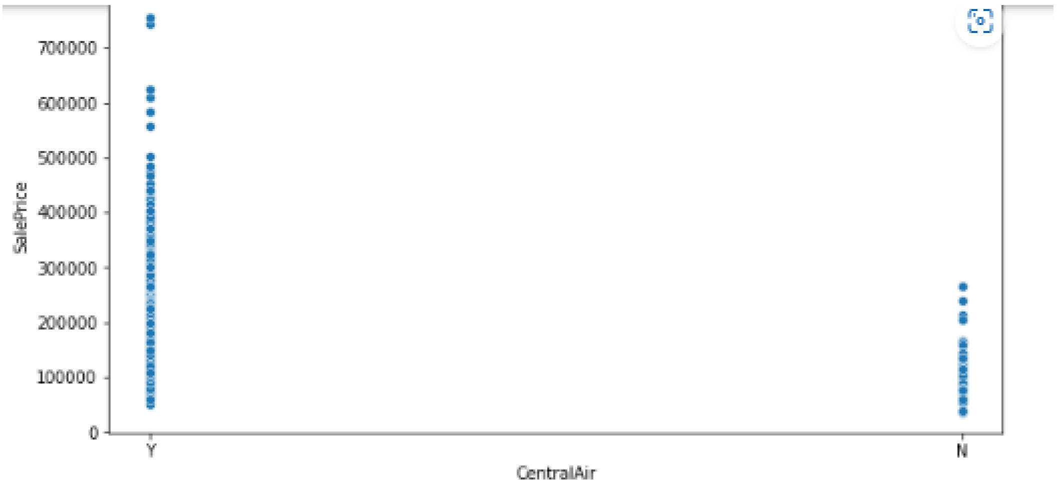
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
Functional

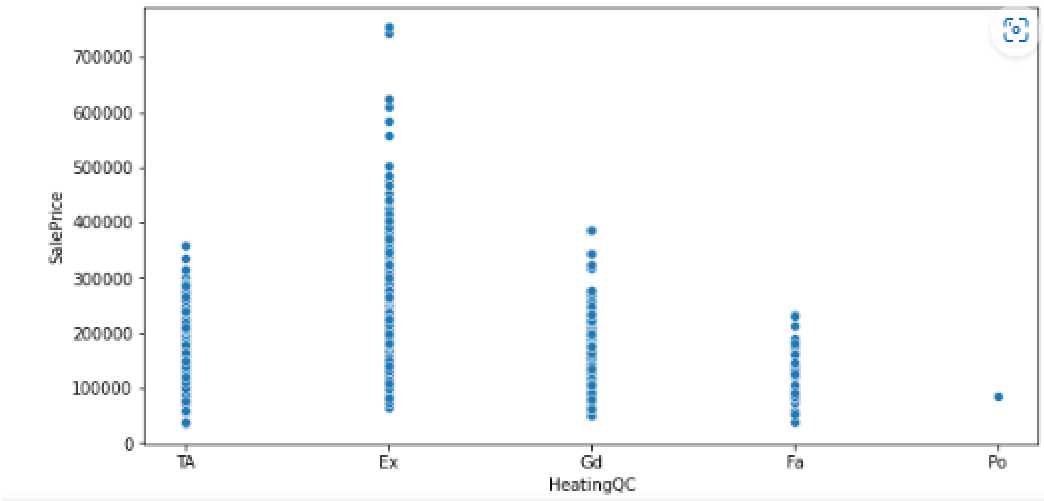


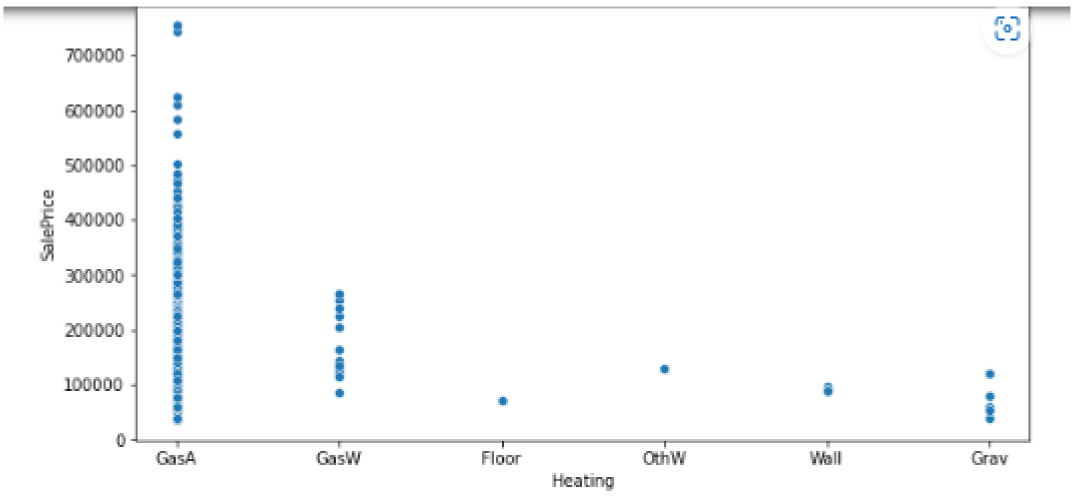
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
TA
Gd
Ex
Fa
KitchenQual

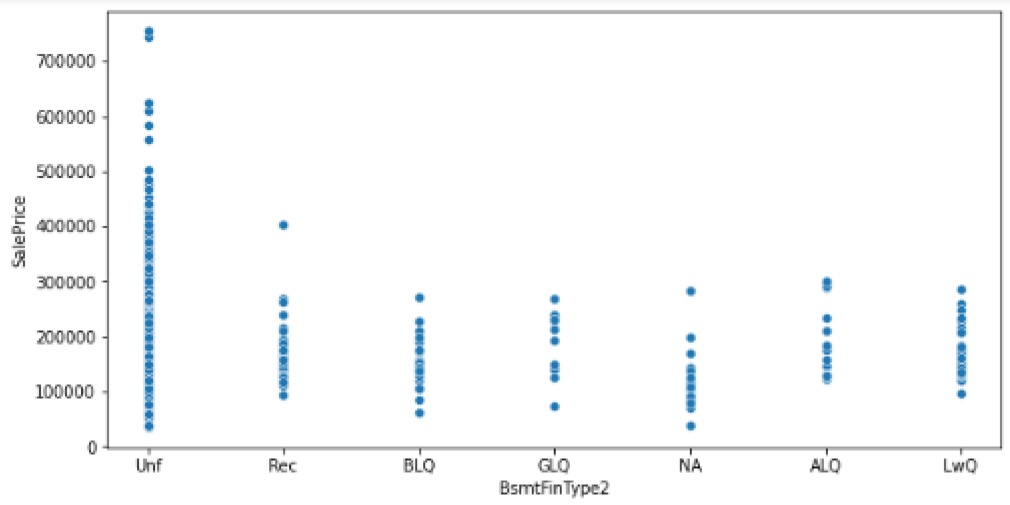


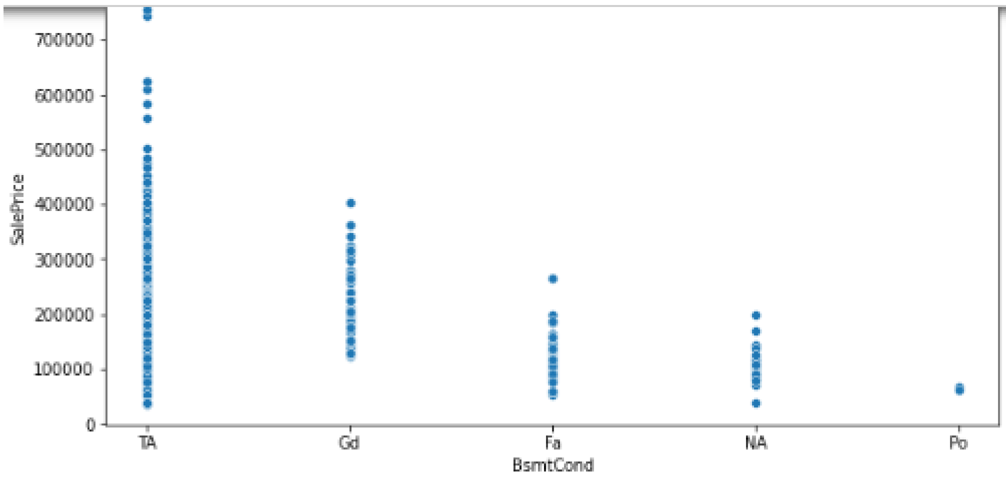
700000
600000
500000
400000
300000
200000
100000
0
SalePrice
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

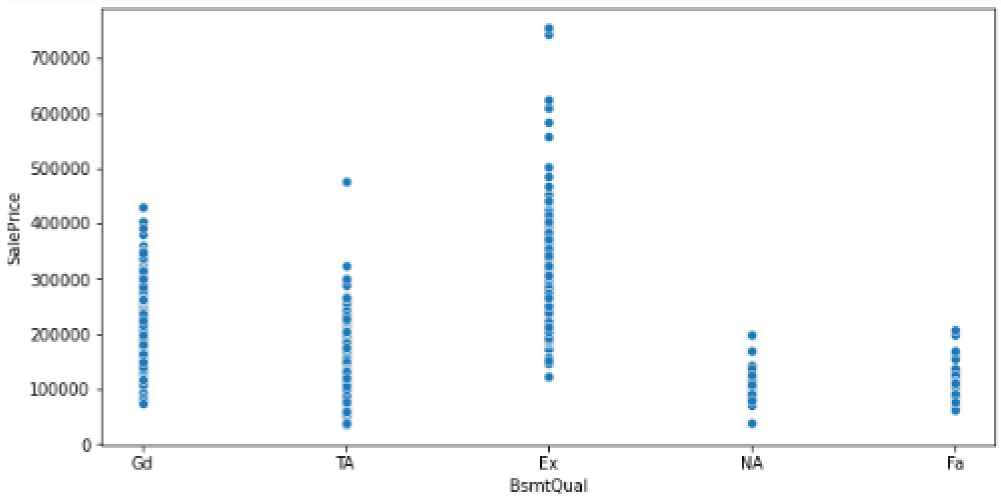

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Y
N
CentralAir

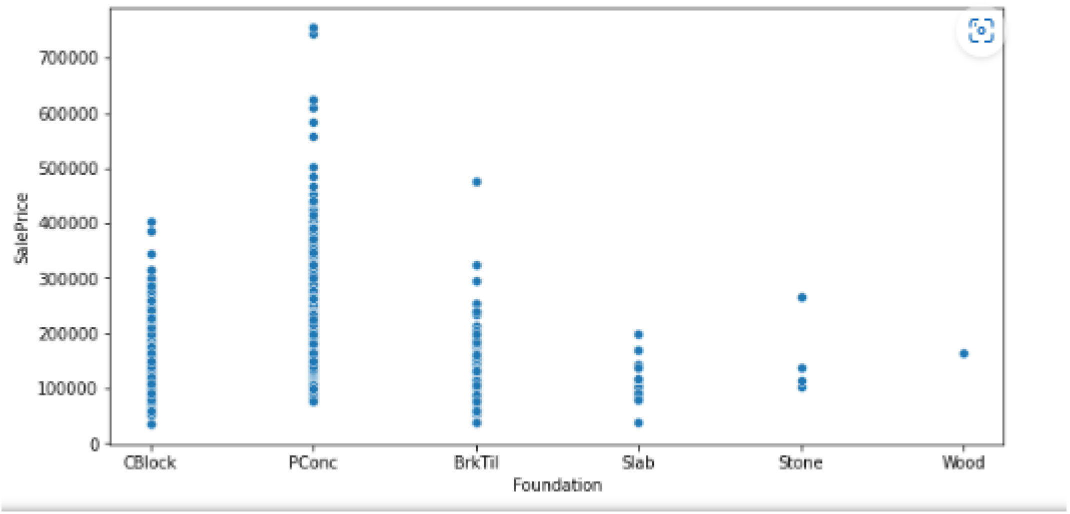

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Ex
Gd
Fa
Po
HeatingQC


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
GasA
GasW
Floor
OthW
Wall
Grav
Heating

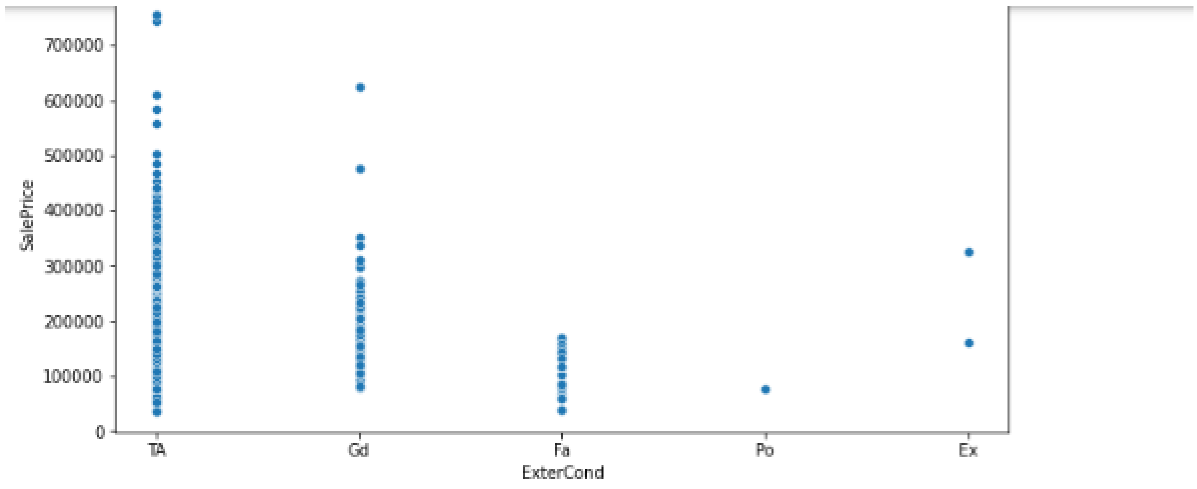

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Unf
Rec
BLQ
GLQ
NA
ALQ
LwQ
BsmtFinType2


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
NA
Po
BsmtCond

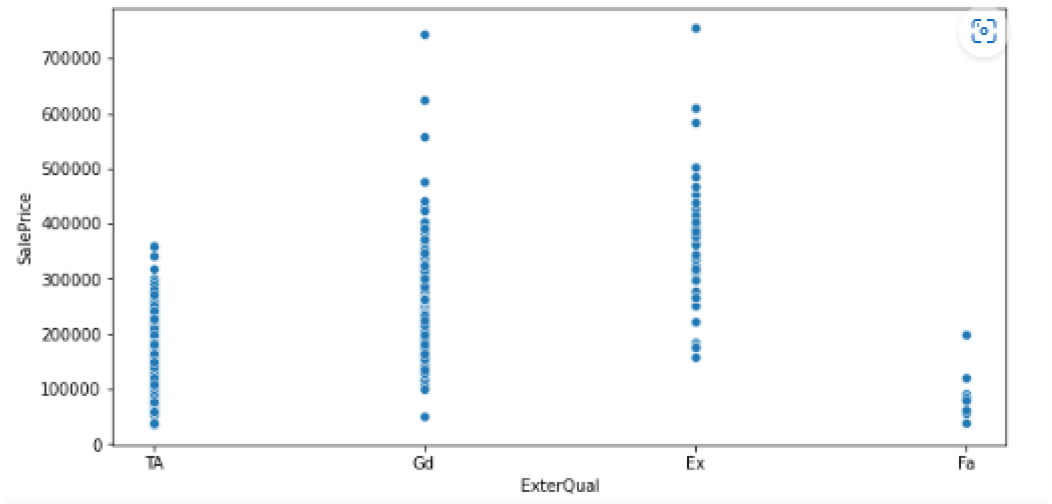

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
NA
Fa
BsmtQual



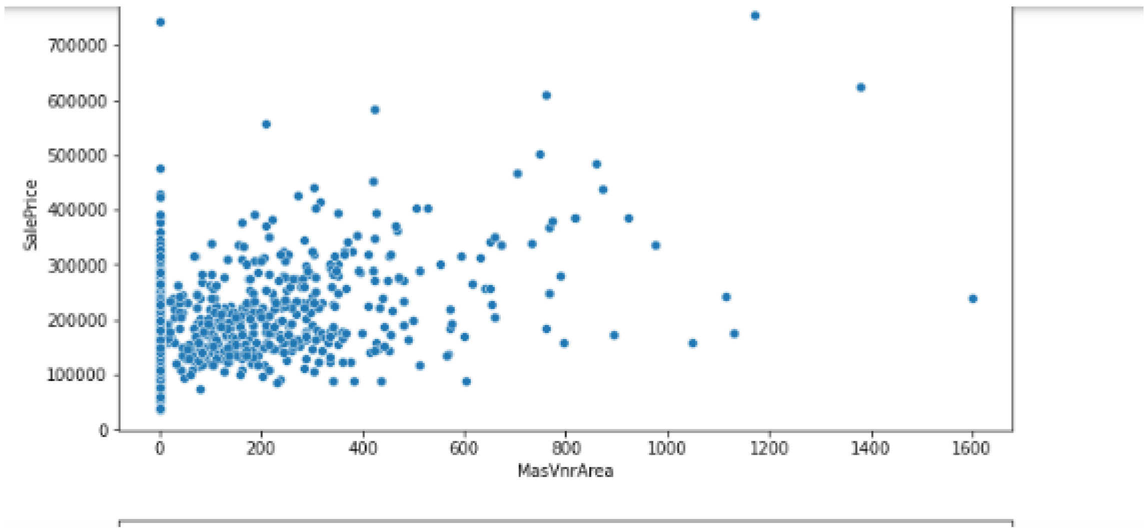
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
CBlock
PConc
BrkTil
Slab
Stone
Wood
Foundation



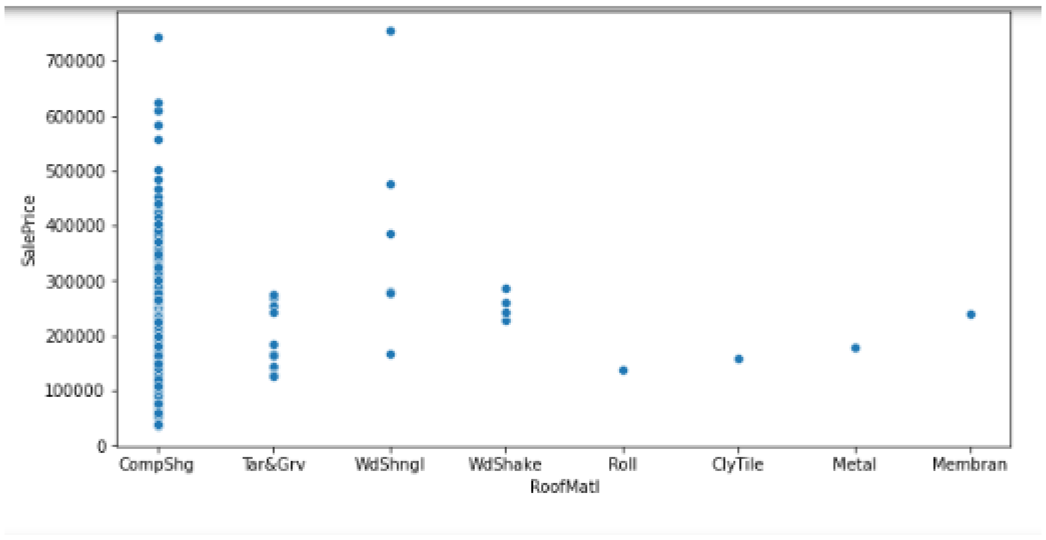
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Fa
Po
Ex
ExterCond



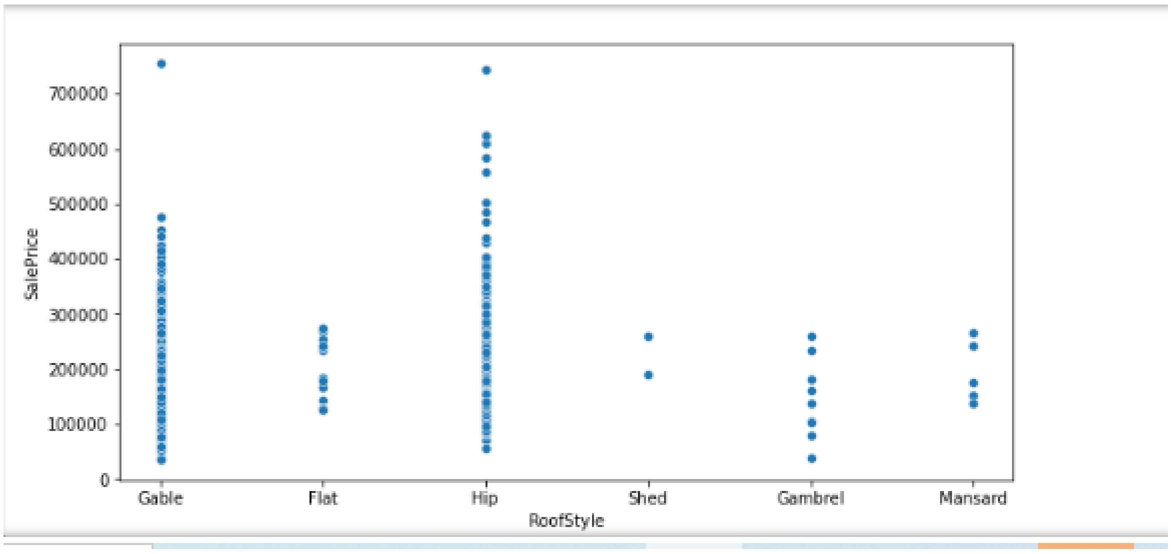
SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Ex
Fa
ExterQual

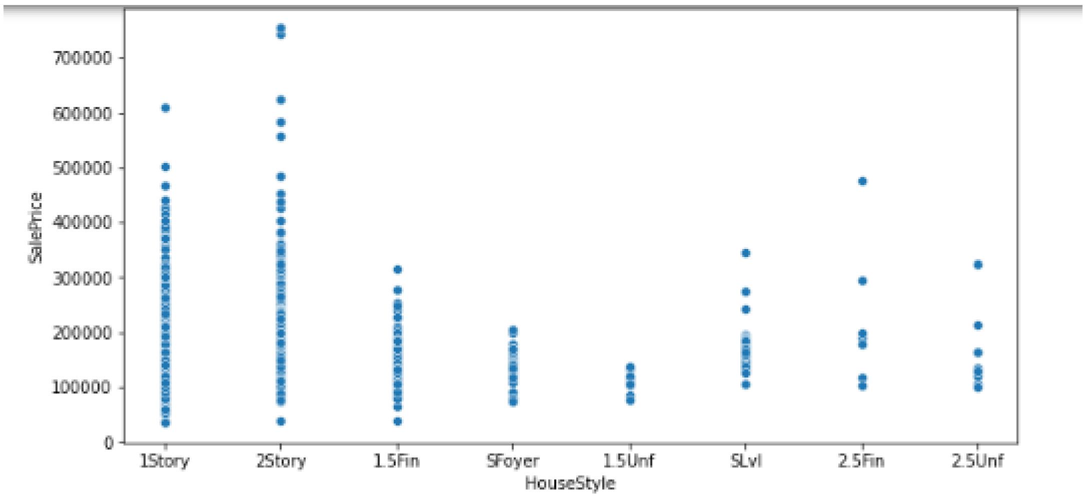


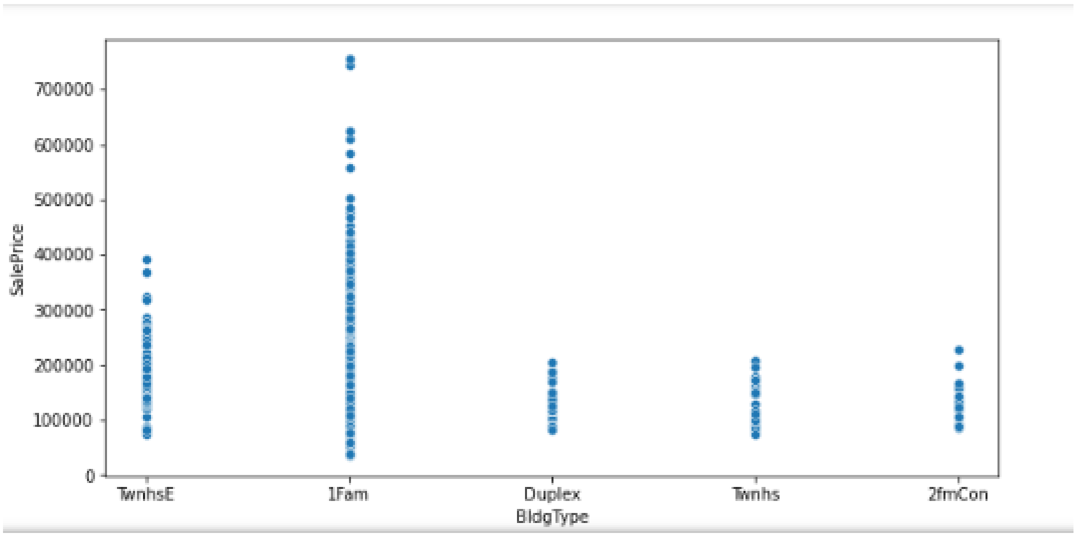
SalePrice
MasVnrArea

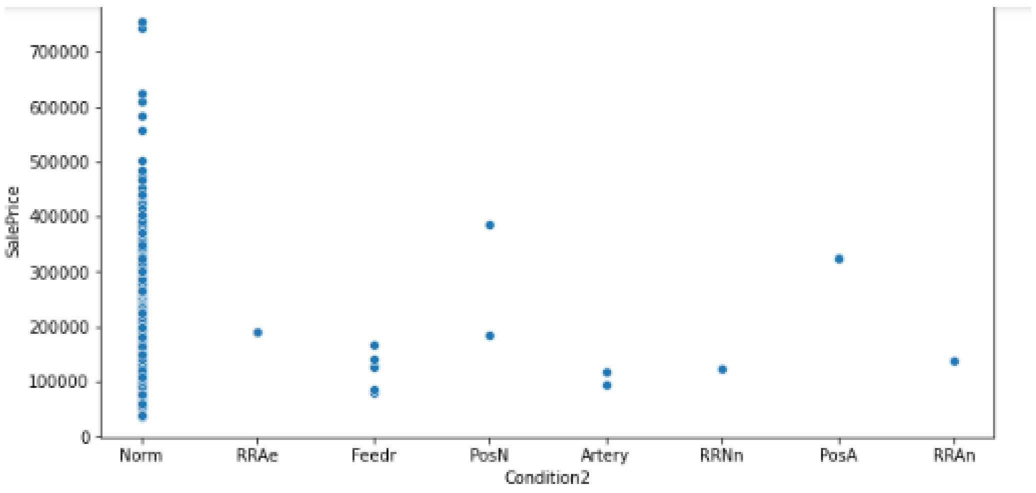


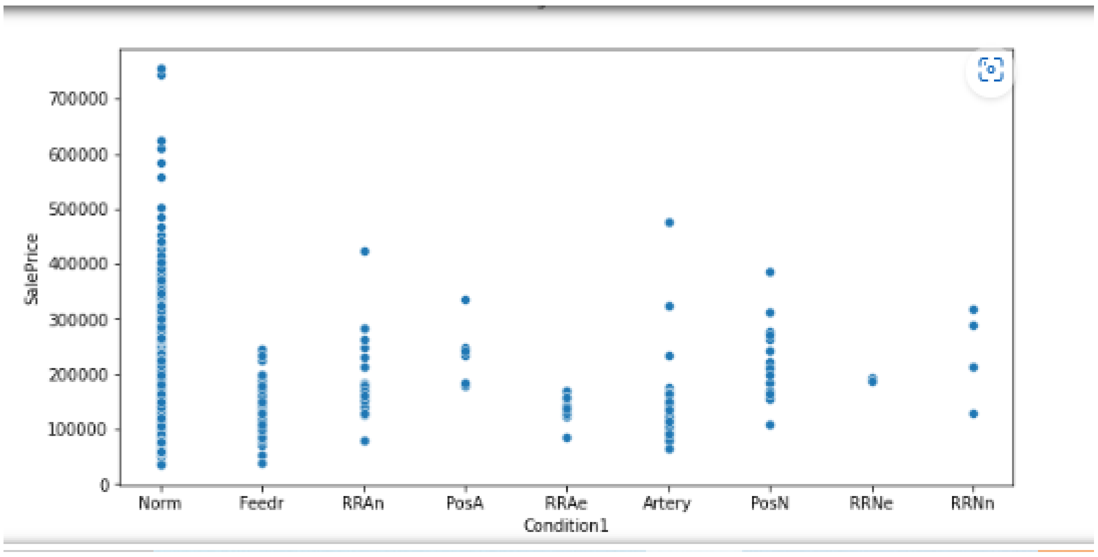
SalePrice
RoofMatl
CompShg
Tar&Grv
WdShngl
WdShake
Roll
ClyTile
Metal
Membran

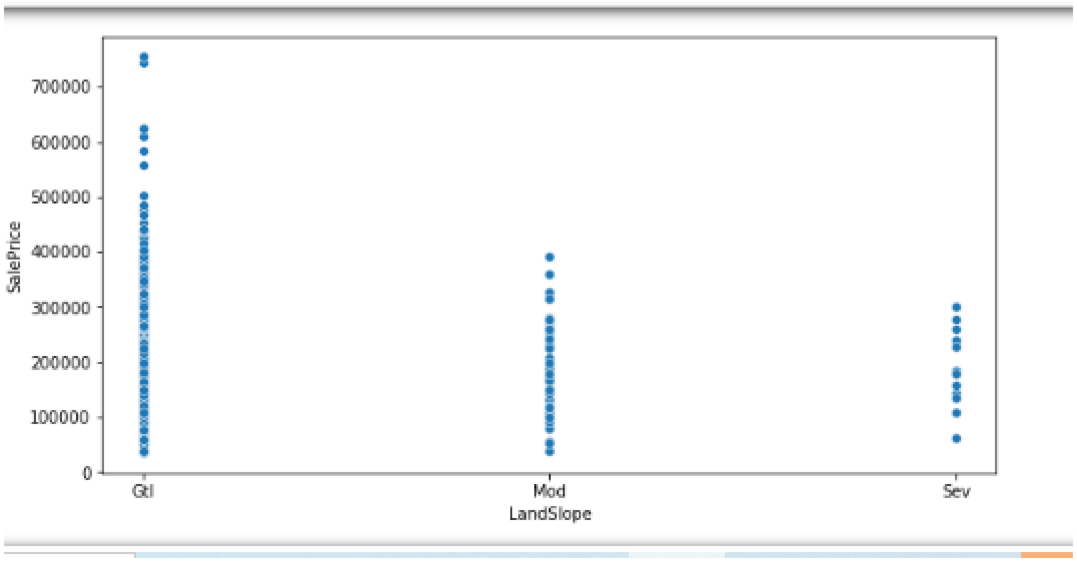

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofStyle

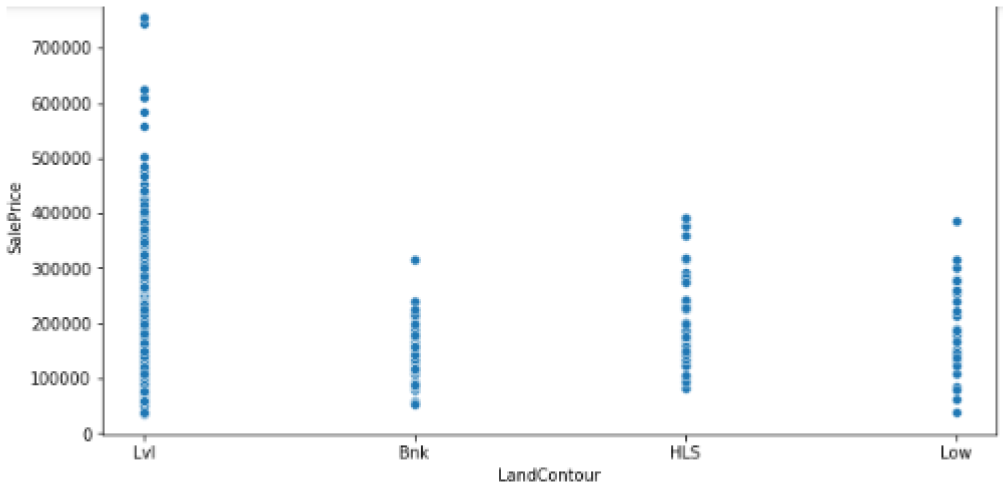

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle

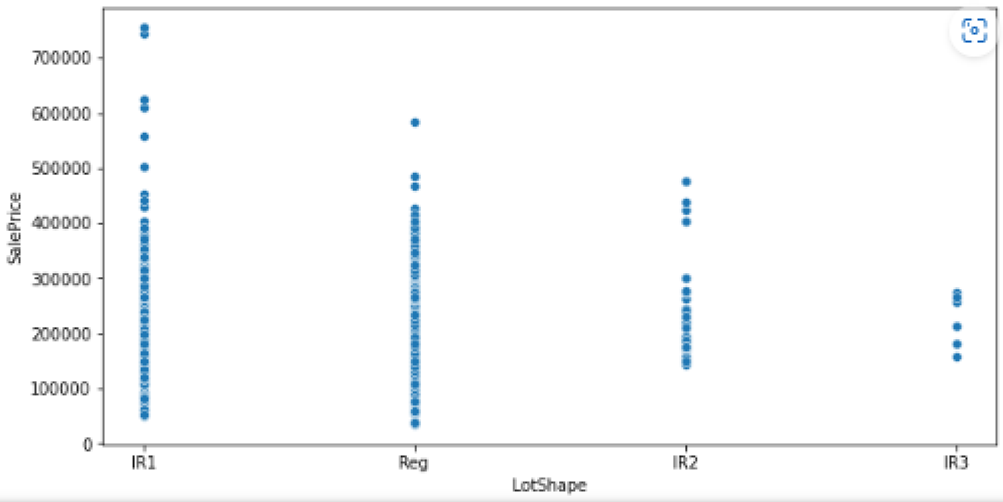

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType

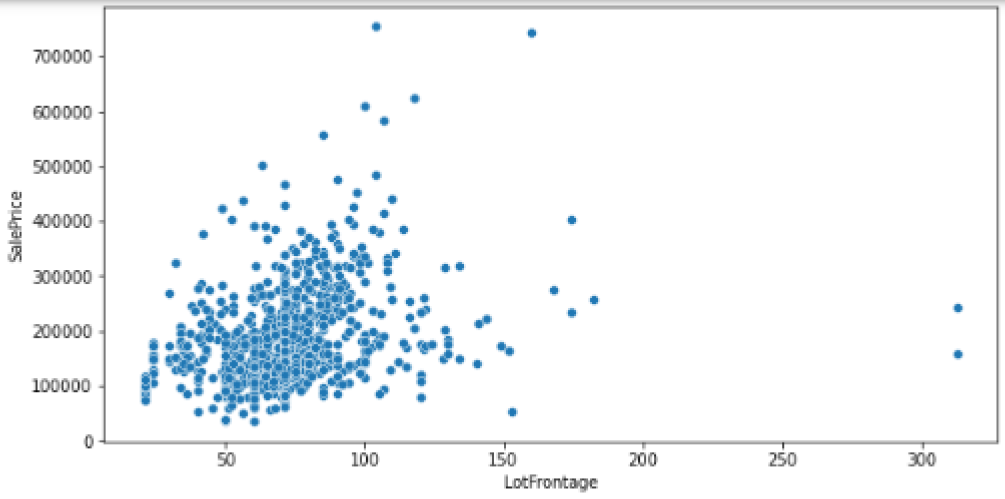

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
Condition2


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gtl
Mod
Sev
LandSlope

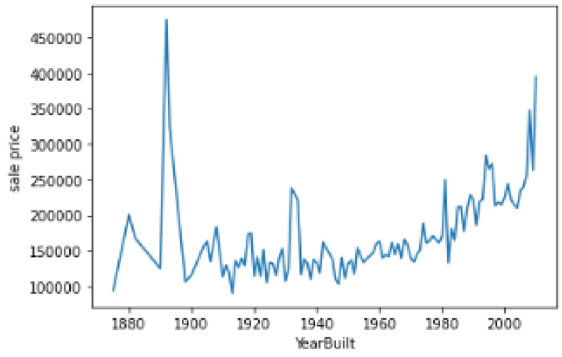

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Lvl
Bnk
HLS
Low
LandContour

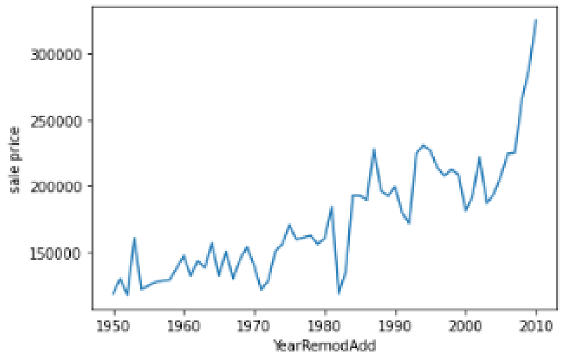

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
IR1
Reg
IR2
IR3
LotShape

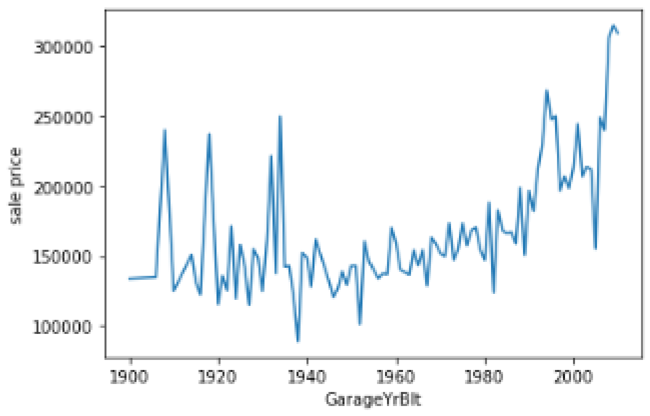

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
50
100
150
200
250
300
LotFrontage

RELATION ESTABLISHED WITH YEARS COLUMNS AND SALES PRICE

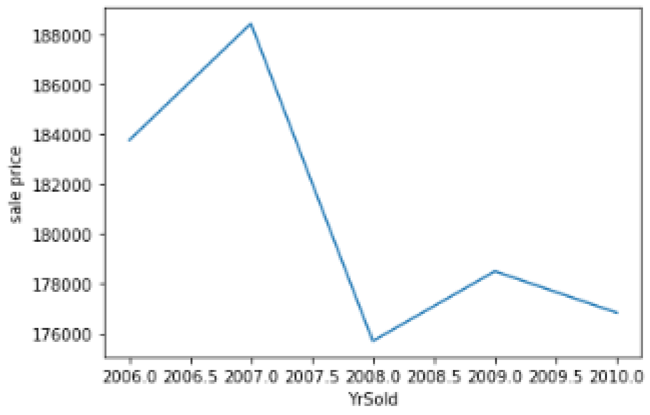
```
In [300]: for i in years :
              data = df_train.copy()
              data.groupby(i)['SalePrice'].mean().plot()
              plt.xlabel(i)
              plt.ylabel("sale price")
              plt.show()
```



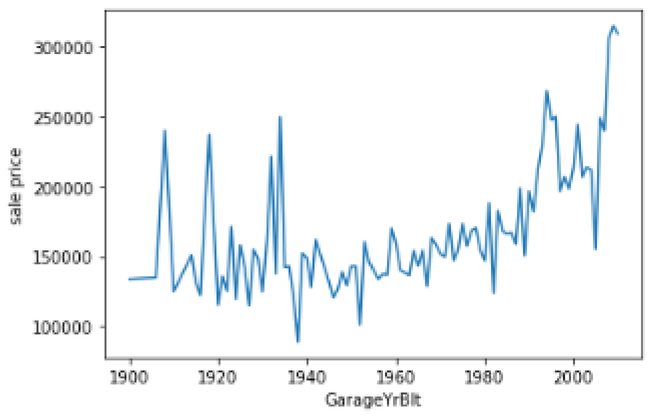




sale price
300000
250000
200000
150000
100000
1900
1920
1940
1960
1980
2000
GarageYrBlt



sale price
188000
186000
184000
182000
180000
178000
176000
2006.0
2006.5
2007.0
2007.5
2008.0
2008.5
2009.0
2009.5
2010.0
YrSold



sale price
300000
250000
200000
150000
100000
1900
1920
1940
1960
1980
2000
GarageYrBlt

sale price
188000
186000
184000
182000
180000
178000
176000
2006.0
2006.5
2007.0
2007.5
2008.0
2008.5
2009.0
2009.5
2010.0
YrSold

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  I used the chi and percentile method to decrease the number of features. I have taken only 80% of features

  Following algorithms used for the training and testing:

  Random Forest

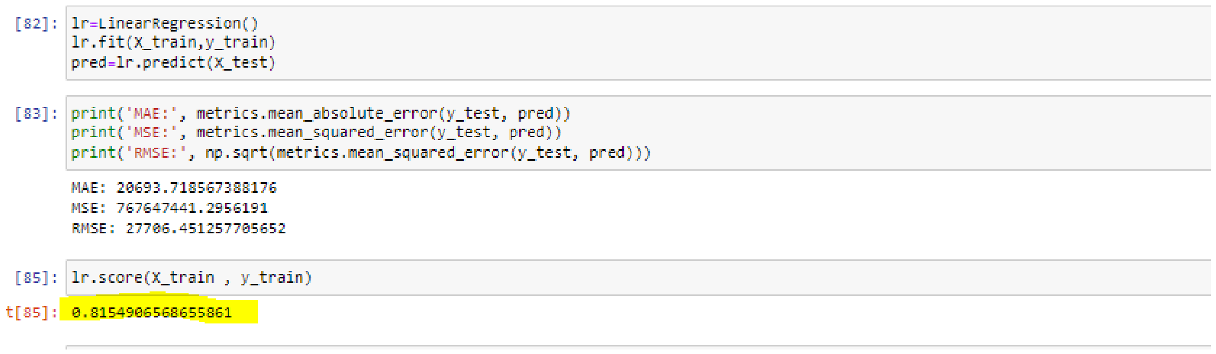
  Decision Tree

  Ada Boost

  Linear regression

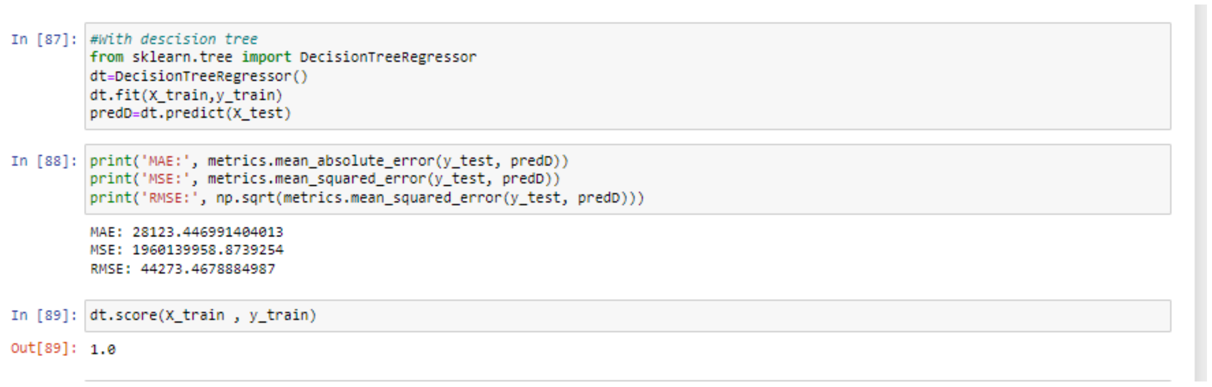
- Run and Evaluate selected models

1. Linear regression: 82%



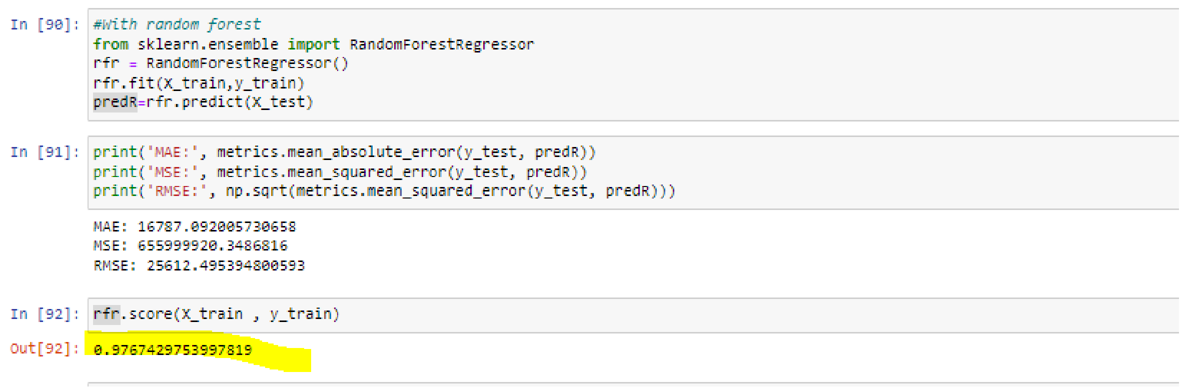
```
[82]: lr=LinearRegression()
      lr.fit(X_train,y_train)
      pred=lr.predict(X_test)

[83]: print('MAE:', metrics.mean_absolute_error(y_test, pred))
      print('MSE:', metrics.mean_squared_error(y_test, pred))
      print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))

      MAE: 20693.718567388176
      MSE: 767647441.2956191
      RMSE: 27706.451257705652

[85]: lr.score(X_train , y_train)

t[85]: 0.8154906568655861
```

2. Decision Tree



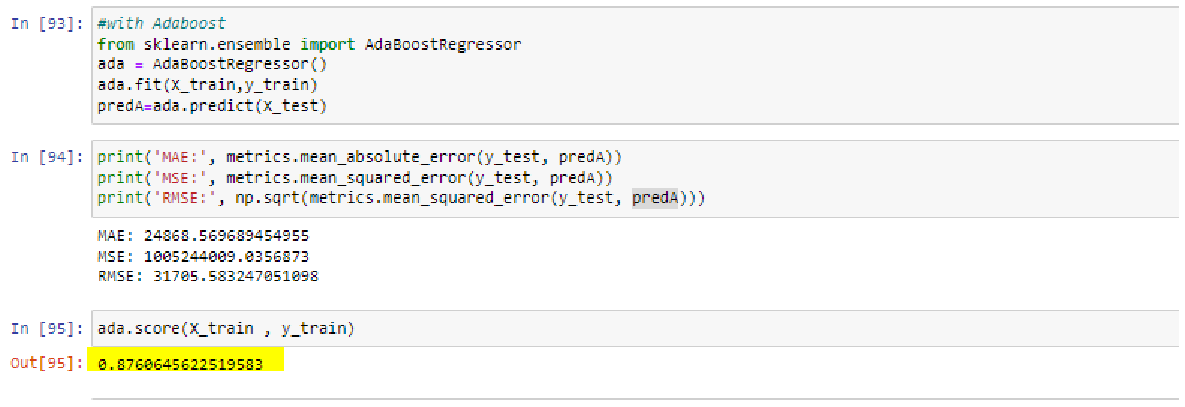
```
In [87]: #With descision tree
         from sklearn.tree import DecisionTreeRegressor
         dt=DecisionTreeRegressor()
         dt.fit(X_train,y_train)
         predD=dt.predict(X_test)

In [88]: print('MAE:', metrics.mean_absolute_error(y_test, predD))
         print('MSE:', metrics.mean_squared_error(y_test, predD))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predD)))

         MAE: 28123.446991404013
         MSE: 1960139958.8739254
         RMSE: 44273.4678884987

In [89]: dt.score(X_train , y_train)

Out[89]: 1.0
```

3. Random Forest – 97%



```
In [90]: #With random forest
         from sklearn.ensemble import RandomForestRegressor
         rfr = RandomForestRegressor()
         rfr.fit(X_train,y_train)
         predR=rfr.predict(X_test)

In [91]: print('MAE:', metrics.mean_absolute_error(y_test, predR))
         print('MSE:', metrics.mean_squared_error(y_test, predR))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predR)))

         MAE: 16787.092005730658
         MSE: 655999920.3486816
         RMSE: 25612.495394800593

In [92]: rfr.score(X_train , y_train)

Out[92]: 0.9767429753997819
```

4. Adaboost-87%



```
In [93]: #with Adaboost
         from sklearn.ensemble import AdaBoostRegressor
         ada = AdaBoostRegressor()
         ada.fit(X_train,y_train)
         predA=ada.predict(X_test)

In [94]: print('MAE:', metrics.mean_absolute_error(y_test, predA))
         print('MSE:', metrics.mean_squared_error(y_test, predA))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predA)))

         MAE: 24868.569689454955
         MSE: 1005244009.0356873
         RMSE: 31705.583247051098

In [95]: ada.score(X_train , y_train)

Out[95]: 0.8760645622519583
```

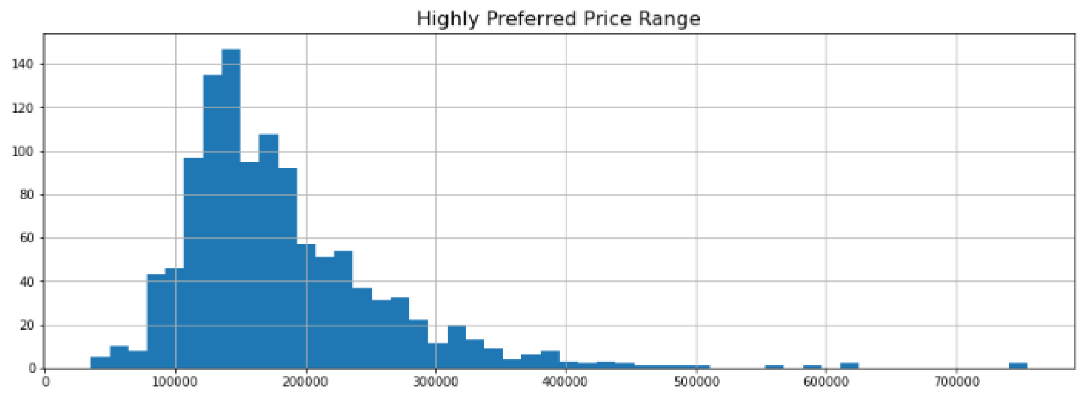
Key Metrics for success in solving problem under consideration

- The df_train.skew was used to find the features that were skewed
- Some of them were:

- MiscVal 22.996908
- PoolArea 13.203388
- Condition2 11.479618
- LotArea 10.714330
- Heating 10.072518
- ...
- GarageCond -3.559980
- SaleType -3.664995
- Functional -4.005071
- PoolQC -15.855023
- Street -16.970487

- The highly preferred price is between 100000-200000

```
In [346]: df_train["SalePrice"].hist(bins=50, figsize=(15, 5))
          plt.title("Highly Preferred Price Range", fontsize=16);
```

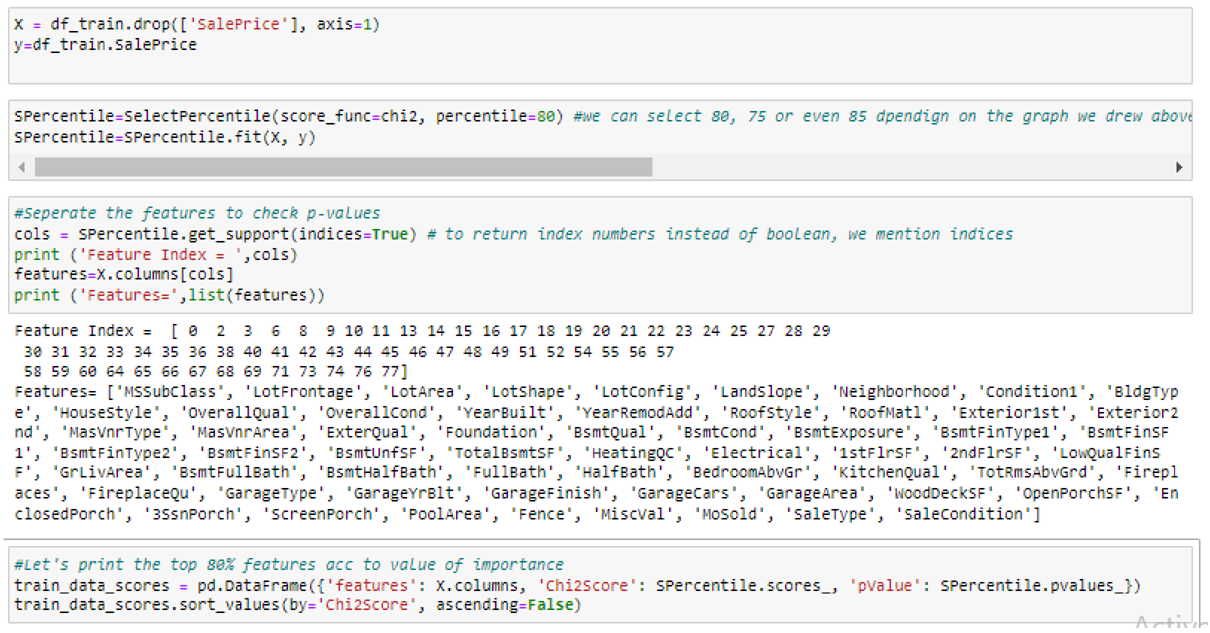


- According to the corr matrix following were highly correlated:

  There is a high correlation between the following features as per correlation matrix:

  - BldgType and MSSubClass
  - overallQual n SalePrice
  - SalePrice and GrLivArea
  - GrLivArea n OverallQual
  - TotRmsAbvGrd vs GrLivArea
  - GarageCars vs GarageArea
  - GrLivArea vs SalePrice
  - GrLivArea vs totrmsabvgrd
  - TotalBsmtSF vs 1stFlrSF

  - Fllwong are the 80% of the features selected for modelling via chi score and percentile method



```
X = df_train.drop(['SalePrice'], axis=1)
y=df_train.SalePrice
```

```
SPercentile=SelectPercentile(score_func=chi2, percentile=80) #we can select 80, 75 or even 85 dpendign on the graph we drew abov
SPercentile=SPercentile.fit(X, y)
```

```
#Seperate the features to check p-values
cols = SPercentile.get_support(indices=True) # to return index numbers instead of boolean, we mention indices
print ('Feature Index = ',cols)
features=X.columns[cols]
print ('Features=',list(features))
```

```
Feature Index =  [ 0  2  3  6  8  9 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25 27 28 29
 30 31 32 33 34 35 36 38 40 41 42 43 44 45 46 47 48 49 51 52 54 55 56 57
 58 59 60 64 65 66 67 68 69 71 73 74 76 77]
Features= ['MSSubClass', 'LotFrontage', 'LotArea', 'LotShape', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'BldgTyp
e', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2
nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF
1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'HeatingQC', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinS
F', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Firepl
aces', 'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'En
closedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'Fence', 'MiscVal', 'MoSold', 'SaleType', 'SaleCondition']
```

```
#Let's print the top 80% features acc to value of importance
train_data_scores = pd.DataFrame({'features': X.columns, 'Chi2Score': SPercentile.scores_, 'pValue': SPercentile.pvalues_})
train_data_scores.sort_values(by='Chi2Score', ascending=False)
```

•

CONCLUSION

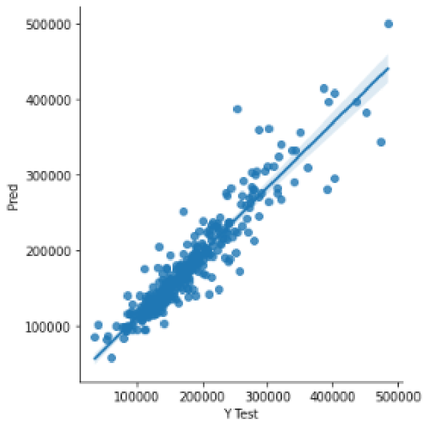
Key Findings and Conclusions of the Study

Random Forest is the best method. Hyper tuning is not used as models already have a high accuracy

```
In [113]: #with random forest
          data = pd.DataFrame({'Y Test':y_test , 'Pred':predR},columns=['Y Test','Pred'])
          sns.lmplot(x='Y Test',y='Pred',data=data,palette='rainbow')
          data.head()
```

Out[113]:

	Y Test	Pred
903	145000	175614.78
281	110500	128515.50
167	178000	183054.98
965	132500	133221.00
329	215200	191404.49


Pred
Y Test