Housing Project

Submitted by:

VINAYAK RATAN

INTRODUCTION

- ## Business Problem Framing

  House prices are increasing every year which has necessitated the modelling of house price prediction. These models constructed, help the customers to purchase a house suitable for their need

- ## Conceptual Background of the Domain Problem

  Houses are one of the necessary needs 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.

- ## Review of Literature

  Dr. M. Thamarai, Dr. S P. Malarvizhi used two algorithms like decision tree regression and multiple linear regression were used in predicting the prices of the houses. Comparatively the performance of multiple linear regression is found to be better than the decision tree regression in predicting the house prices.

- ## Motivation for the Problem Undertaken

  House prices are increasing every year which has necessitated the modelling of house price prediction. These models constructed, help the customers to purchase a house suitable for their need

Analytical Problem Framing

- ## Data Sources and their formats

  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:

  What are the data sources, their origins, their formats and other details that you find necessary? They can be described here. Provide a proper data description. You can also add a snapshot of the data.

- ## Data Pre-processing Done

```
In [22]: housing_df['MasVnrType'].fillna('None', inplace = True)

In [23]: housing_df['MasVnrType'].isnull().sum()
Out[23]: 0

In [24]: housing_df['BsmtQual'].fillna('No Basement', inplace = True)
         housing_df['BsmtCond'].fillna('No Basement', inplace = True)
         housing_df['BsmtExposure'].fillna('No Basement', inplace = True)
         housing_df['BsmtFinType1'].fillna('No Basement', inplace = True)
         housing_df['BsmtFinType2'].fillna('No Basement', inplace = True)

In [25]: housing_df[['BsmtQual','BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2']].isnull().sum()
Out[25]: BsmtQual       0
         BsmtCond       0
         BsmtExposure   0
         BsmtFinType1   0
         BsmtFinType2   0
         dtype: int64

In [26]: housing_df['FireplaceQu'].fillna('No Fireplace', inplace = True)
         housing_df['GarageType'].fillna('No Garage', inplace = True)
         housing_df['GarageFinish'].fillna('No Garage', inplace = True)
         housing_df['GarageQual'].fillna('No Garage', inplace = True)
         housing_df['GarageCond'].fillna('No Garage', inplace = True)

In [27]: housing_df[['FireplaceQu','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']].isnull().sum()
Out[27]: FireplaceQu    0
         GarageType     0
         GarageFinish   0
         GarageQual     0
         GarageCond     0
         dtype: int64

In [28]: housing_df['LotFrontage'].fillna(housing_df['LotFrontage'].mean(), inplace = True)
         housing_df['GarageYrBlt'].fillna(housing_df['GarageYrBlt'].mean(), inplace = True)

In [29]: housing_df['MasVnrArea'].fillna(housing_df['MasVnrArea'].mean(), inplace = True)

In [30]: housing_df[num].isnull().sum()
```

- **Hardware and Software Requirements and Tools Used**

  Hardware requirements:

  - Laptop or PC to analyse the data
  - Software requirements: 4GB/8GB RAM (8GB or high preferred), minimum 1GB graphics card (any), 512GB HDD.
  - Tools Used: Jupyter Notebook – Entire model
  - Libraries used: NumPy (for numerical computations like imputations), Pandas (used reading the data, data cleaning, data manipulation, correlation, summary statistics, skewness, feature engineering) Seaborn (for plotting), Matplotlib (for plotting), scikit-learn, SciPy (for z score), stats models (for VIF to check multicollinearity), joblib (for pickling i.e. deploying the best model trained)

Model Development and Evaluation

- **Problem solving methods**

  Approach followed during the course of predicting the house sales price

  - Data given was imported as DataFrame. Next, the basic information like the datatypes, checking the count of non-null values in the data with info() function, then summary statistics of both numerical and categorical data which provides us the information about the count, frequency, unique, top values for the categorical data and mean, standard deviation, minimum, maximum, 25^{th} percentile, 50^{th} percentile (median), 75^{th} percentile of the numerical data.
  - Checking null values, imputing the null values, dropping the columns with null values almost close to the size to data, plots like countplot, barplot, regression plot, categorical plot were plotted to check the effectiveness of the input variable with the target variable and its behaviour.

- Next, correlation matrix was plotted through which we could identify the how the input variable is correlated with the target variable and also, we see the multicollinearity which is even checked by seeing the variance inflation factor.
- Next skewness was checked and removed with log and square transforming methods. Next outliers were identified.
- Later the data was split into features (X) and target (y).
- MinMaxScaler method was used for transforming and to reduce the dimensionality of the data PCA (principal component analysis) was used.
- Model Building – 5 models used for training along with cross validation and the best model was tuned with hyperparameters. And the two models were deployed one with tuned one and other without and comparison was done with predicted and original with plots.
- Lastly, the price was predicted for the test data provided which was cleaned before predicting the sales price of the house.

• Algorithms used

I have used total of 5 models for training purpose as listed below

- Linear Regression
- Decision Tree Regressor
- K Neighbour Regressor
- Support Vector Regressor
- Random Forest Regressor

All the models were run to get the best accuracy for best random state in a range of 0 to 200. Later cross validation was performed for all the models for cross validation range 2 to 15. The Best model was Random forest regressor which was tuned using gridsearchCV.

• Run and evaluate selected models

Applying the algorithms for model building

In [75]:
```
# Importing the libraries for model building
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
```

In [76]:
```
# Instantiating each algorithm
lr = LinearRegression()
dtr = DecisionTreeRegressor()
knr = KNeighborsRegressor()
svr = SVR()
rfr = RandomForestRegressor()

# List of instantiated algorithms
model = [lr, dtr, knr, svr, rfr]
```

In [77]:
```
# Maximum accuracy
maxAccu = 0

# Best random state value for which accuracy is achieved
maxRS = 0

for m in model:
    for i in range(0,200):
        x_train, x_test,y_train, y_test = train_test_split(X,y, random_state=i, test_size=0.20)
        m.fit(x_train, y_train)
        pred_train= m.predict(x_train)
        pred_test = m.predict(x_test)
        score_train = m.score(x_train, y_train)
        score_test = m.score(x_test, y_test)
        r2 = r2_score(y_test, pred_test)
        print(f'----------------At random State {i} ------------------')
        print(f"Training score is {score_train}")
        print(f"Testing score is {score_test}")
        print(f'r2 score of {m} is {r2}')

        if r2 > maxAccu:
            maxAccu = r2
            maxRS = i
            print(f"Maximum score of {m} is {maxAccu} at Random state {i}")
            print('\n')
    print('*'*120)
```

Random Forest Regressor

- Maximum score of RandomForestRegressor() is 0.8134658396095735 at Random state 107
- Training score is 0.9533843304262911
- Testing score is 0.8134658396095735
- r2 score of RandomForestRegressor() is 0.8134658396095735

Cross validation of all the Algorithms for regression

In [78]:
```
# Before proceeding with the hpyer parameter tunning we will check for cross validation
from sklearn.model_selection import cross_val_score
max_cvscore = 0
max_cv = 0
for m in model:
    print(f'----------------------- Cross Validation of {m} --------------------------')
    for j in range(2,15):
        cv_score = cross_val_score(m, X,y,cv=j)
        cv_mean = cv_score.mean()
        print(f"At cross fold {j} the cv score is {cv_mean}")
        print('\n')

        if cv_mean > max_cvscore:
            max_cvscore=cv_mean
            max_cv = j
            print(f"At cross fold {j} the Maximum CV score is {max_cvscore}")
            print('\n')
    print('*'*120)
    print('\n')
```

For Randomforest at cross fold 3 the Maximum CV score is 0.7180619511702199

In [79]:
```
# Training Random forest Regressor at Random state 107
x_train, x_test,y_train, y_test = train_test_split(X,y, random_state=107, test_size=0.20)
rfr.fit(x_train, y_train)
pred_train= rfr.predict(x_train)
pred_test = rfr.predict(x_test)
score_train = rfr.score(x_train, y_train)
score_test = rfr.score(x_test, y_test)
r2 = r2_score(y_test, pred_test)
print(f"Training score is {score_train}")
print(f"Testing score is {score_test}")
print(f'r2 score of {rfr} is {r2}')
```

```
Training score is 0.9498713345390536
Testing score is 0.8149124912641879
r2 score of RandomForestRegressor() is 0.8149124912641879
```

Hyper Parameter Tunning

In [80]:
```
from sklearn.model_selection import GridSearchCV
```

In [81]:
```
# from sklearn.model_selection import GridSearchCV
param = {'n_estimators':[20,100,200,300],'max_depth': [2,4,6,8],'criterion': ['mse','mae'],'max_features':['sqrt','log2']}

gcv1 = GridSearchCV(rfr,param,cv=3,scoring= 'accuracy')

# training
gcv1.fit(x_train, y_train)

# Getting the best parameters
print("Best parameters:", gcv1.best_params_)

print("Best Estimator:",gcv1.best_estimator_)

gcv_pred=gcv1.best_estimator_.predict(x_test)

print('Final Accuracy with Random Forest Regressor:', r2_score(y_test,gcv_pred))
```

```
Best parameters: {'criterion': 'mse', 'max_depth': 2, 'max_features': 'sqrt', 'n_estimators': 20}
Best Estimator: RandomForestRegressor(criterion='mse', max_depth=2, max_features='sqrt',
                      n_estimators=20)
Final Accuracy with Random Forest Regressor: 0.41435673830978015
```

- **Key Metrics for success in solving problem under consideration**

In [12]:
```
# Columns with integer datatype which are probably the categorical column must be object datatype
# Hence, we need to replace numbers with their values in their respective columns

# MSSubClass dictionary
MSSubClass_dic = {20:'1-STORY 1946 & NEWER ALL STYLES',
                  30:'1-STORY 1945 & OLDER',
                  40:'1-STORY W/FINISHED ATTIC ALL AGES',
                  45:'1-1/2 STORY - UNFINISHED ALL AGES',
                  50:'1-1/2 STORY FINISHED ALL AGES',
                  60:'2-STORY 1946 & NEWER',
                  70:'2-STORY 1945 & OLDER',
                  75:'2-1/2 STORY ALL AGES',
                  80:'SPLIT OR MULTI-LEVEL',
                  85:'SPLIT FOYER',
                  90:'DUPLEX - ALL STYLES AND AGES',
                  120:'1-STORY PUD (Planned Unit Development) - 1946 & NEWER',
                  150:'1-1/2 STORY PUD - ALL AGES',
                  160:'2-STORY PUD - 1946 & NEWER',
                  180:'PUD - MULTILEVEL - INCL SPLIT LEV/FOYER',
                  190:'2 FAMILY CONVERSION - ALL STYLES AND AGES'}

condition = {10:'Very Excellent', 9:'Excellent', 8:'Very Good', 7:'Good',
             6:'Above Average',5:'Average', 4:'Below Average',3:'Fair',
             2:'Poor',1:'Very Poor'}

housing_df['MSSubClass'] = housing_df['MSSubClass'].map(MSSubClass_dic)
housing_df['OverallQual'] = housing_df['OverallQual'].map(condition)
housing_df['OverallCond'] = housing_df['OverallCond'].map(condition)
```

Splitting the data

In [63]:
```
X = housing_df.iloc[:,:-1]
y = housing_df.iloc[:,-1]
```

In [64]:
```
X.shape, y.shape
```

Out[64]:
```
((1168, 74), (1168,))
```

Checking variance inflation factor

In [65]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_hpp=pd.DataFrame()
vif_hpp['VIF_Factor']= [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_hpp['features names']=X.columns
print(vif_hpp)
```

```
      VIF_Factor features names
0      92.389837     MSSubClass
1      13.681922       MSZoning
2     113.971108    LotFrontage
3     975.102959        LotArea
4     340.366577         Street
..           ...            ...
69      1.183557       MiscVal
70      7.013965        MoSold
71  32684.833378        YrSold
72     18.302399      SaleType
73     12.055049  SaleCondition

[74 rows x 2 columns]
```

Scaling the data using MinMaxScaler

In [72]:
```
from sklearn.preprocessing import MinMaxScaler
mms_hpp = MinMaxScaler()
X = mms_hpp.fit_transform(X)
X
```

Out[72]:
```
array([[0.5625    , 0.37164242, 0.27516426, ..., 0.25      , 1.        ,
        0.64      ],
       [0.5625    , 0.47992606, 0.51666423, ..., 0.25      , 1.        ,
        0.64      ],
       [0.5625    , 0.46763652, 0.41967013, ..., 0.25      , 1.        ,
        0.64      ],
       ...,
       [0.5625    , 0.03432275, 0.11599031, ..., 0.75      , 1.        ,
        0.64      ],
       [0.        , 0.25125379, 0.38775984, ..., 0.5       , 1.        ,
        0.64      ],
       [0.5625    , 0.37164242, 0.37161715, ..., 0.        , 1.        ,
        0.64      ]])
```

Scaling is used to transform the data to normal distribution.

PCA

- Since there are more features we will apply PCA

In [73]:
```
from sklearn.decomposition import PCA
```

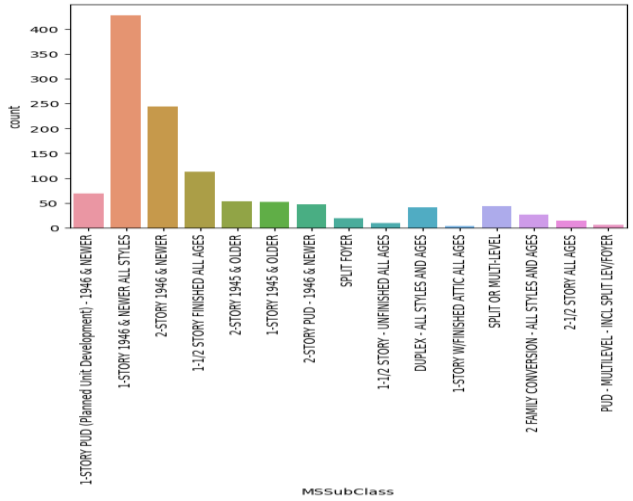
In [74]:
```
pca = PCA(n_components=10)
xpca = pca.fit_transform(X)
X = xpca
pd.DataFrame(data = X)
```

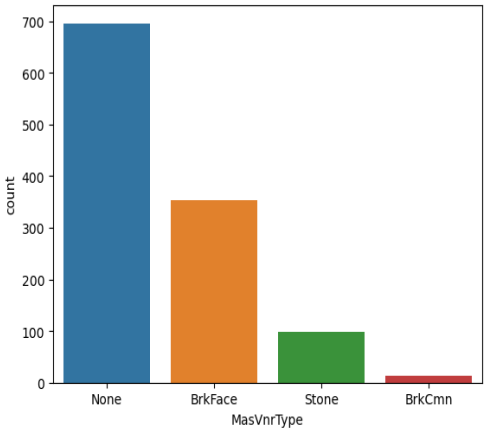
Out[74]:

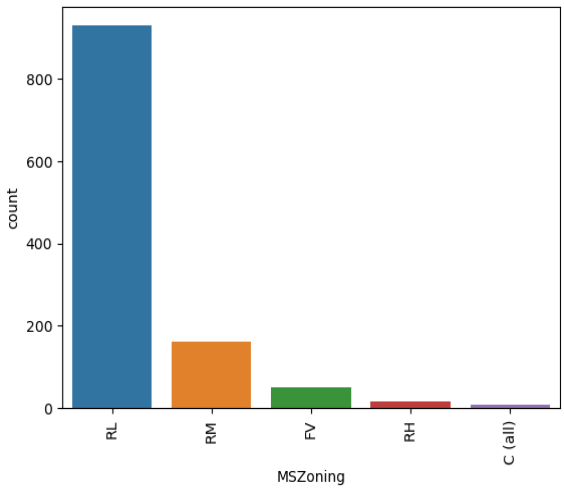
	0	1	2	3	4	5	6	7	8	9
0	-0.193557	0.565626	0.053045	-0.383790	0.625316	0.370473	-0.375570	-0.573220	0.194241	0.251494
1	0.498892	0.541537	0.262133	0.731679	0.372041	-0.635116	-0.232543	0.691742	0.654691	-0.302106
2	0.804574	0.604241	0.672339	-0.020050	-0.299700	-0.209119	-0.000054	-0.014828	0.168208	-0.242075
3	0.360770	0.492949	0.035688	-0.483149	0.272973	-0.079902	-0.600977	-0.369559	-0.560915	0.022970
4	0.610432	0.848005	0.432991	0.273961	-0.082661	0.079417	0.173378	-0.267831	-0.341940	0.255947

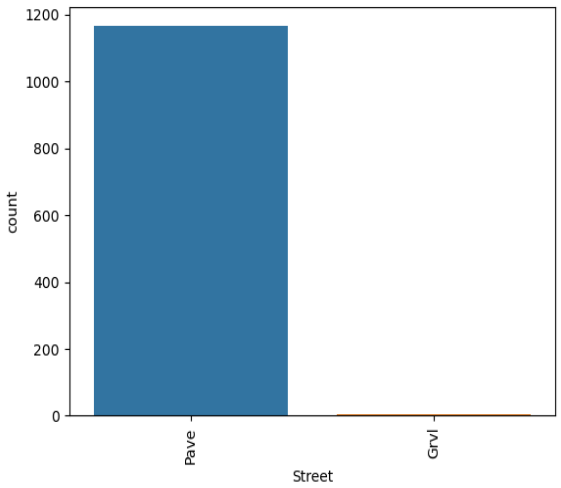
PCA (principal component analysis) is used since the number of features are to many, in order to reduce the dimensionality of the problem.

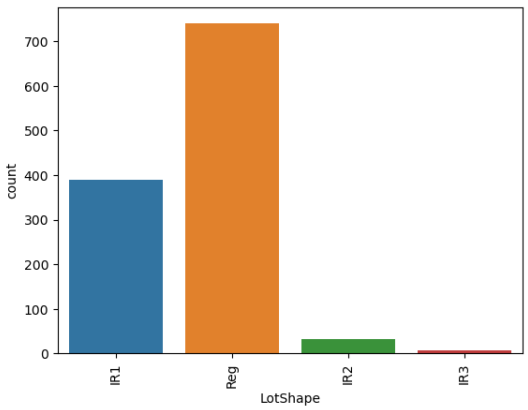
- **Visualizations**

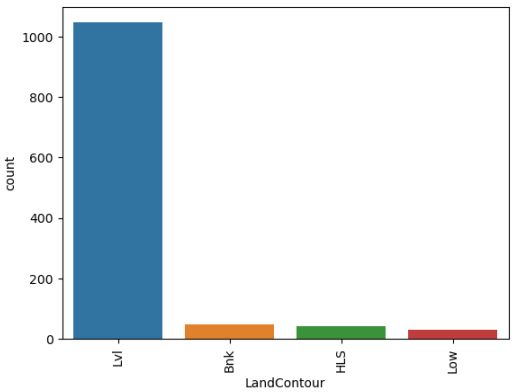


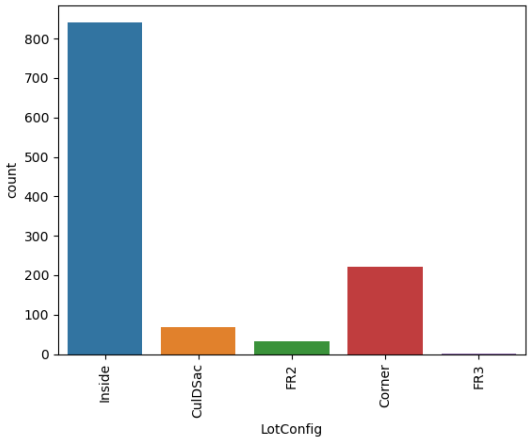


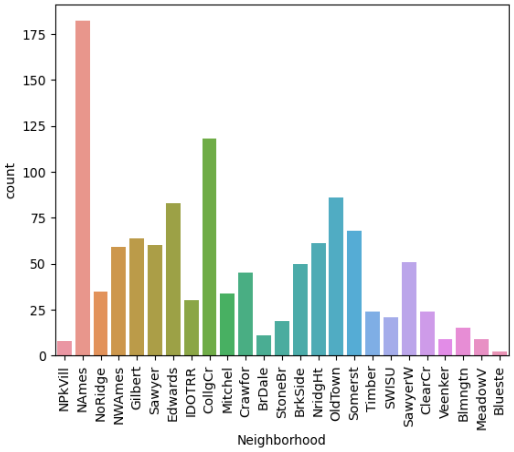


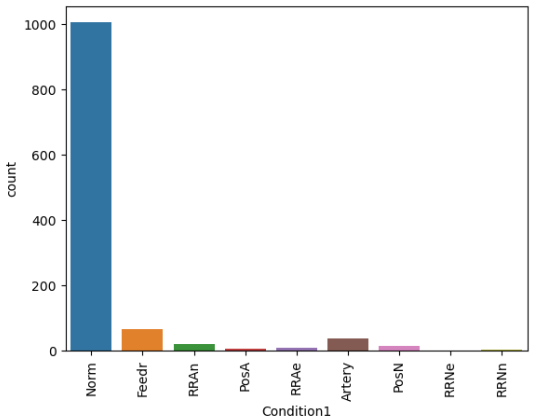


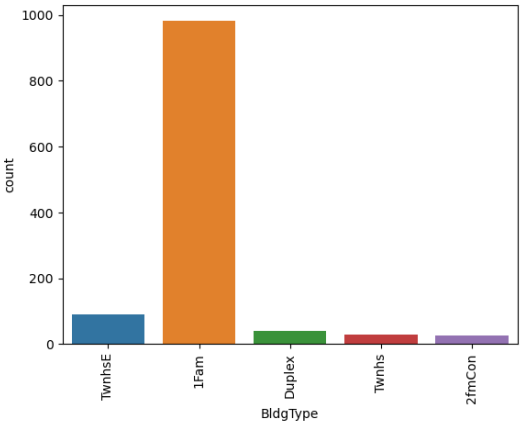


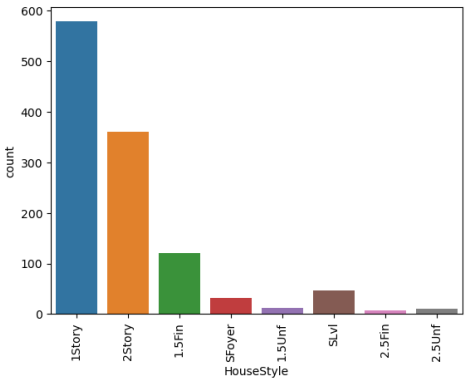


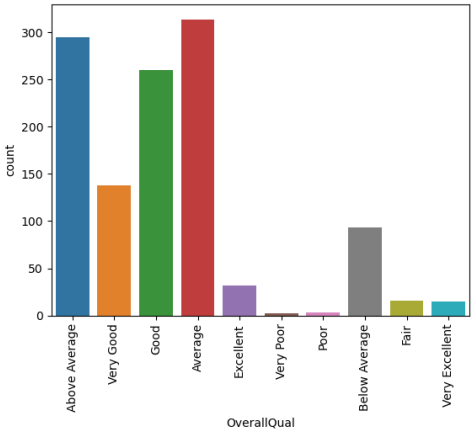

count
LotConfig
Inside
CulDSac
FR2
Corner
FR3

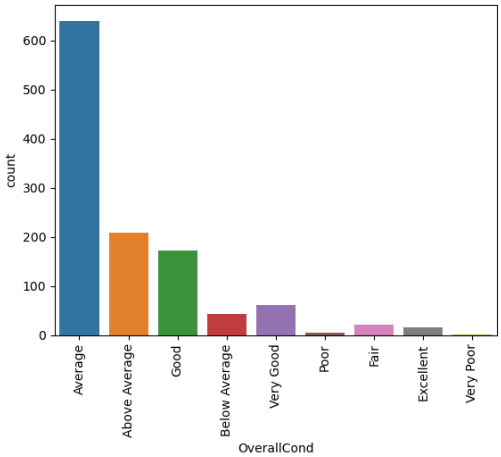

count
Neighborhood
NPkVill
NAmes
NoRidge
NWAmes
Gilbert
Sawyer
Edwards
IDOTRR
CollgCr
Mitchel
Crawfor
BrDale
StoneBr
BrkSide
NridgHt
OldTown
Somerst
Timber
SWISU
SawyerW
ClearCr
Veenker
Blmngtn
MeadowV
Blueste

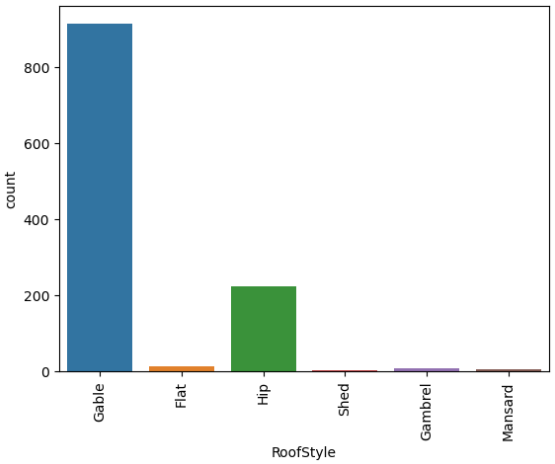

count
Condition1
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn

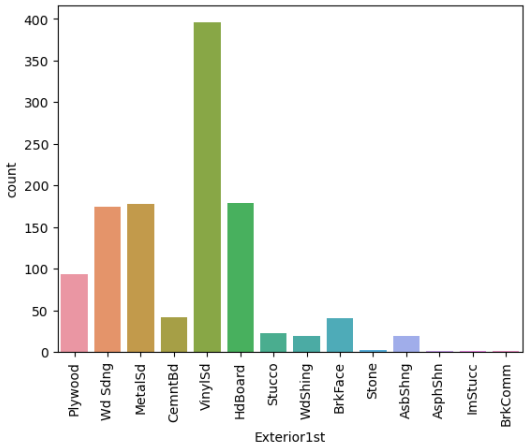

count
BldgType
TwnhsE
1Fam
Duplex
Twnhs
2fmCon

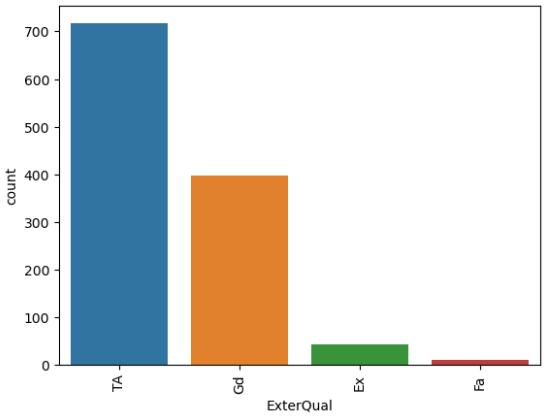

count
HouseStyle
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf

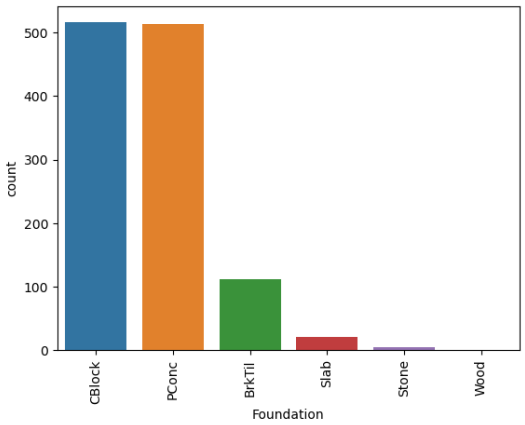

count
OverallQual
Above Average
Very Good
Good
Average
Excellent
Very Poor
Poor
Below Average
Fair
Very Excellent

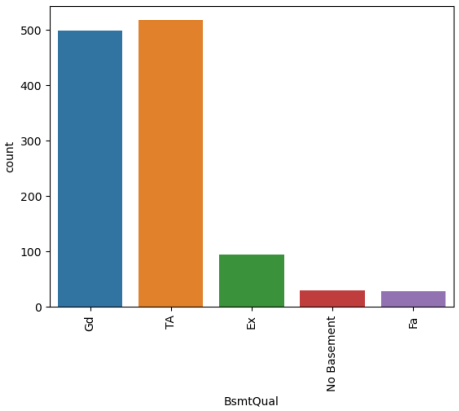

count
OverallCond
Average
Above Average
Good
Below Average
Very Good
Poor
Fair
Excellent
Very Poor

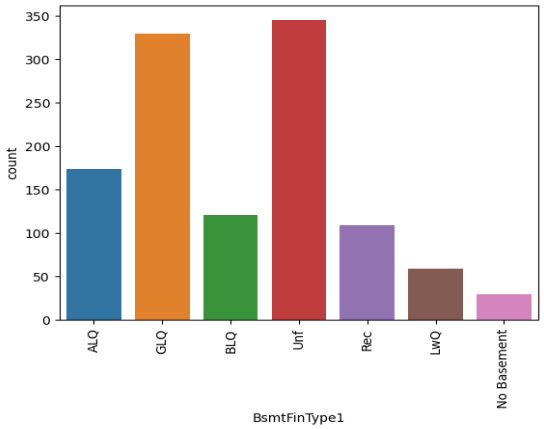

count
RoofStyle
Gable
Flat
Hip
Shed
Gambrel
Mansard

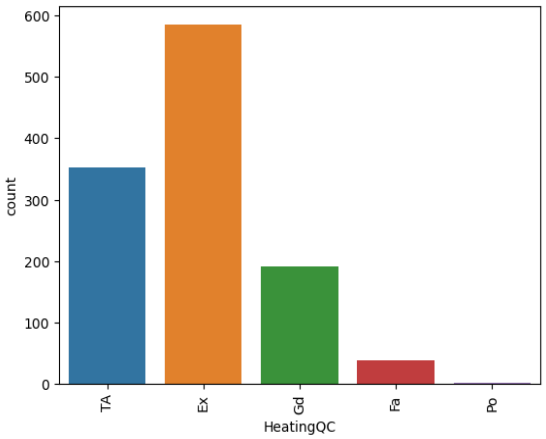

count
Plywood
Wd Sdng
MetalSd
CemntBd
VinylSd
HdBoard
Stucco
WdShing
BrkFace
Stone
AsbShng
AsphShn
ImStucc
BrkComm
Exterior1st

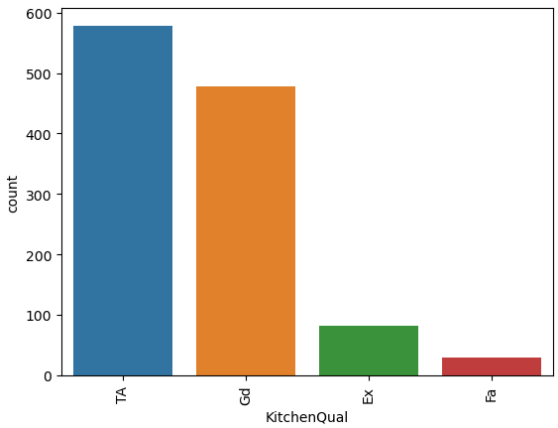

count
TA
Gd
Ex
Fa
ExterQual

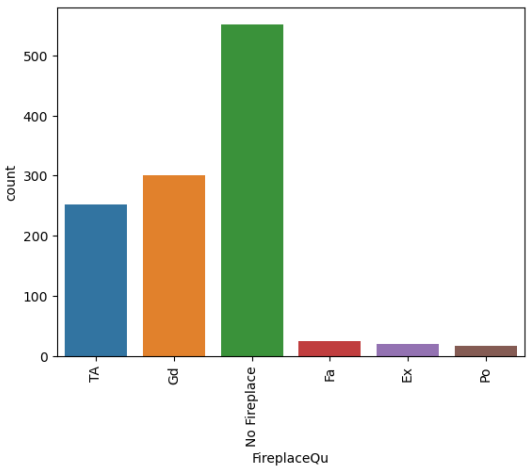

count
CBlock
PConc
BrkTil
Slab
Stone
Wood
Foundation

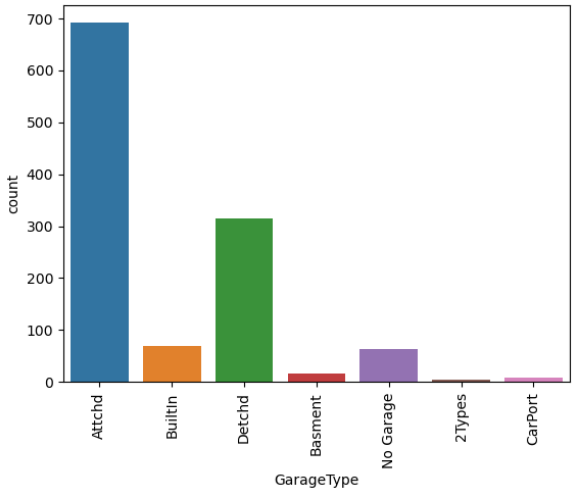

count
Gd
TA
Ex
No Basement
Fa
BsmtQual

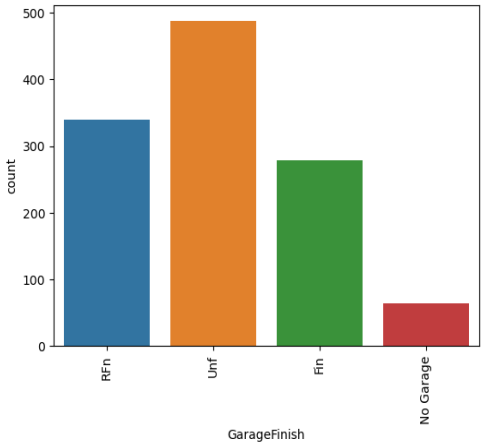

count
ALQ
GLQ
BLQ
Unf
Rec
LwQ
No Basement
BsmtFinType1

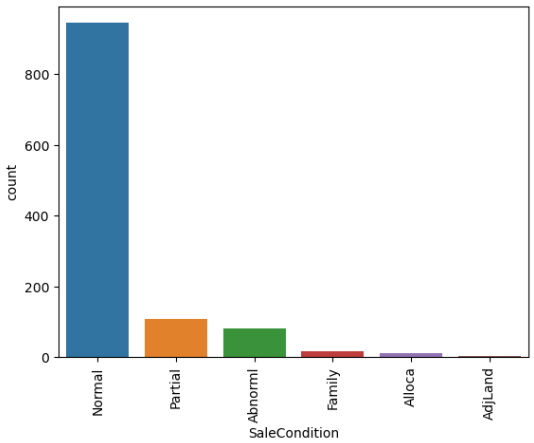

count
TA
Ex
Gd
Fa
Po
HeatingQC

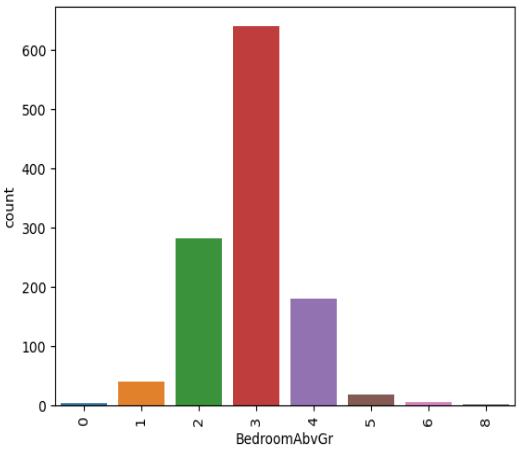

count
TA
Gd
Ex
Fa
KitchenQual

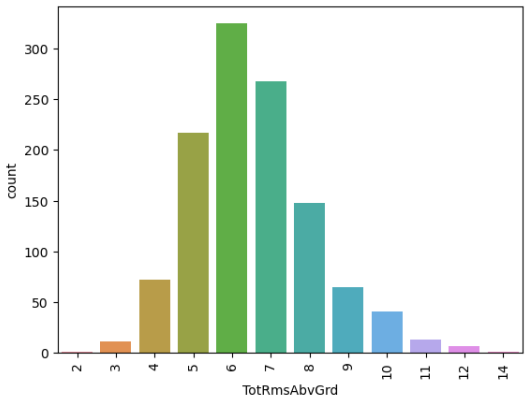

count
TA
Gd
No Fireplace
Fa
Ex
Po
FireplaceQu

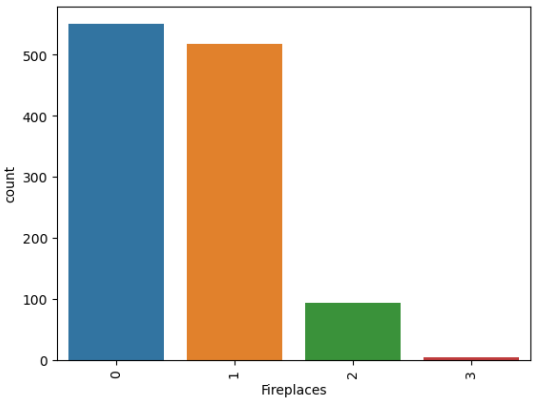

count
GarageType
Attchd
BuiltIn
Detchd
Basment
No Garage
2Types
CarPort

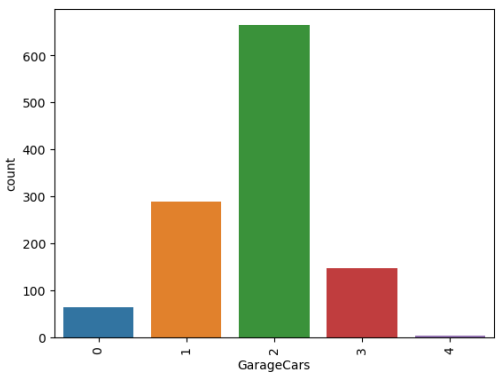

count
GarageFinish
RFn
Unf
Fin
No Garage

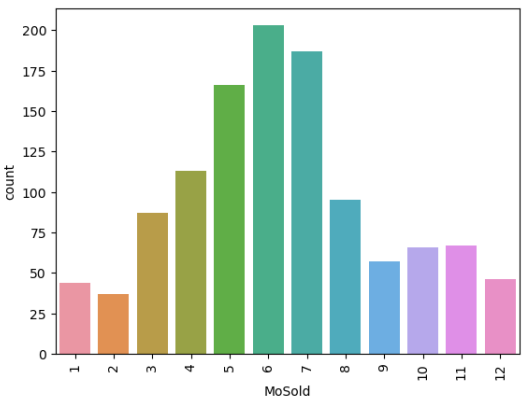

count
SaleCondition
Normal
Partial
Abnorml
Family
Alloca
AdjLand

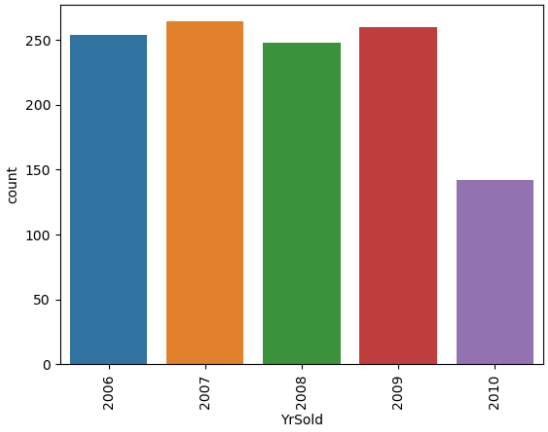

count
BedroomAbvGr

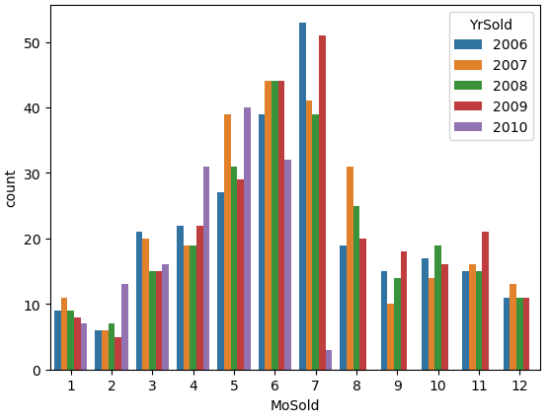

count
TotRmsAbvGrd

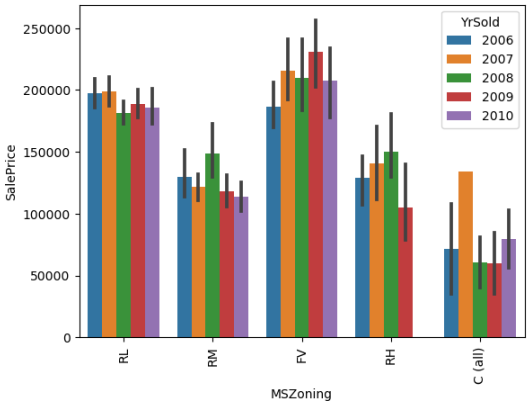

count
Fireplaces

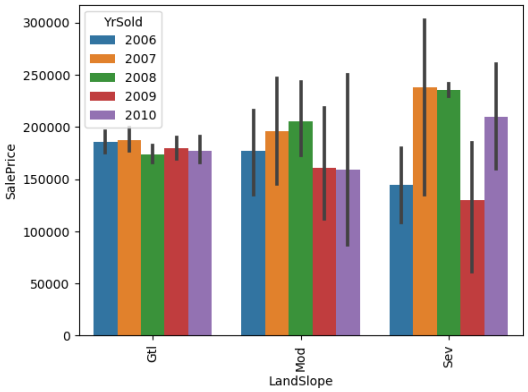

count
GarageCars

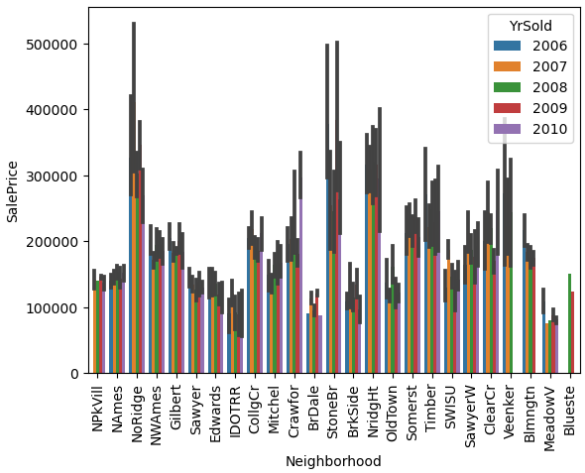

count
MoSold

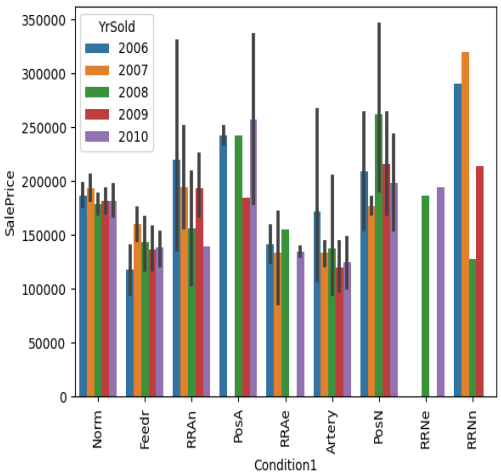

count
YrSold
2006
2007
2008
2009
2010

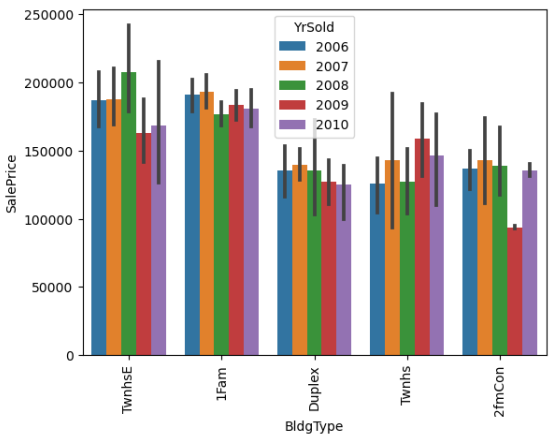

YrSold
2006
2007
2008
2009
2010
count
MoSold

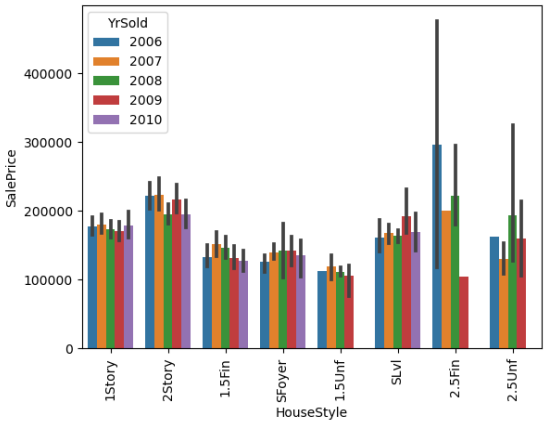

YrSold
2006
2007
2008
2009
2010
SalePrice
RL
RM
FV
RH
C (all)
MSZoning

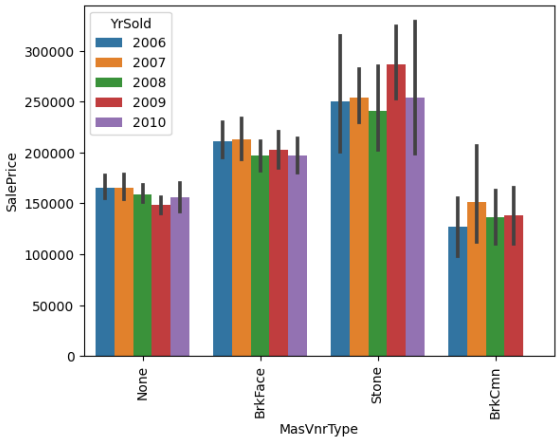

YrSold
2006
2007
2008
2009
2010
SalePrice
Gtl
Mod
Sev
LandSlope

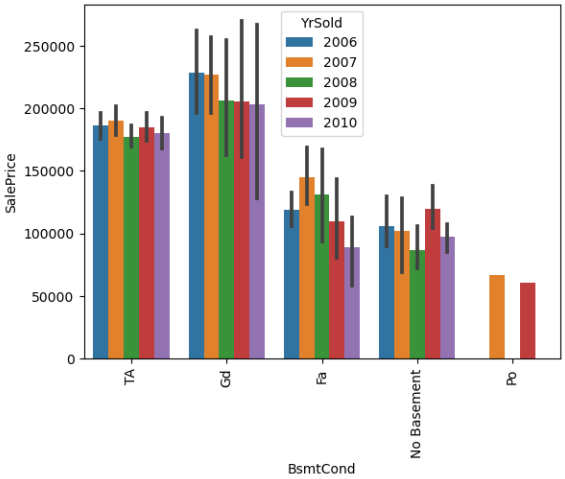

YrSold
2006
2007
2008
2009
2010
SalePrice
NPkVill
NAmes
NoRidge
NWAmes
Gilbert
Sawyer
Edwards
IDOTRR
CollgCr
Mitchel
Crawfor
BrDale
StoneBr
BrkSide
NridgHt
OldTown
Somerst
Timber
SWISU
SawyerW
ClearCr
Veenker
Blmngtn
MeadowV
Blueste
Neighborhood

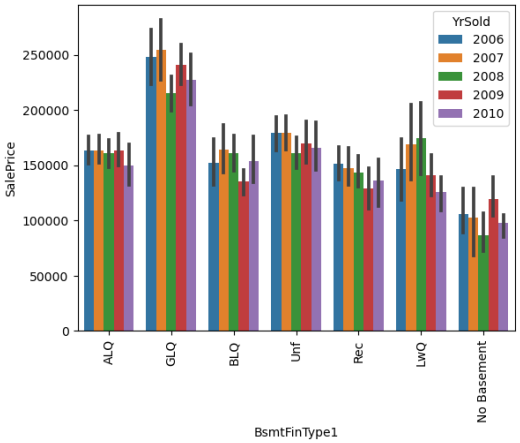

YrSold
2006
2007
2008
2009
2010
SalePrice
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1

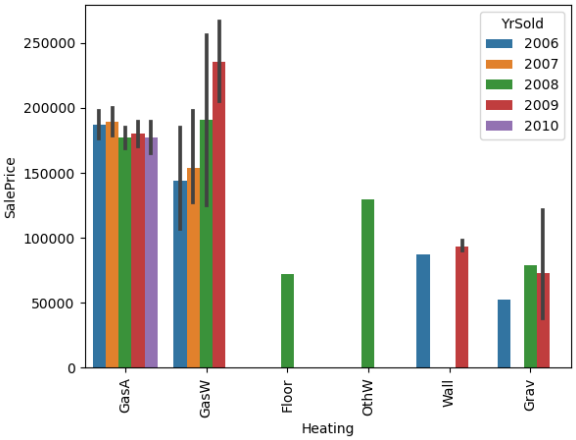

YrSold
2006
2007
2008
2009
2010
SalePrice
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType

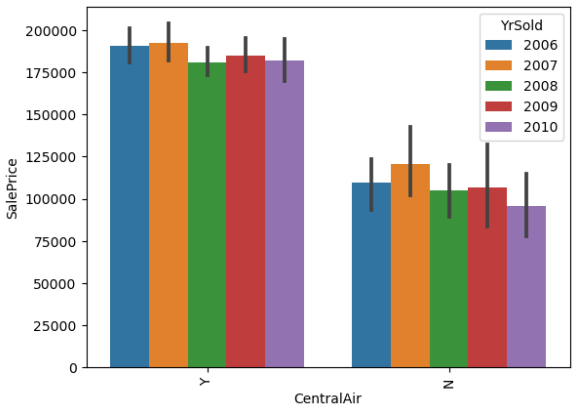

YrSold
2006
2007
2008
2009
2010
SalePrice
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle

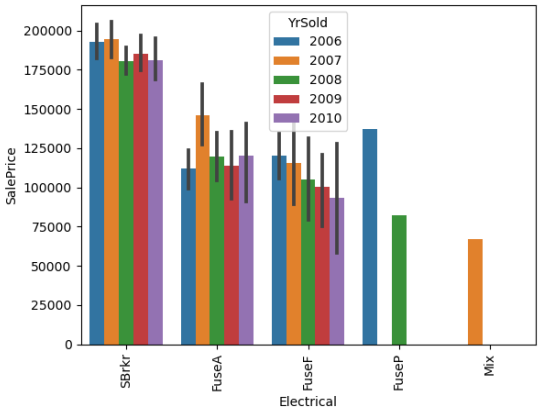

YrSold
2006
2007
2008
2009
2010
SalePrice
0
50000
100000
150000
200000
250000
300000
None
BrkFace
Stone
BrkCmn
MasVnrType

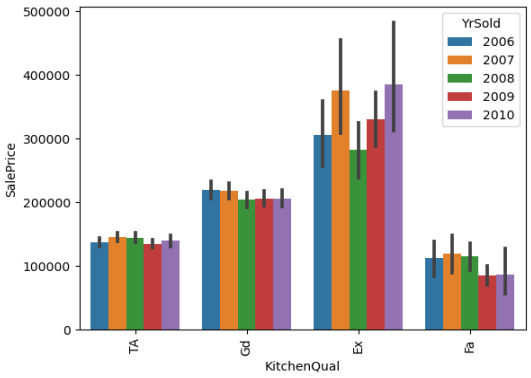

YrSold
2006
2007
2008
2009
2010
SalePrice
0
50000
100000
150000
200000
250000
TA
Gd
Fa
No Basement
Po
BsmtCond

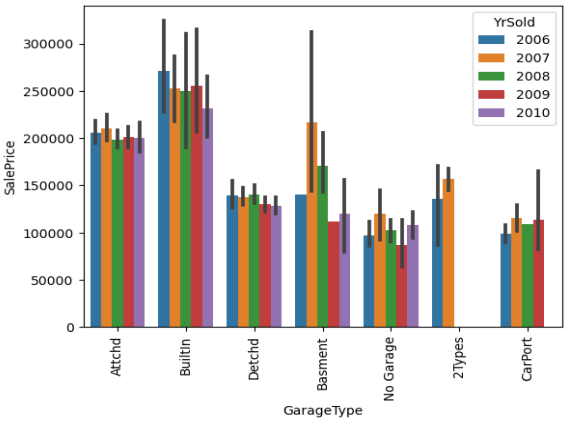

YrSold
2006
2007
2008
2009
2010
SalePrice
0
50000
100000
150000
200000
250000
ALQ
GLQ
BLQ
Unf
Rec
LwQ
No Basement
BsmtFinType1

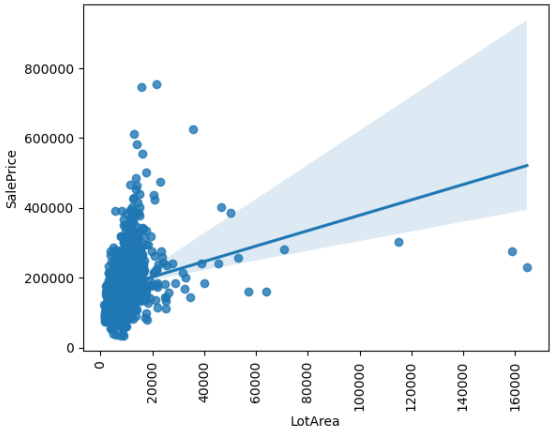

YrSold
2006
2007
2008
2009
2010
SalePrice
0
50000
100000
150000
200000
250000
GasA
GasW
Floor
OthW
Wall
Grav
Heating

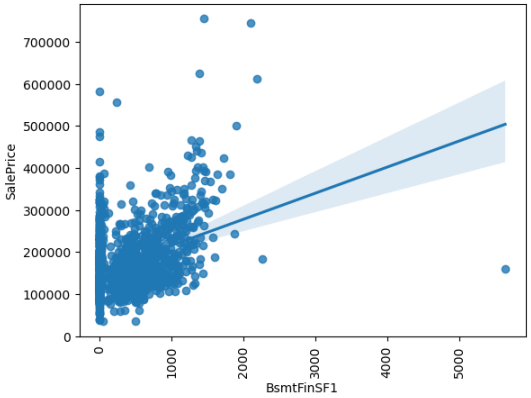

YrSold
2006
2007
2008
2009
2010
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
Y
N
CentralAir

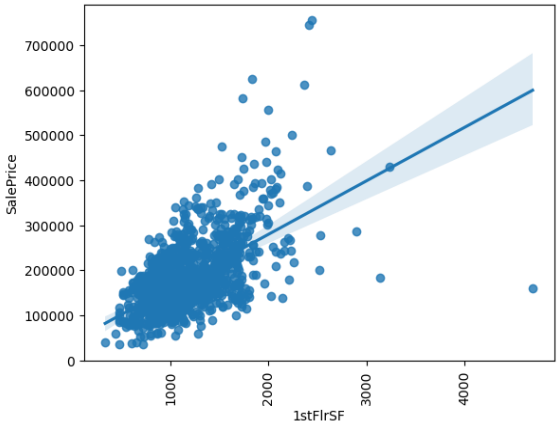

YrSold
2006
2007
2008
2009
2010
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

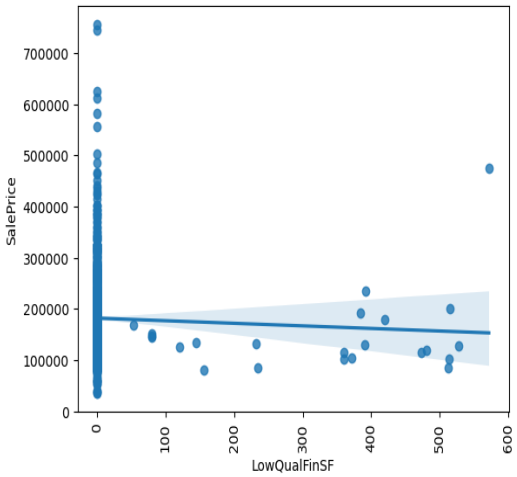

YrSold
2006
2007
2008
2009
2010
SalePrice
0
100000
200000
300000
400000
500000
TA
Gd
Ex
Fa
KitchenQual

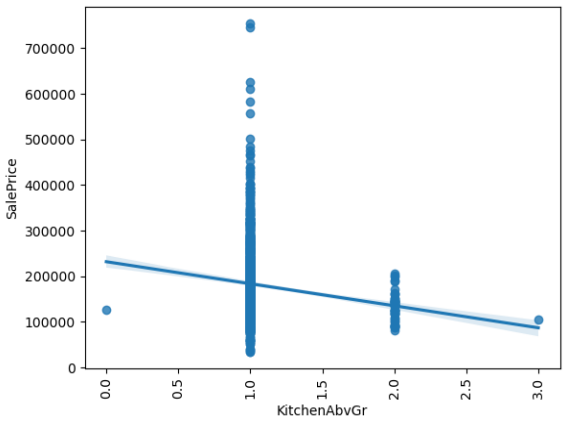

YrSold
2006
2007
2008
2009
2010
SalePrice
0
50000
100000
150000
200000
250000
300000
Attchd
BuiltIn
Detchd
Basment
No Garage
2Types
CarPort
GarageType

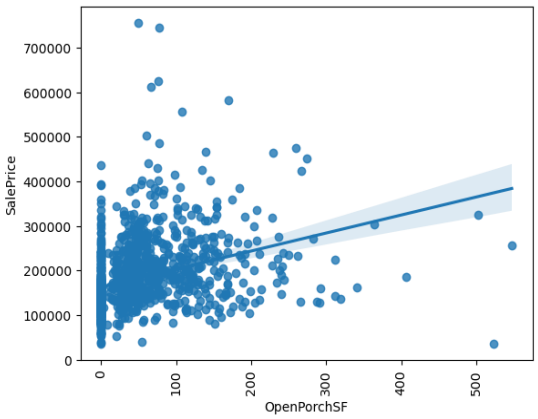

SalePrice
LotArea

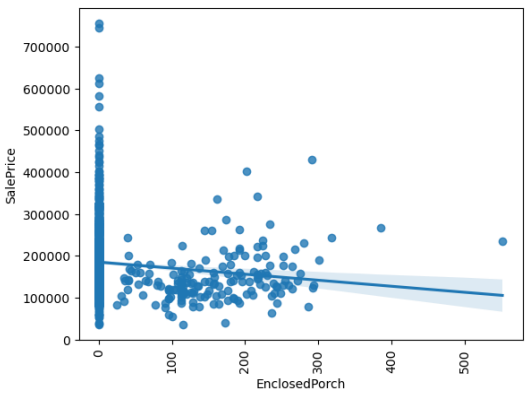

SalePrice
BsmtFinSF1

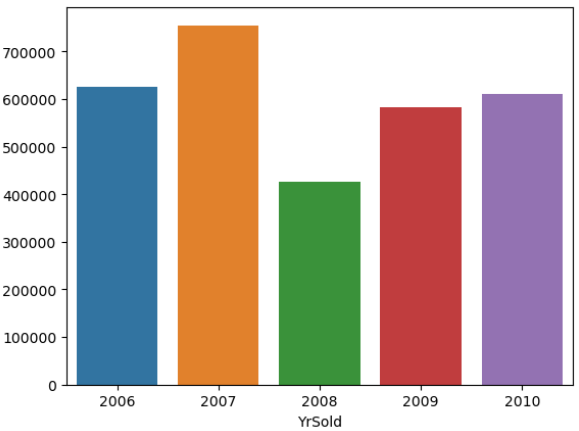

SalePrice
1stFlrSF

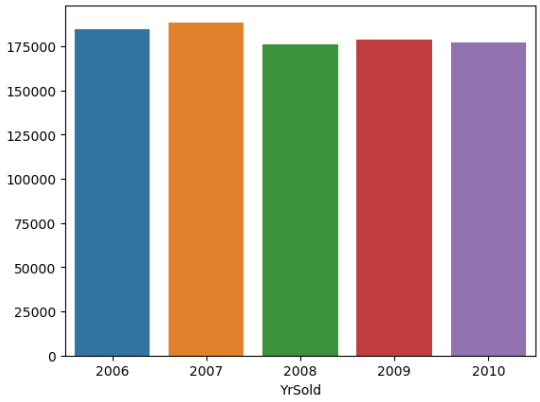

SalePrice
LowQualFinSF

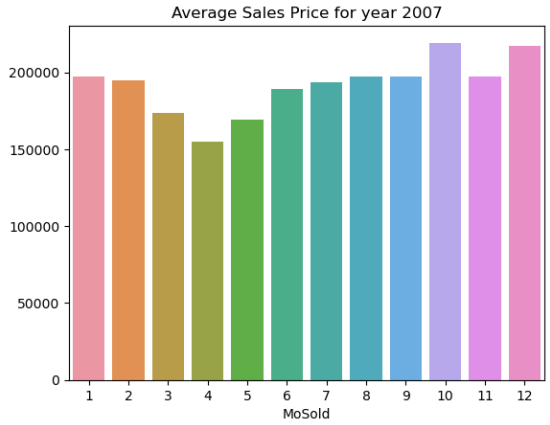

SalePrice
KitchenAbvGr

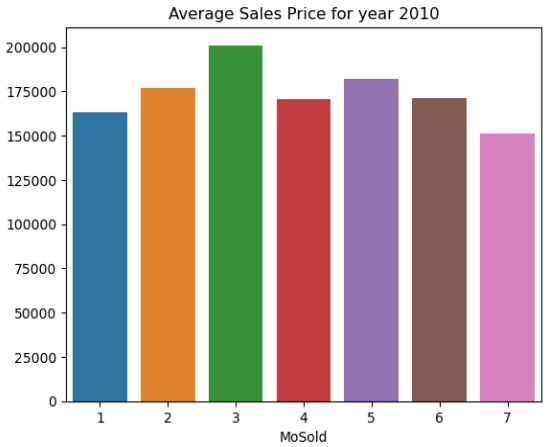

SalePrice
OpenPorchSF

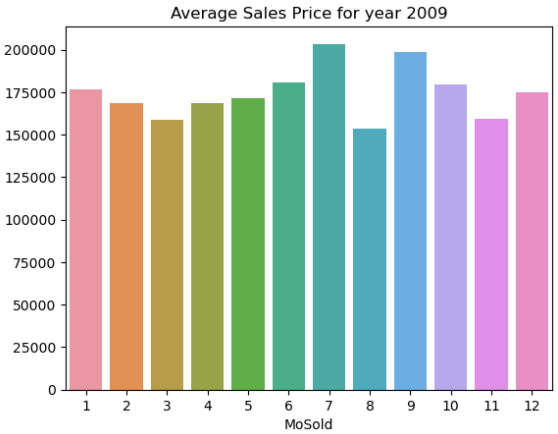

SalePrice
EnclosedPorch


2006
2007
2008
2009
2010
YrSold

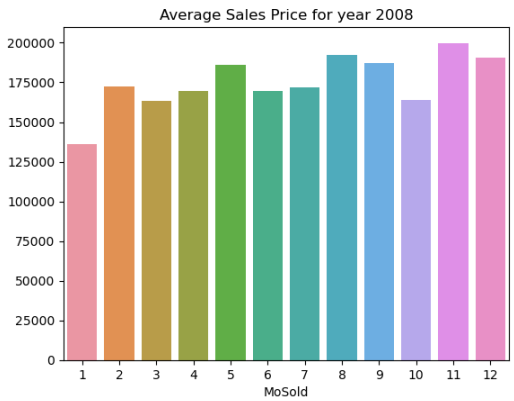

0
25000
50000
75000
100000
125000
150000
175000
2006
2007
2008
2009
2010
YrSold

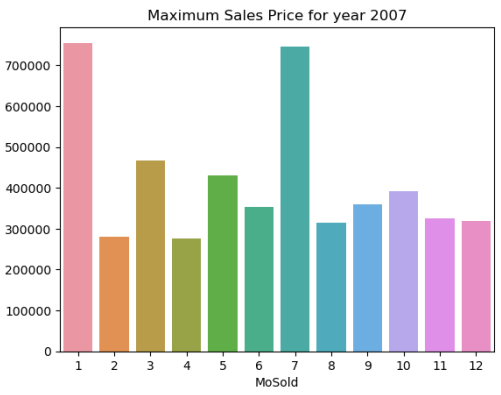

Average Sales Price for year 2007
0
50000
100000
150000
200000
1
2
3
4
5
6
7
8
9
10
11
12
MoSold

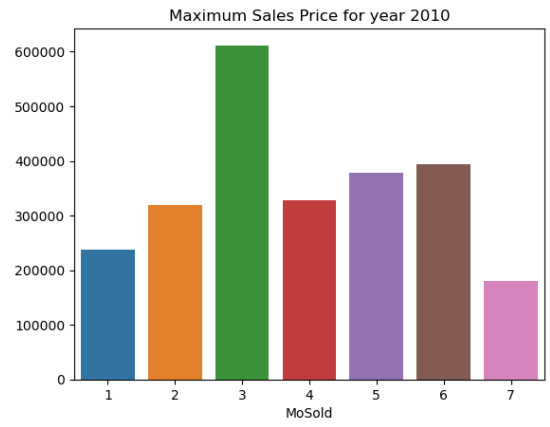

Average Sales Price for year 2010
0
25000
50000
75000
100000
125000
150000
175000
200000
1
2
3
4
5
6
7
MoSold

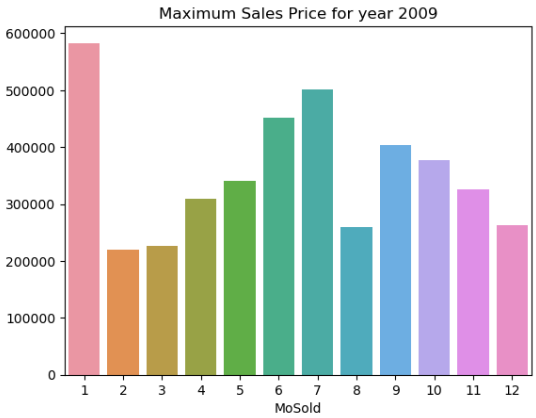

Average Sales Price for year 2009
0
25000
50000
75000
100000
125000
150000
175000
200000
1
2
3
4
5
6
7
8
9
10
11
12
MoSold

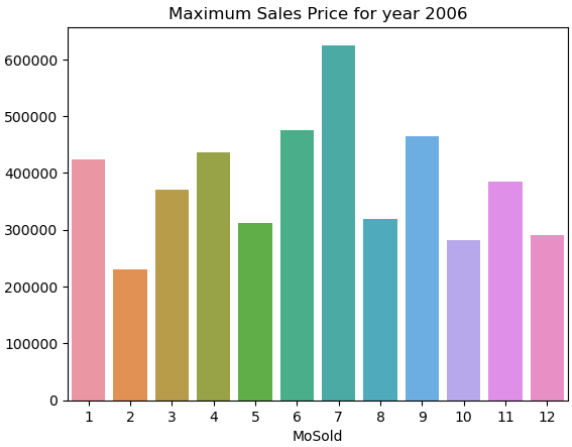

Average Sales Price for year 2006
0
50000
100000
150000
200000
1
2
3
4
5
6
7
8
9
10
11
12
MoSold


Average Sales Price for year 2008
0
25000
50000
75000
100000
125000
150000
175000
200000
1
2
3
4
5
6
7
8
9
10
11
12
MoSold

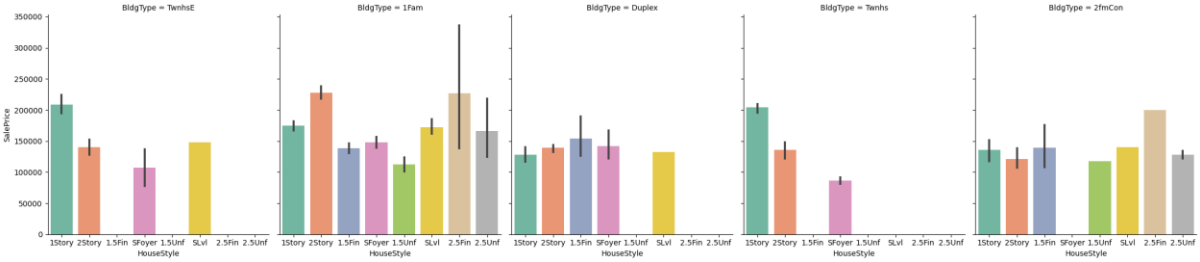

Maximum Sales Price for year 2007
0
100000
200000
300000
400000
500000
600000
700000
1
2
3
4
5
6
7
8
9
10
11
12
MoSold

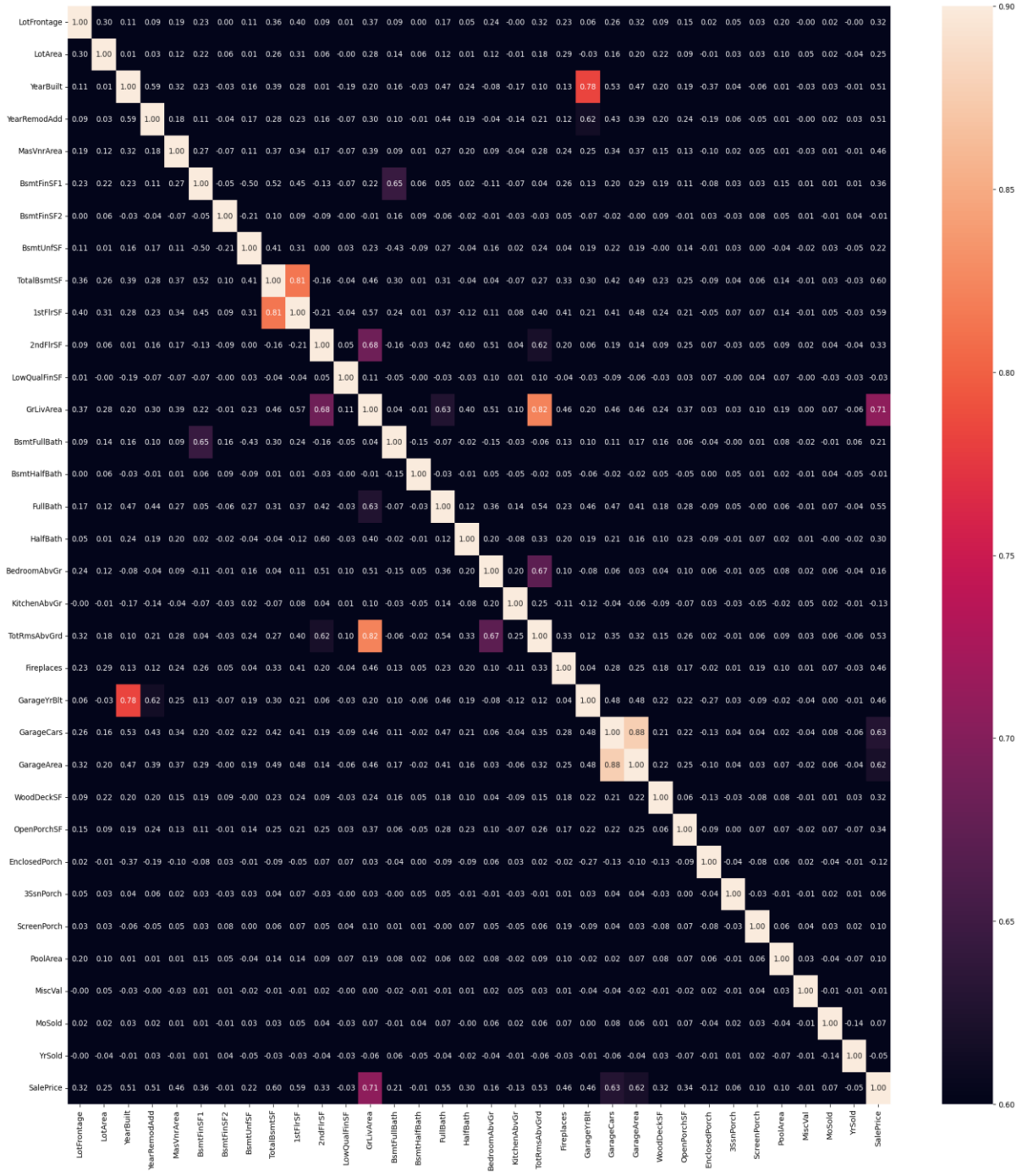

Maximum Sales Price for year 2010
0
100000
200000
300000
400000
500000
600000
1
2
3
4
5
6
7
MoSold


Maximum Sales Price for year 2009
600000
500000
400000
300000
200000
100000
0
1 2 3 4 5 6 7 8 9 10 11 12
MoSold

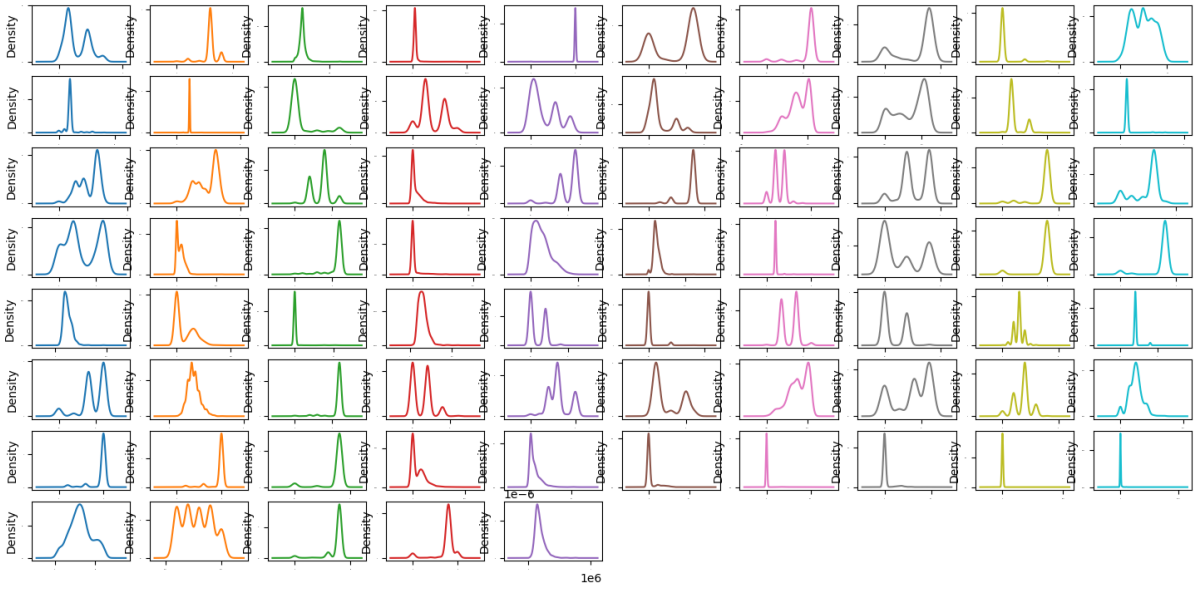

Maximum Sales Price for year 2006
600000
500000
400000
300000
200000
100000
0
1 2 3 4 5 6 7 8 9 10 11 12
MoSold


Maximum Sales Price for year 2008
400000
350000
300000
250000
200000
150000
100000
50000
0
1 2 3 4 5 6 7 8 9 10 11 12
MoSold


BldgType = TwnhsE
BldgType = 1Fam
BldgType = Duplex
BldgType = Twnhs
BldgType = 2fmCon
SalePrice
1Story 2Story 1.5Fin SFoyer 1.5Unf SLvl 2.5Fin 2.5Unf
HouseStyle



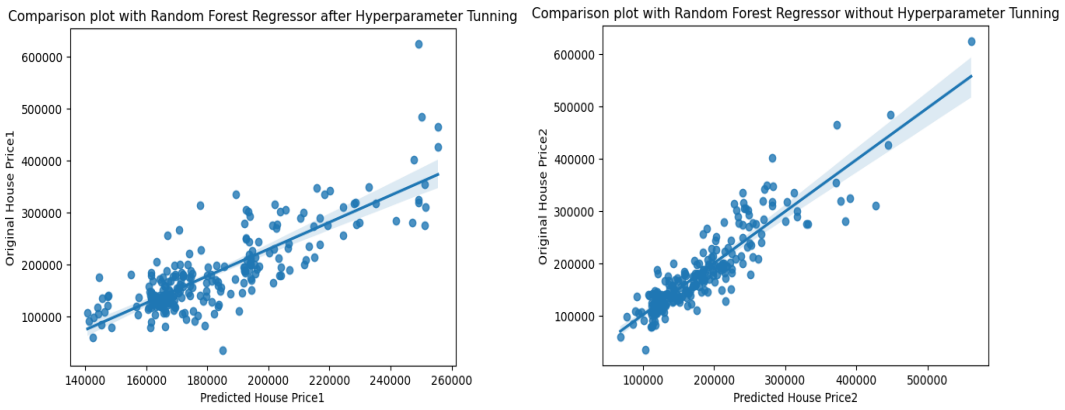
With the help of correlation matrix, we can identify the multicollinearity present in the data i.e., the features which are correlated to each other and then with variance inflation factor(vif) and correlation matrix we can decide on dropping the features which is less correlated between the two features.

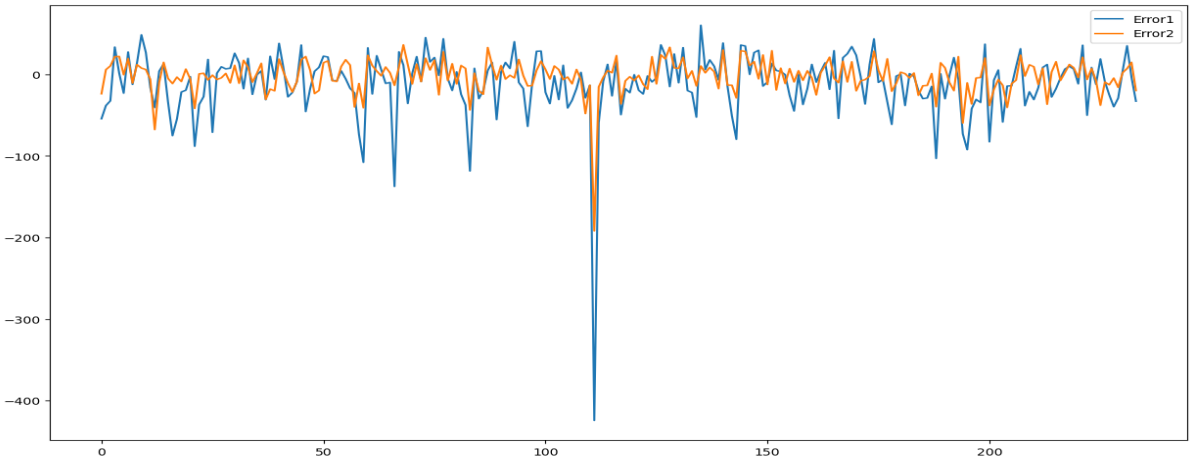


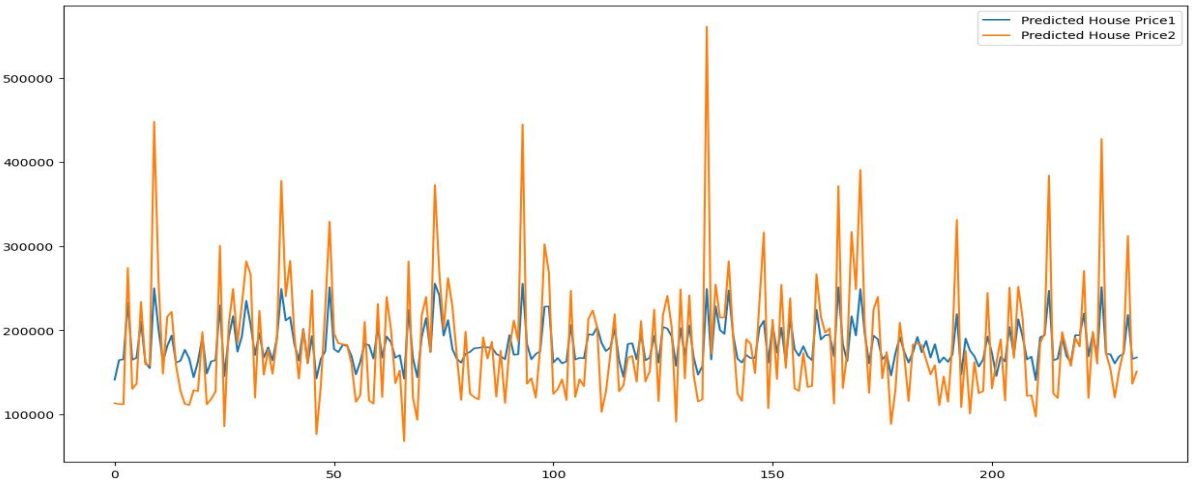
Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail.

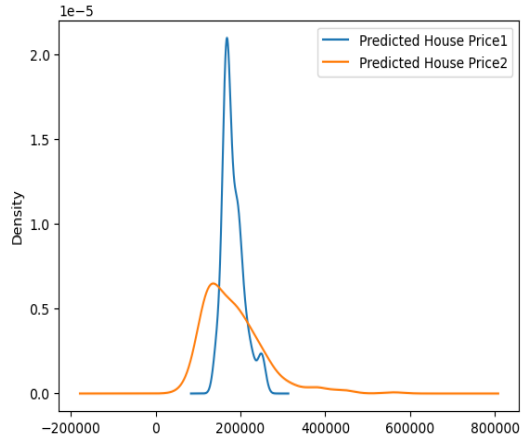
If different platforms were used, mention that as well.

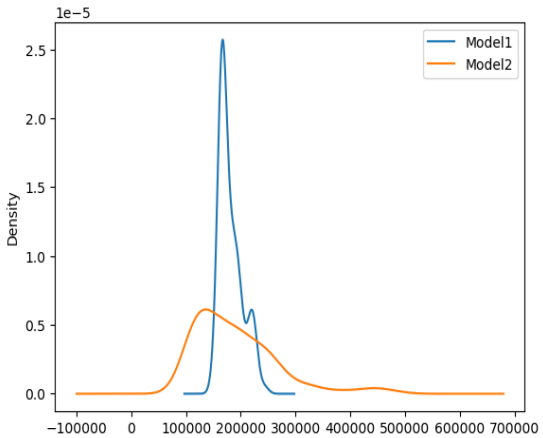
- **Interpretation of the Results**

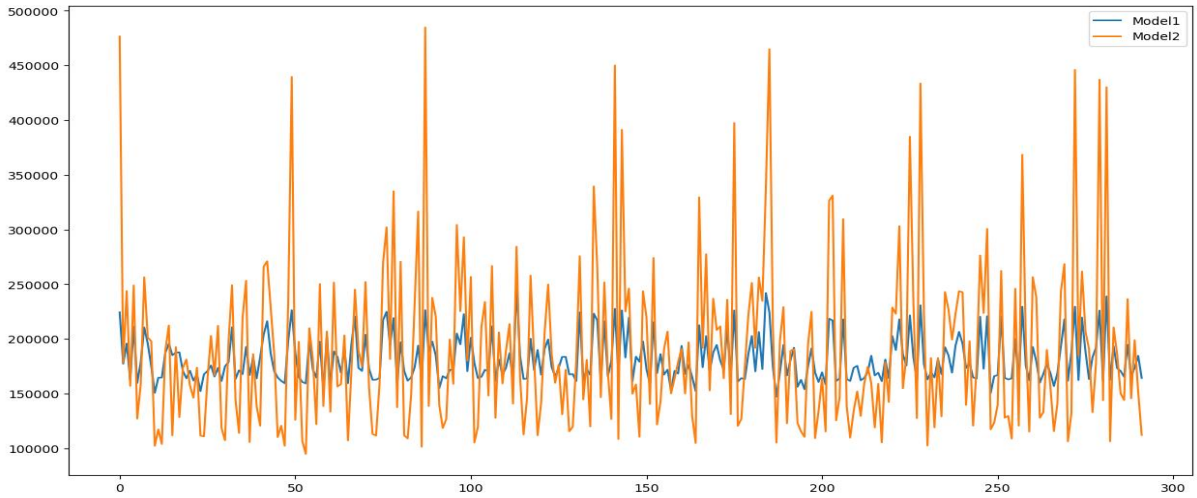



Error1
Error2
0
−100
−200
−300
−400
0
50
100
150
200


Predicted House Price1
Predicted House Price2
500000
400000
300000
200000
100000
0
50
100
150
200


1e−5
Predicted House Price1
Predicted House Price2
Density
2.0
1.5
1.0
0.5
0.0
−200000
0
200000
400000
600000
800000


1e−5
Model1
Model2
Density
2.5
2.0
1.5
1.0
0.5
0.0
−100000
0
100000
200000
300000
400000
500000
600000
700000



Give a summary of what results were interpreted from the visualizations, pre-processing and modelling.

CONCLUSION

- **Key Findings and Conclusions of the Study**
  - Count plot provided the count of each categorical value present in column which provided how the data is distributed.
  - These plots gave an idea how to replace the NaN values present in categorical features.
  - Bar plots were plotted for all categorical variables with respect to the target variable (sales price) for all the years and one can see that neighbourhood, Excellent kitchen quality, building type, house style with Two and one-half story: 2nd level finished, good basement condition etc play important role in the sales price house.
  - They later with the help of bar plot maximum sales and average sale each year and for every year sales every month. The Maximum sales happened to be in 2007.
  - Next, a categorical plot was plotted for house style building type for sales price, the sales most commonly occurred for building type one family.
  - After training, the random forest algorithm gave better results with accuracy of 81.34% for model without been

tuned using hyperparameter tunning. With tunned model the final accuracy id 41.43%.

- The accuracy was less because the outliers were present in the data. The data was getting reduced to half if we remove outliers.
- Later after pickling the predicted and original sales price was plotted to check the comparison of both the models with and without tunning. The model without tunning gave better results as one can see through the error plot. The kde plot gave us the range how the data is getting predicted, the tunned model spread was lower with high density and whereas the one without tunning has good spread for predicted the sales price.
- Lastly, the sales price was predicted for the test data which was provided after applying data pre-processing, feature engineering as done for the training data.

- **Learning Outcomes of the Study in respect of Data Science**
  - What I learned here is how one has to extensively look at the data and see its behaviour with help of EDA. This will help us identify the important features for predicting the target.
- **Limitations of this work and Scope for Future Work**
  - The data can be more explored for feature engineering with more literature review.
  - The Outliers present can be treated in a more efficient way to improve the accuracy of the model.
  - We can apply more algorithms along with hyper tunning parameters to improve the accuracy.