FLIP ROBO

HOUSE PRICE PREDICTION

Submitted by:

KONATALA MOHIT

ACKNOWLEDGMENT

I am grateful for the writer, developers and authority of the website pages machinelearningmastry, geekforgeeks, wikipedia and stackoverflow which have helped me to refer for whenever there was need for guidance. House specification and Real Estate Basics helped me to get a better understanding of the data to analyse and explore data. I am mostly grateful to DataTrained who have guided me in learning and enhancing my data science skills required to analyse and solve this project.

INTRODUCTION:

In this report I will be discussing about house price prediction using few of the machine learning models via python and its libraries. In today's era every individual has a keen interest in purchasing house as soon as one lands on a good job in their career. House price prediction helps us to understand how the real estate works i.e., to determine the value of the house property before acquiring or putting it up for a sale to get value for money while making a purchase and decent profit margin while selling the property.

Domain Knowledge in Real Estate, Material and Design Engineering has been used to understand and analysis the data. Minimum real estate knowledge is required to gets to know what are specifications of house and what are factor which affect its price. Material and design Engineering helps in decide the quality and luxury of house which assists in scaling the house price.

Review of Literature:

For the better understanding of the data referred to house building specification & real estate market and material & design engineering. In house building the mainframe is creating blueprint of the house property and to make the blueprint that is customed to the property layout, in hand land property helps in designing the blueprint of the house model. Material & design Engineering help in creating the good quality structure of the house, Architecture is used to enhance the luxury and comfort if worked along with civil design engineering. Real Estate is also necessary as it helps in selecting plot location, size and factors related to the land property which in return helps in estimating the value of the property.

Undertaken Problem:

Objective of the project is to predict the house price by analysis various factors that effect the house price outcome. House price prediction is one the classical problem in Data Science as it ticks all the basic fundamentals required to analyse data and creating a machine learning model to predict the necessary outcome.

Mathematical/ Analytical Modelling of the Problem:

Sale Price is the target data in out dataset and the variables present the target data continuous in form, therefore Regression Machine Models are used in this project. Regression is for estimating continuous form of data by using established relation between feature and target variables. Descriptive Analysis is used to study and observe the data.

Data Sources and their formats:

The source of the data is by the company name Surprise Housing. The data is provided in the CSV format. The provided data contains train and test data. Test data does not contain the Sale price column in it. Data contains 1460 entries each having 81 variables and 292 entries each having 80 variables in train data and test data respectively.

Data Pre-processing:

Data pre-processing has two main steps i.e., Data Cleaning and Data Transforming.

Data Cleaning:

Data Cleaning is one of the most important steps creating a machine model. If an uncleaned data is fed to a machine learning model, then the model will perform very poorly.

The and foremost step in data cleaning is removing and replacing null values in the dataset.

As we can see all the NAN values in the FireplaceQu have 0 as its correspondent element in Fireplace Columns, so we can replace it with Not available and also columns named ID, Alley, PoolQC, Fence, MiscFeature have way too much of null values so dropping them as replacing null values will make the data heavily biased

For the columns GarageType, GarageYrBlt, GarageFinish, GarageQual and GarageCond having NaN values have GarageArea zero which means there is no garage for these houses. Hence replacing the NaN as NA (Not Available).

For the columns BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1 and BsmtFinType2 having NaN values TotalBsmtArea zero which means there is no basement for these houses. Hence replacing the NaN as NA (Not Available)

As lotfrontage has more unique value replacing it with the mean value and round them to a whole number. MasVnrType has very fewer null values dropping the missing values

('Street','Utilities','LandSlope','Condition2','RoofMatl','ExterCond','BsmtCond','BsmtFinType2','Heating','CentralAir','Electrical','Functional','GarageQual','GarageCond','PavedDrive','BsmtHalfBath','KitchenAbvGr','PoolArea') removing these columns as one individual element out of all the unique elements containing in these columns is heavily biased which affect ML model negatively

And now as the null values are cured moving to the next data cleaning step i.e., arresting the noise in the data (outliers)

As all IR1, IR2, IR3 in lotshape vs saleprice are overlapping one another, concluding them as one element so house is either regular in shape ore irregular. Also, in BldgType column replacing all the elements except for 1fam into 2fam. In HouseStyle column combining unfinished and finished type into one and split type houses into one type. As most of the values are overlapping one another dividing OverallQual into 3 division (has 10 division due which the outliers exist). Making the ExterQual a Categorial columns that is element is either TA or Gd(depending on the overlapping of the value).

Making the BsmtFullBath, HalfBath and Fireplaces into categorial data as the saleprice are overlapping for bivariate analysis.

('LandContour','RoofStyle','Exterior2nd','FullBath','GarageYrBlt','GarageCars','SaleType') removing these below columns as one individual element out of all the unique elements containing in these columns is heavily which affect ML model negatively.

For outliers in these columns ('LotFrontage','LotArea','YearBuilt','MasVnrArea','BsmtFinSF1','BsmtUnfSF','TotalBsmtSF','1stFlrSF','2ndFlrSF','GrLivArea','BedroomAbvGr','TotRmsAbvGrd','OpenPorchSF','GarageArea','WoodDeckSF','SalePrice') using IQR method to arrest outliers.

('MSSubClass','LotFrontage','YearBuilt','ExterQual','BsmtQual', '1stFlrSF','2ndFlrSF','BsmtFinType1','TotRmsAbvGrd', BsmtUnfSF) dropping these columns as they have high corelation with other feature variable in the data

Data Transforming:

After Data Cleaning the data must be transformed into numerical form as one can't feed ordinal data to machine learning model and also the data must be normalized as normalizing the data the machine learning model gives equal importance to all the data.

In this project I am using label encoder to transform the ordinal data present in the data set into numerical form. Using Standard Scalar function to normalize the data.

Note: Before normalize the data separated the target variable from feature variables.

Data Inputs- Logic- Output Relationships:

After Separating the Input (feature) and output (target) data we split them into train and test division one part of the data is used to train the ML model and other part of the data to predict the output. When the train data is fed to machine learning model it generates an algorithm or simply put an equation that is applicable to all the data and when test data is fed to it implements the trained data equation to the current input values to predict the outcome. If the input has no outliers and is clean that there is no over or underfitting in the outcome.

Hardware and Software Requirements and Tools Used:

Hardware Required for Jupyter Notebook Software is as follows:

Memory and disk space required per user: 1GB RAM + 1GB of disk + .5 CPU core.

Server overhead: 2-4GB or 10% system overhead (whatever is larger), .5 CPU cores, 10GB disk space.

Port requirements: Port 8000 plus 5 unique, random ports per notebook

Libraries Used:

Pandas library: To frame raw data, visualize and perform task on it via other libraries.

Numpy library: To perform mathematical functions on the framed data numpy is used. In this project used it to location nan values and replace them with desired value also to find mean and standard values.

Matplotlib library: This library is used to visualize the data. Used to visualize univariate and bivariate analysis (pie plot, count plot and scatter plot) also to visualize outliers via box plot.

Seaborn library: heatmap to see co-relation between feature variable to arrest high collinearity.

Sklearn library: Imported this library to normalize the data, split the data into train and test data, various machine learning model and cross validation techniques.

Warnings library: To ignore filter warning shown while compiling block of codes

Pickle: To save the trained machine learning model.

Algorithm Used:

1. K-Neighbors Regressor
2. XGB Regression
3. Random Forest Regressor

4. Decision Tree Regressor
5. Gradient Boosting Regressor

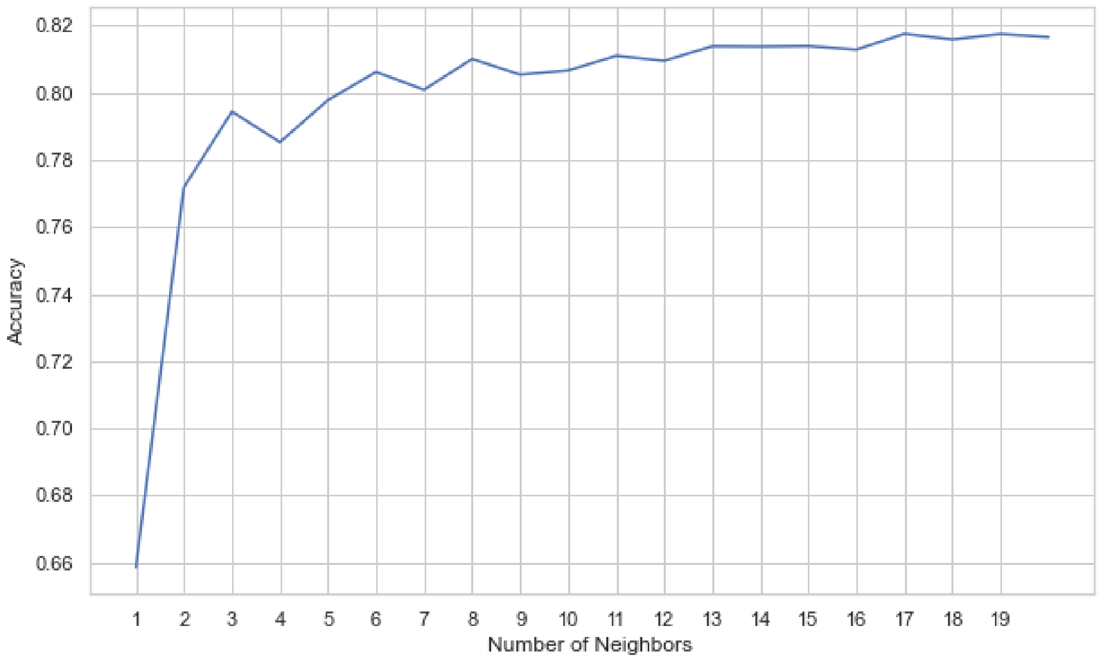
Model Training and Selection:

K-Neighbors Regressor:

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same *neighbourhood.*

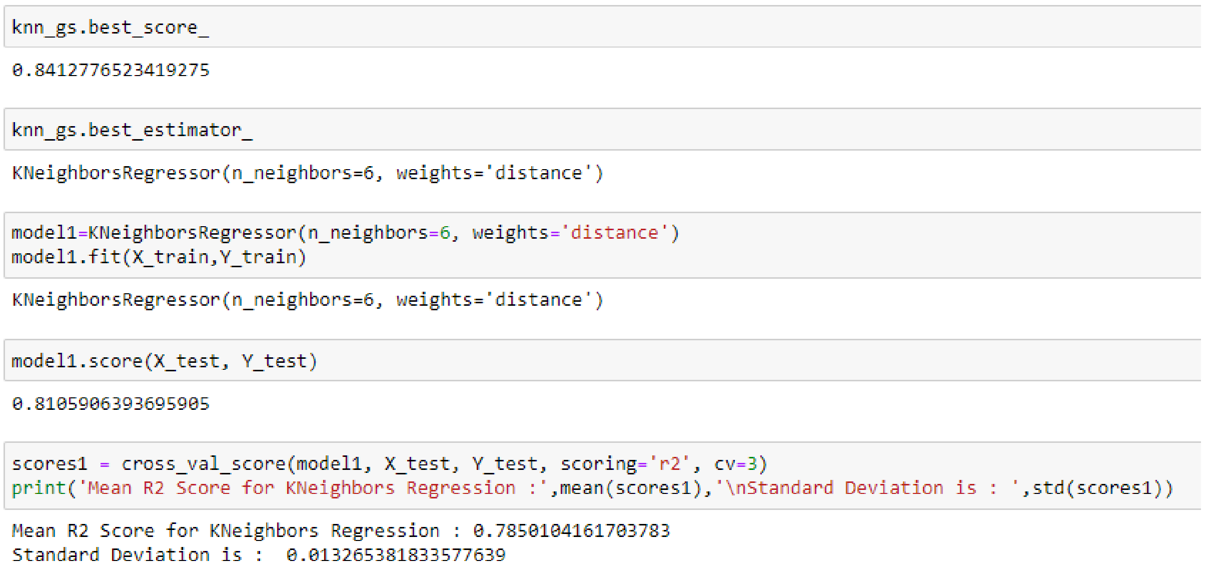
```
knn_p=KNeighborsRegressor()
mean_acc = np.zeros(20)
for i in range(1,21):
    #Train Model and Predict
    knn = KNeighborsRegressor(n_neighbors = i).fit(X_train,Y_train)
    yhat2= knn.predict(X_test)
    mean_acc[i-1] = metrics.r2_score(Y_test, yhat2)

loc = np.arange(1,20,step=1.0)
plt.figure(figsize = (10, 6))
plt.plot(range(1,21), mean_acc)
plt.xticks(loc)
plt.xlabel('Number of Neighbors ')
plt.ylabel('Accuracy')
plt.show()
```



```
knn=KNeighborsRegressor()
para={
        'n_neighbors':[2,3,4,5,6,9,10,11,14,18,19,20],
        'weights':['uniform', 'distance'],
        'algorithm':['auto', 'ball_tree', 'kd_tree', 'brute','auto'],
        'leaf_size':[30,40,50],
        'p':[2],
        'metric':['minkowski']

}
knn_gs= gs(estimator =knn, param_grid=para,cv=5, n_jobs=10)
knn_gs.fit(X_train,Y_train)
```



```
knn_gs.best_score_
```

```
0.8412776523419275
```

```
knn_gs.best_estimator_
```

```
KNeighborsRegressor(n_neighbors=6, weights='distance')
```

```
model1=KNeighborsRegressor(n_neighbors=6, weights='distance')
model1.fit(X_train,Y_train)
```

```
KNeighborsRegressor(n_neighbors=6, weights='distance')
```

```
model1.score(X_test, Y_test)
```

```
0.8105906393695905
```

```
scores1 = cross_val_score(model1, X_test, Y_test, scoring='r2', cv=3)
print('Mean R2 Score for KNeighbors Regression :',mean(scores1),'\nStandard Deviation is : ',std(scores1))
```

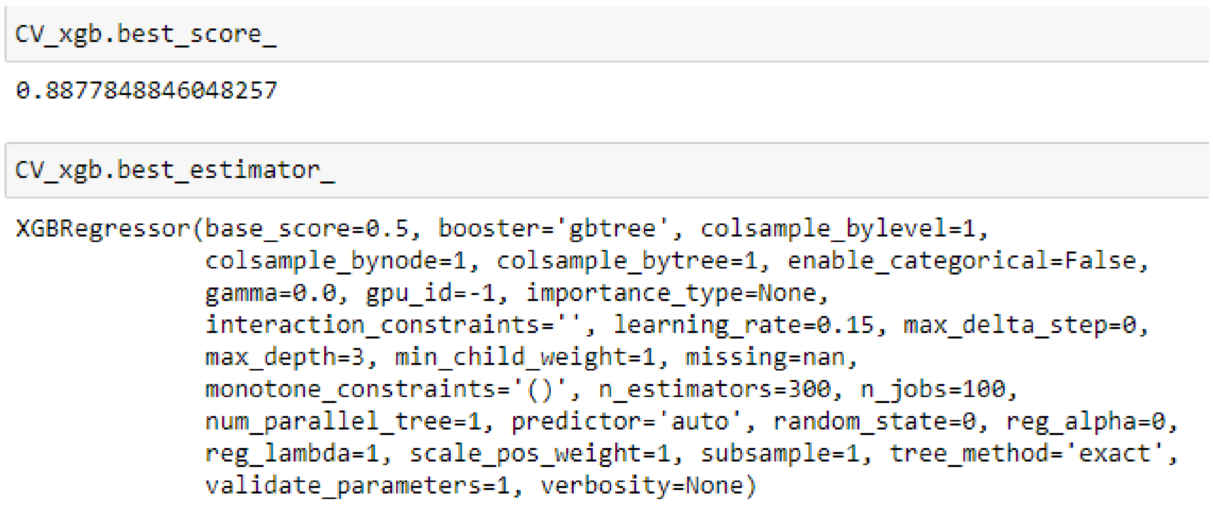
```
Mean R2 Score for KNeighbors Regression : 0.7850104161703783
Standard Deviation is :  0.013265381833577639
```

Mean R2 Score for K-Neighbors Regression: 0.7850104161703783
Standard Deviation is: 0.013265381833577639

XGB Regressor:

XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XGBoost is a popular supervised machine learning model with characteristics like computation speed, parallelization, and performance

```
xgb=XGBRegressor()
param_grid = {
    'n_estimators': [250,300,350],
    'max_depth': [3, 4, 5, 6, 8,10],
    'learning_rate':[0.1,0.15,0.2,0.3],
    'gamma':[0.0,0.1,0.2,0.3],
    'n_jobs':[100,200,300]
}
CV_xgb = gs(xgb,param_grid=param_grid,scoring ='r2',cv=5,verbose=5)
CV_xgb.fit(X_train,Y_train)
```

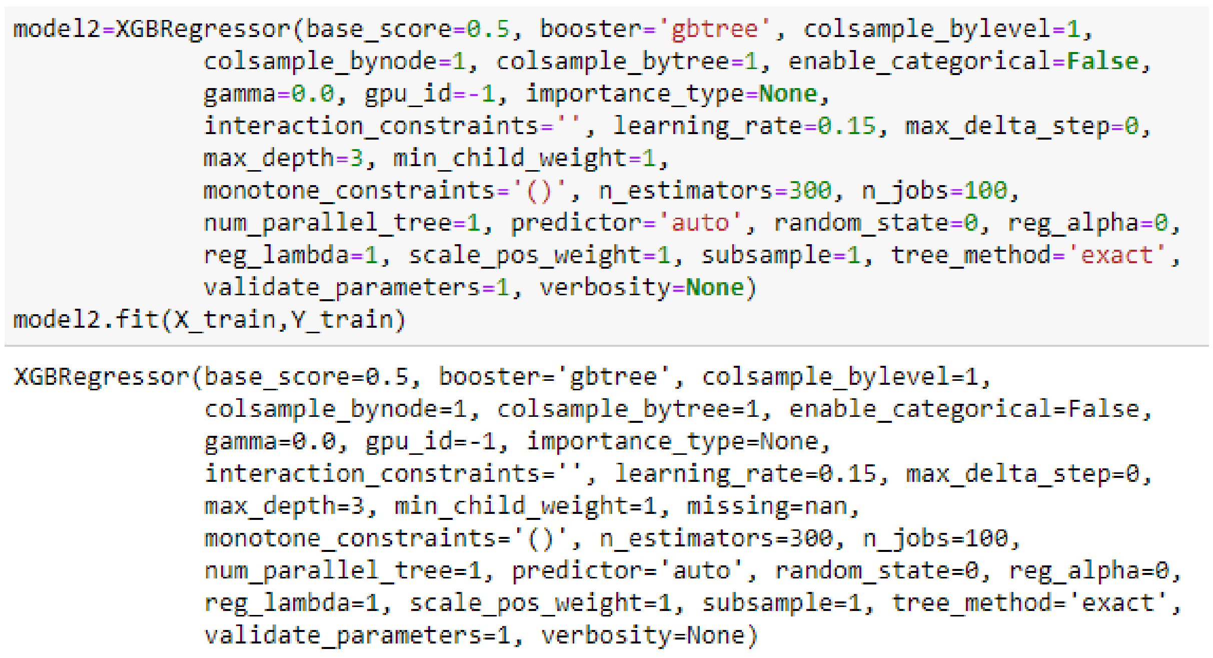


```
CV_xgb.best_score_
```

```
0.8877848846048257
```

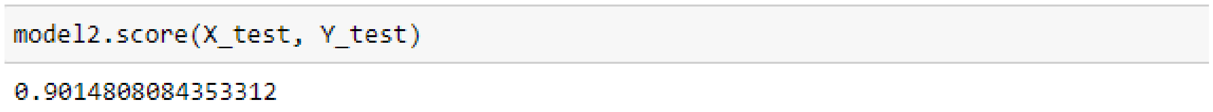
```
CV_xgb.best_estimator_
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0.0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.15, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=300, n_jobs=100,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```



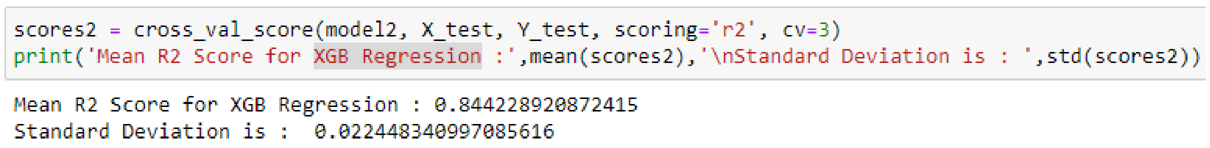
```
model2=XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0.0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.15, max_delta_step=0,
             max_depth=3, min_child_weight=1,
             monotone_constraints='()', n_estimators=300, n_jobs=100,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
model2.fit(X_train,Y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0.0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.15, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=300, n_jobs=100,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```



```
model2.score(X_test, Y_test)
```

```
0.9014808084353312
```



```
scores2 = cross_val_score(model2, X_test, Y_test, scoring='r2', cv=3)
print('Mean R2 Score for XGB Regression :',mean(scores2),'\nStandard Deviation is : ',std(scores2))
```

```
Mean R2 Score for XGB Regression : 0.844228920872415
Standard Deviation is :  0.022448340997085616
```

Mean R2 Score for XGB Regression : 0.844228920872415
Standard Deviation is : 0.022448340997085616

Random Forest Regressor:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting

```
rfr= RandomForestRegressor()
parameter_rfr={
          'n_estimators': [200,250,300,350],
          'max_features': ['auto', 'sqrt'],
          'max_depth': [2,4,6,8],
          'min_samples_split': [2, 5, 10],
          'min_samples_leaf': [1, 2,3,4,5],
          'bootstrap': [True, False]

}
rfr_gs= gs(estimator =rfr, param_grid=parameter_rfr,cv=5,verbose=3,n_jobs=5)
rfr_gs.fit(X_train,Y_train)
```

```
Fitting 5 folds for each of 960 candidates, totalling 4800 fits

GridSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=5,
             param_grid={'bootstrap': [True, False], 'max_depth': [2, 4, 6, 8],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 3, 4, 5],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [200, 250, 300, 350]},
             verbose=3)
```

```
rfr_gs.best_score_
```

```
0.8773801350020094
```

```
rfr_gs.best_estimator_
```

```
RandomForestRegressor(bootstrap=False, max_depth=8, max_features='sqrt',
                      n_estimators=200)
```

```
model3=RandomForestRegressor(bootstrap=False, max_depth=8, max_features='sqrt',
                      n_estimators=200)
model3.fit(X_train,Y_train)
```

```
RandomForestRegressor(bootstrap=False, max_depth=8, max_features='sqrt',
                      n_estimators=200)
```

```
model3.score(X_test, Y_test)
```

```
0.8759288276217723
```

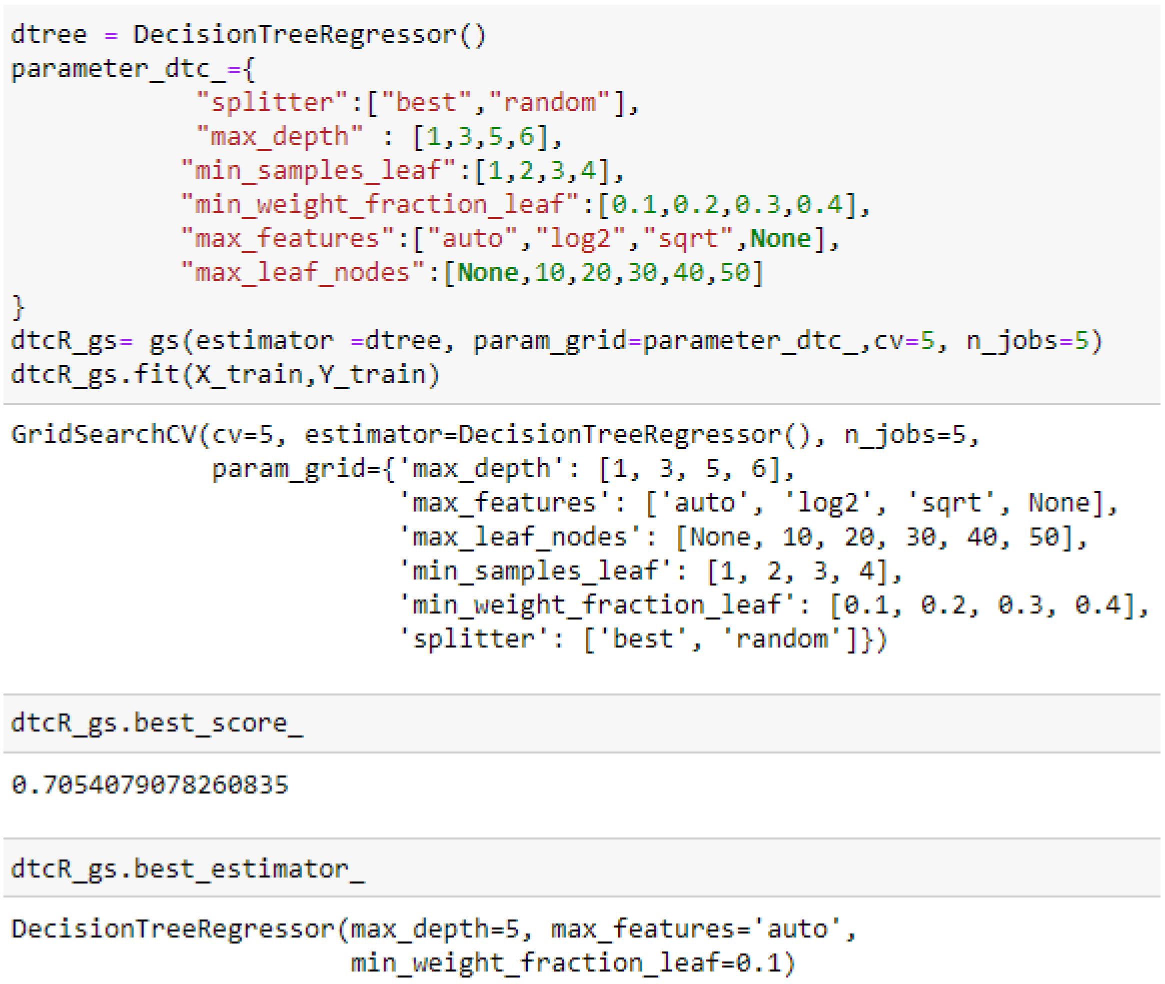
```
scores3 = cross_val_score(model3, X_test, Y_test, scoring='r2', cv=3)
print('Mean R2 Score for RandomForest :',mean(scores3),'\nStandard Deviation is : ',std(scores3))
```

```
Mean R2 Score for RandomForest : 0.8300785336375308
Standard Deviation is :  0.014663867153028372
```

Mean R2 Score for RandomForest : 0.8300785336375308
Standard Deviation is : 0.014663867153028372

Decision Tree Regressor:

The general regression tree building methodology allows input variables to be a mixture of continuous and categorical variables. A decision tree is generated when each decision node in the tree contains a test on some input variable's value. The terminal nodes of the tree contain the predicted output variable values.



```
dtree = DecisionTreeRegressor()
parameter_dtc_={
            "splitter":["best","random"],
            "max_depth" : [1,3,5,6],
           "min_samples_leaf":[1,2,3,4],
           "min_weight_fraction_leaf":[0.1,0.2,0.3,0.4],
           "max_features":["auto","log2","sqrt",None],
           "max_leaf_nodes":[None,10,20,30,40,50]
}
dtcR_gs= gs(estimator =dtree, param_grid=parameter_dtc_,cv=5, n_jobs=5)
dtcR_gs.fit(X_train,Y_train)
```

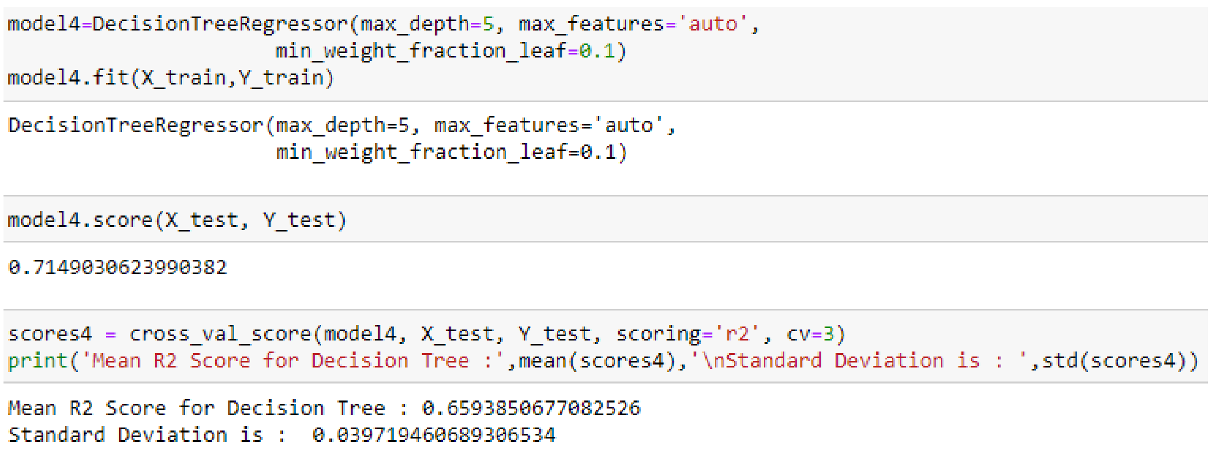
```
GridSearchCV(cv=5, estimator=DecisionTreeRegressor(), n_jobs=5,
             param_grid={'max_depth': [1, 3, 5, 6],
                         'max_features': ['auto', 'log2', 'sqrt', None],
                         'max_leaf_nodes': [None, 10, 20, 30, 40, 50],
                         'min_samples_leaf': [1, 2, 3, 4],
                         'min_weight_fraction_leaf': [0.1, 0.2, 0.3, 0.4],
                         'splitter': ['best', 'random']})
```

```
dtcR_gs.best_score_
```

```
0.7054079078260835
```

```
dtcR_gs.best_estimator_
```

```
DecisionTreeRegressor(max_depth=5, max_features='auto',
                      min_weight_fraction_leaf=0.1)
```



```
model4=DecisionTreeRegressor(max_depth=5, max_features='auto',
                      min_weight_fraction_leaf=0.1)
model4.fit(X_train,Y_train)
```

```
DecisionTreeRegressor(max_depth=5, max_features='auto',
                      min_weight_fraction_leaf=0.1)
```

```
model4.score(X_test, Y_test)
```

```
0.7149030623990382
```

```
scores4 = cross_val_score(model4, X_test, Y_test, scoring='r2', cv=3)
print('Mean R2 Score for Decision Tree :',mean(scores4),'\nStandard Deviation is : ',std(scores4))
```

```
Mean R2 Score for Decision Tree : 0.6593850677082526
Standard Deviation is :  0.039719460689306534
```

Mean R2 Score for Decision Tree : 0.6593850677082526

Standard Deviation is : 0.039719460689306534

Gradient Boosting Regressor:
Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function

```
gbr=GradientBoostingRegressor()
parameter_gbr_={
         'n_estimators': [200,250,300,350,400],
         'max_depth': [2,4,6,8],
         'min_samples_split': [2,4,5,6],
         "min_samples_leaf":[1,2,3,4,5],
     'learning_rate':[0.1,0.15,0.2,0.3],
}
gbr_gs= gs(estimator =gbr, param_grid=parameter_gbr_,cv=5, n_jobs=5)
gbr_gs.fit(X_train,Y_train)
```

```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(), n_jobs=5,
             param_grid={'learning_rate': [0.1, 0.15, 0.2, 0.3],
                         'max_depth': [2, 4, 6, 8],
                         'min_samples_leaf': [1, 2, 3, 4, 5],
                         'min_samples_split': [2, 4, 5, 6],
                         'n_estimators': [200, 250, 300, 350, 400]})
```

```
gbr_gs.best_score_
```

```
0.8892234343902599
```

```
gbr_gs.best_estimator_
```

```
GradientBoostingRegressor(max_depth=4, min_samples_leaf=3, min_samples_split=5,
                          n_estimators=200)
```

```
model5=GradientBoostingRegressor(max_depth=4, min_samples_leaf=3, min_samples_split=5,
                          n_estimators=200)
model5.fit(X_train,Y_train)
```

```
GradientBoostingRegressor(max_depth=4, min_samples_leaf=3, min_samples_split=5,
                          n_estimators=200)
```

```
model5.score(X_test, Y_test)
```

```
0.9055888612847679
```

```
scores5 = cross_val_score(model5, X_test, Y_test, scoring='r2', cv=3)
print('Mean R2 Score for Gradient Boosting :',mean(scores5),'\nStandard Deviation is : ',std(scores5))
```

```
Mean R2 Score for Gradient Boosting : 0.8438019285583485
Standard Deviation is :  0.017938673170569304
```

Metrics Used in the project:

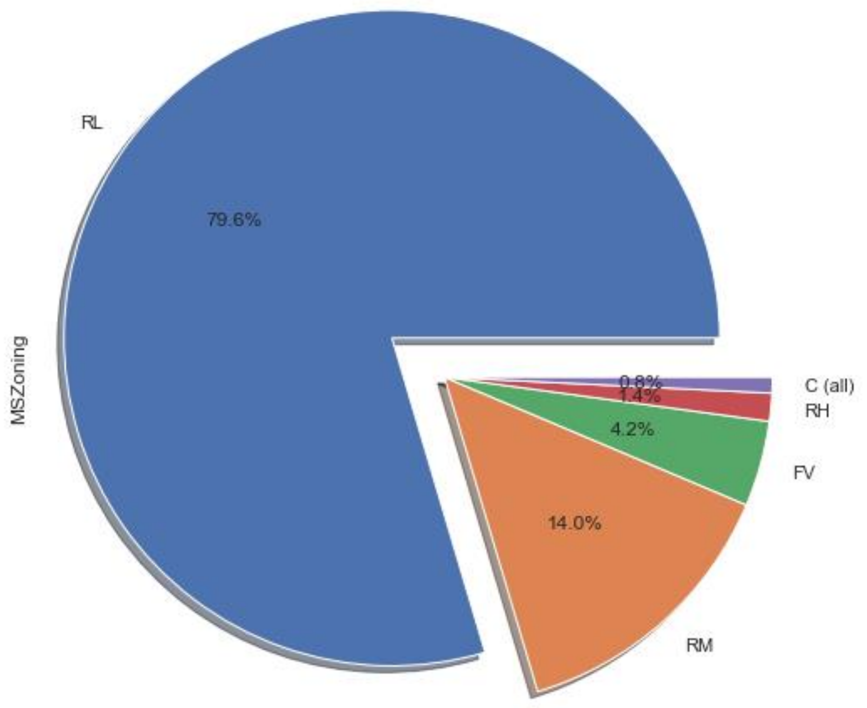
cross_val_score: Used cross_val_score to evaluate the model and observe (r2 value) it perform for particular number of fold to determine which model performs better

mean: To get the mean value of cross validation score and determine the model accuracy.

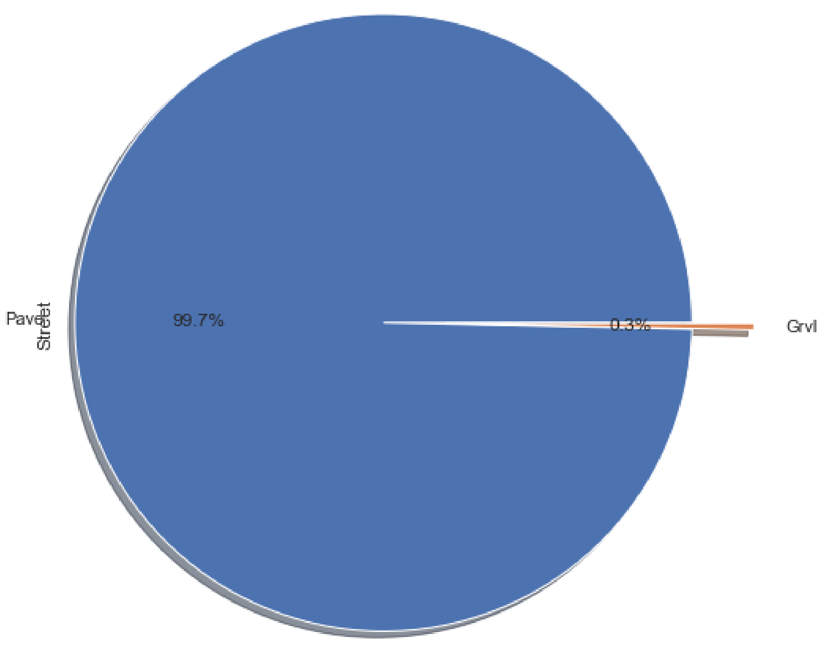
Standard deviation: To determine the average deviation for all the fold observed in cross validation scores

Visualization:

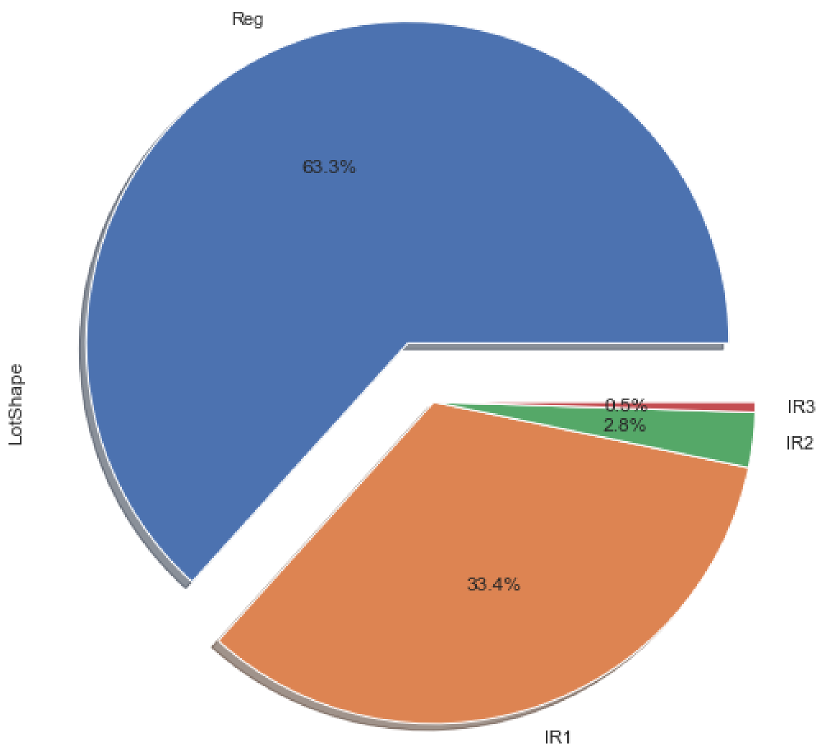
MSZoning



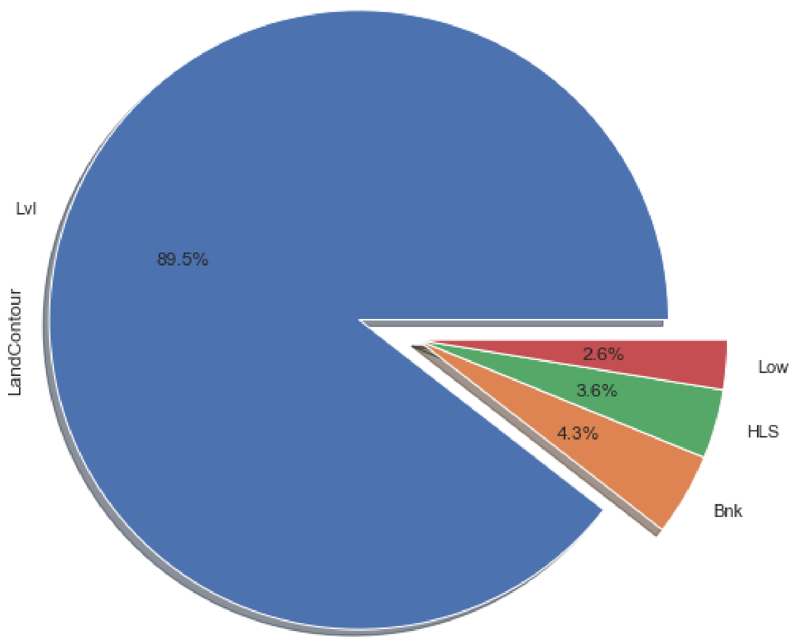
Street



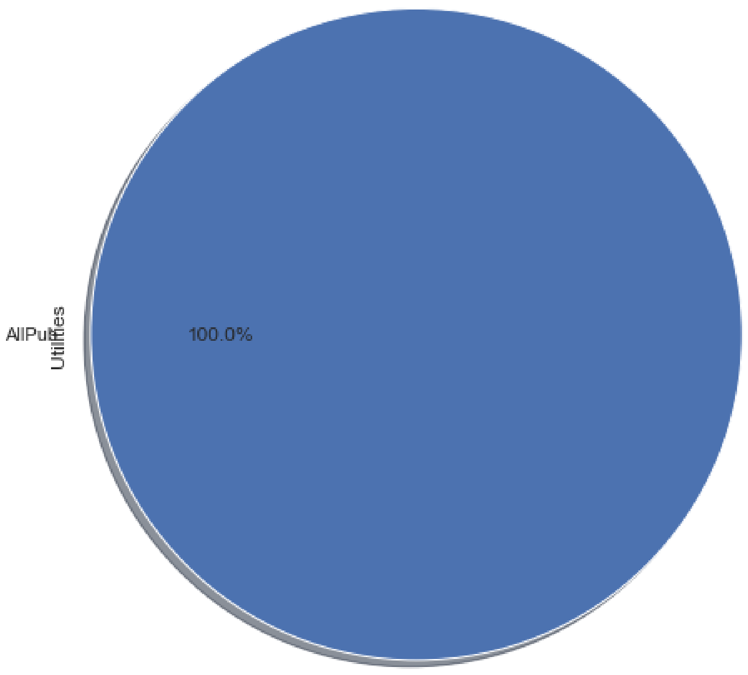
LotShape



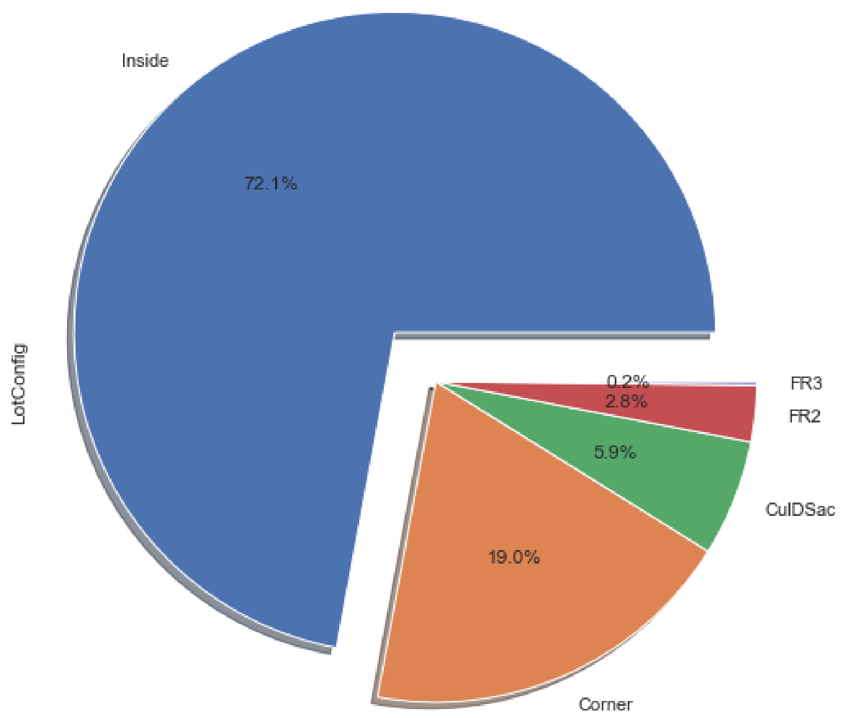
```
LandContour
```



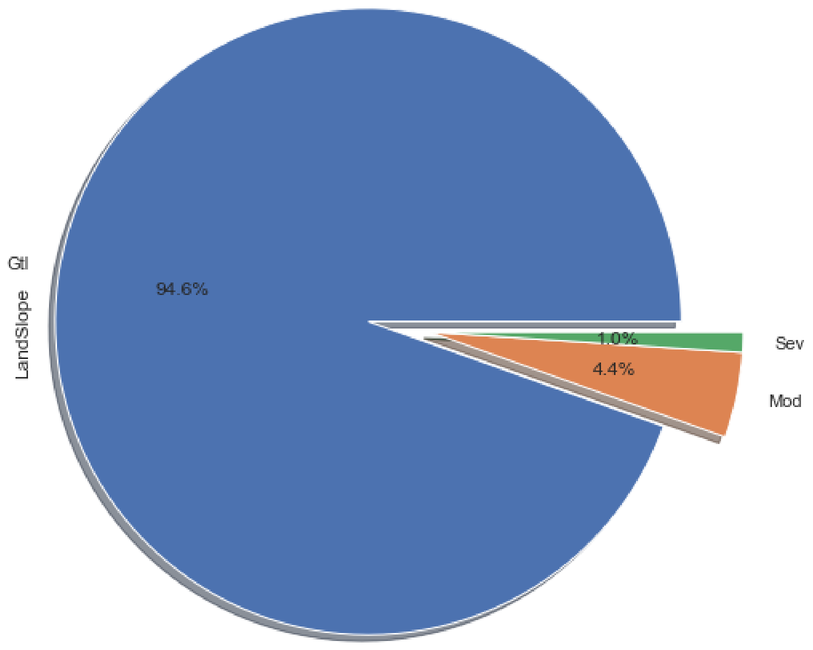
```
Utilities
```



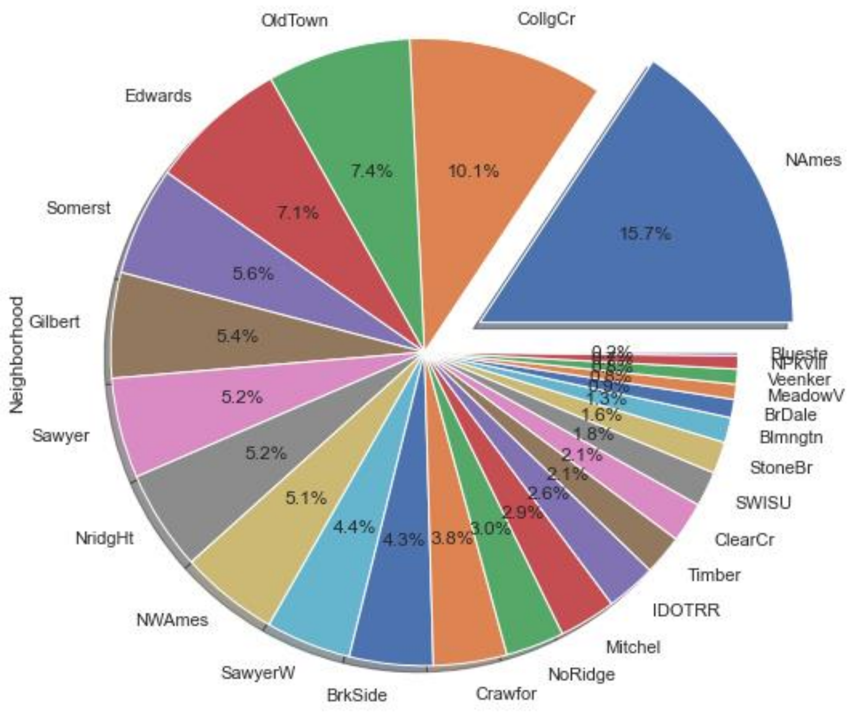
LotConfig



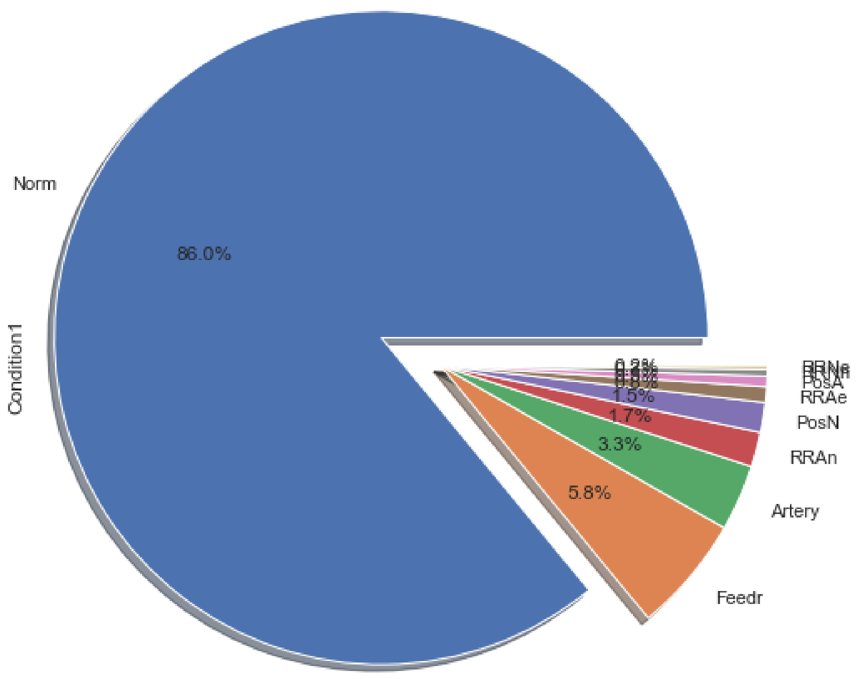
LandSlope



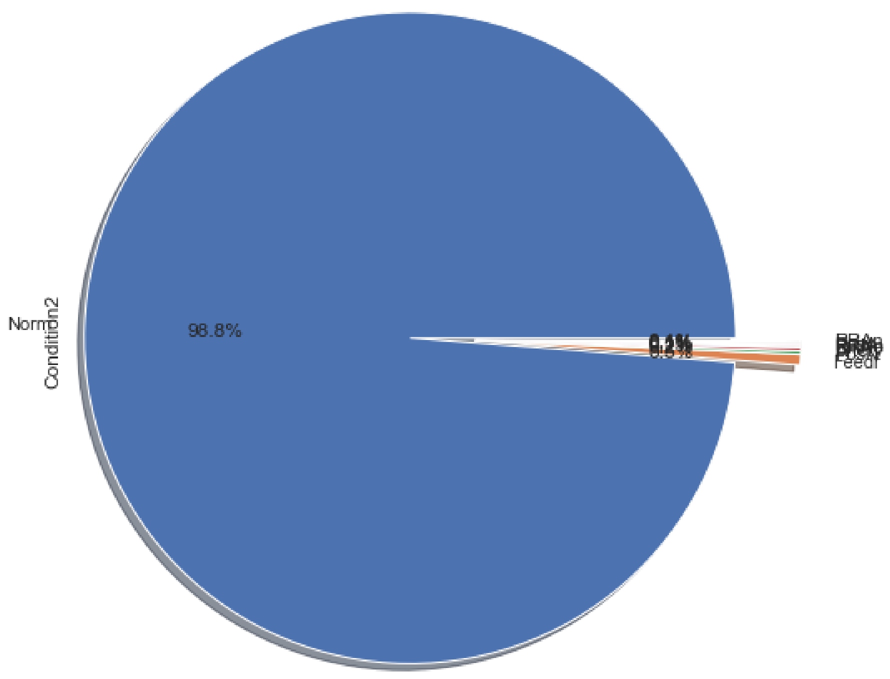
```
Neighborhood
```



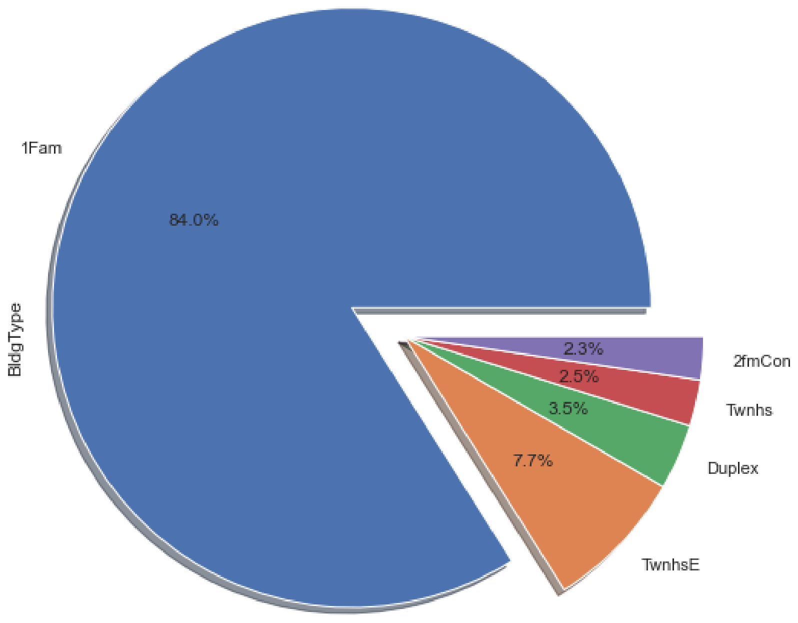
```
Condition1
```



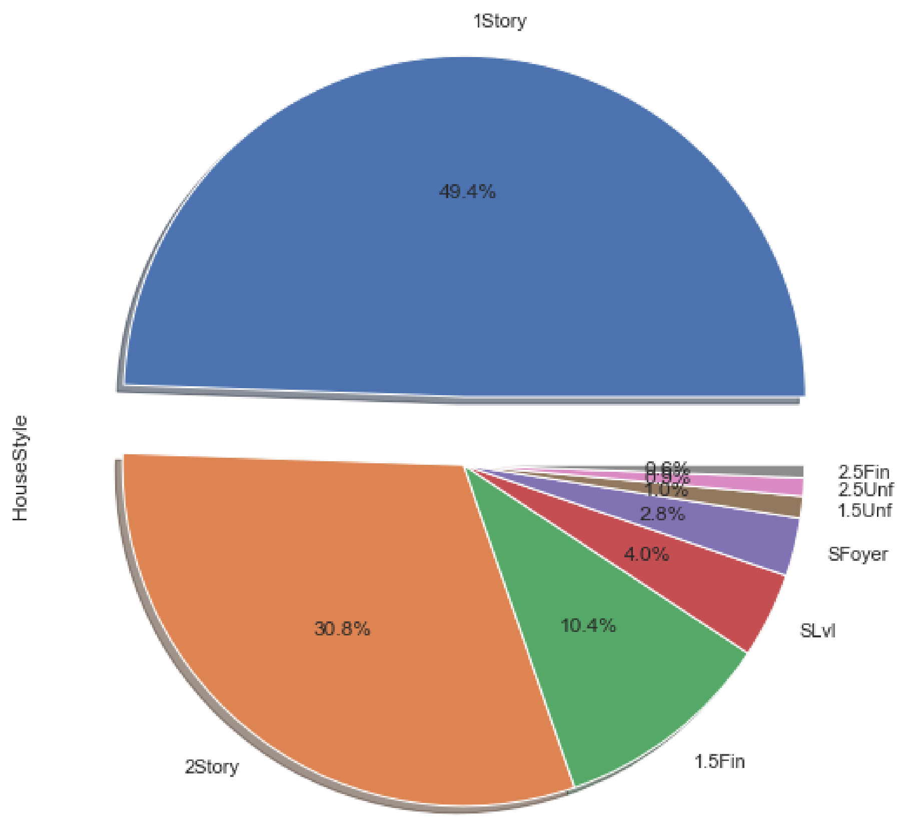
Condition2



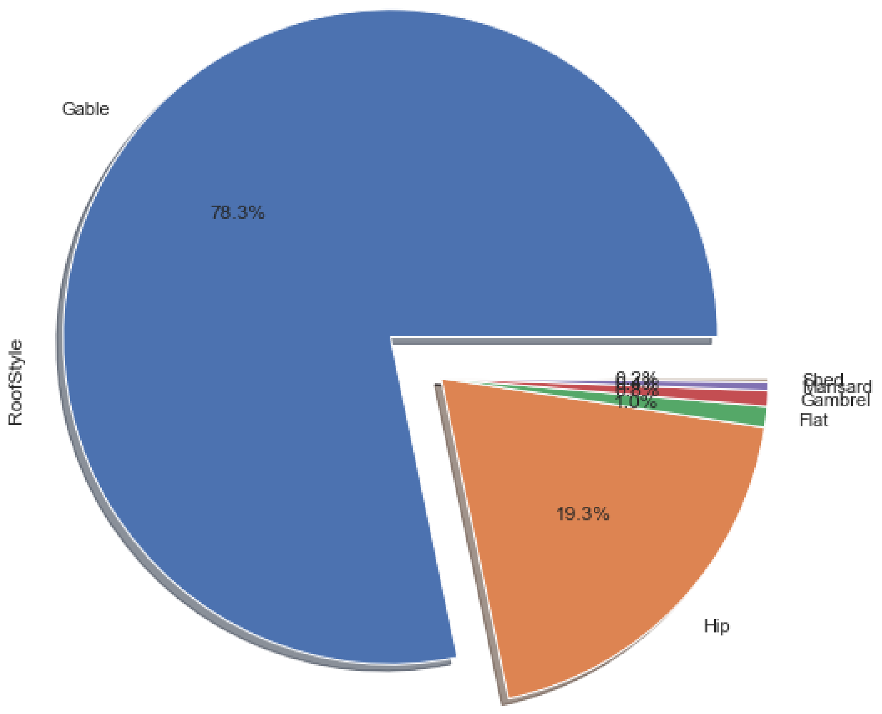
BldgType



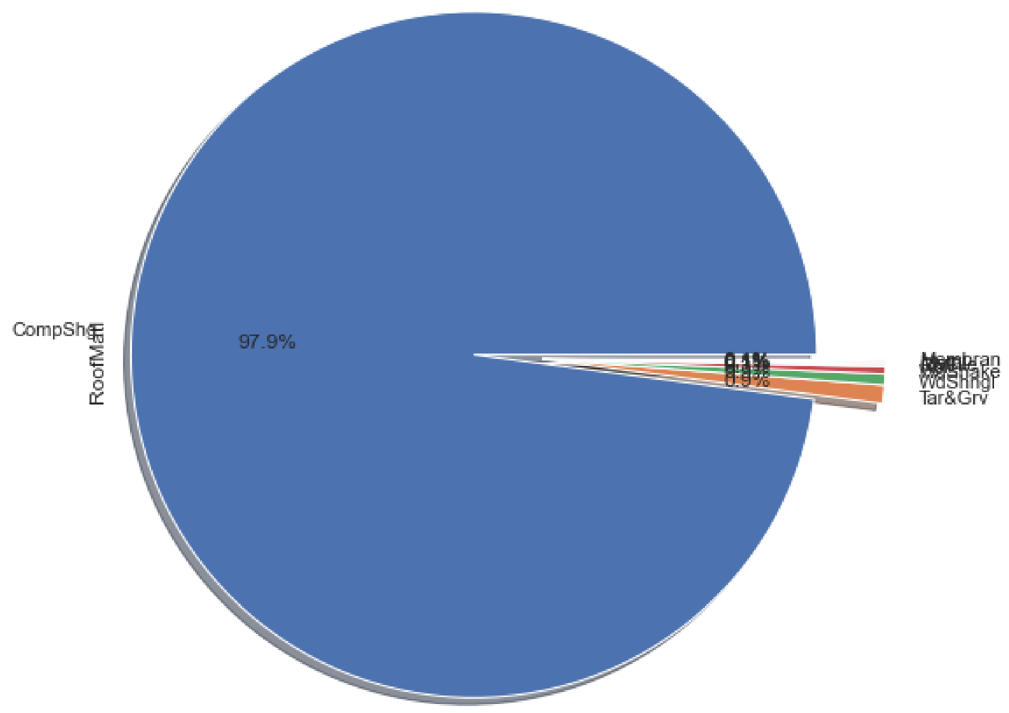
`HouseStyle`



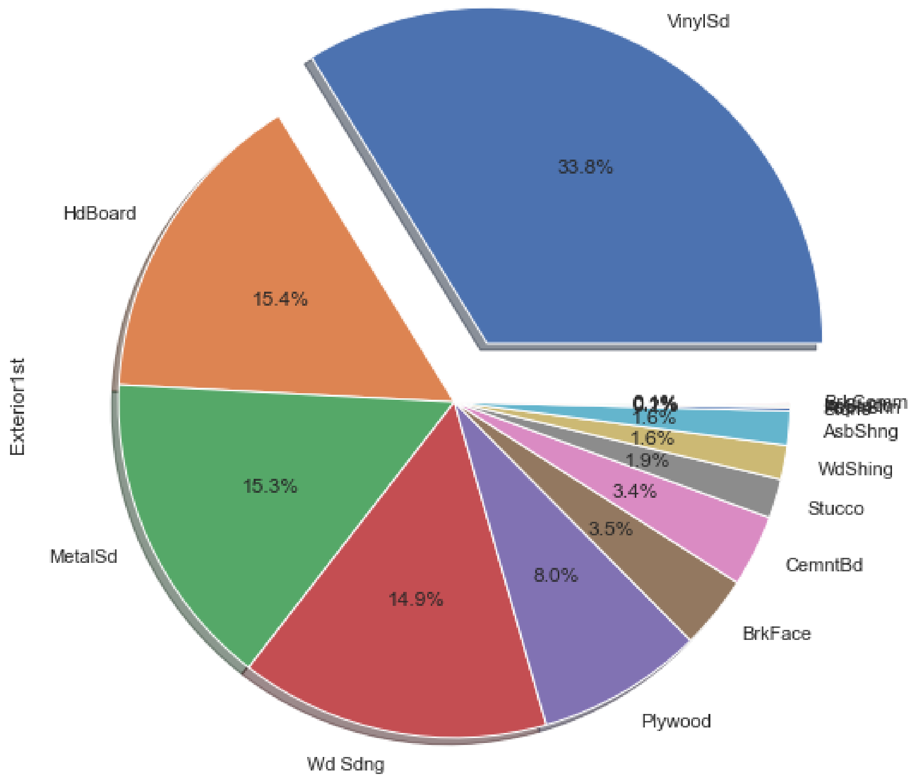
`RoofStyle`



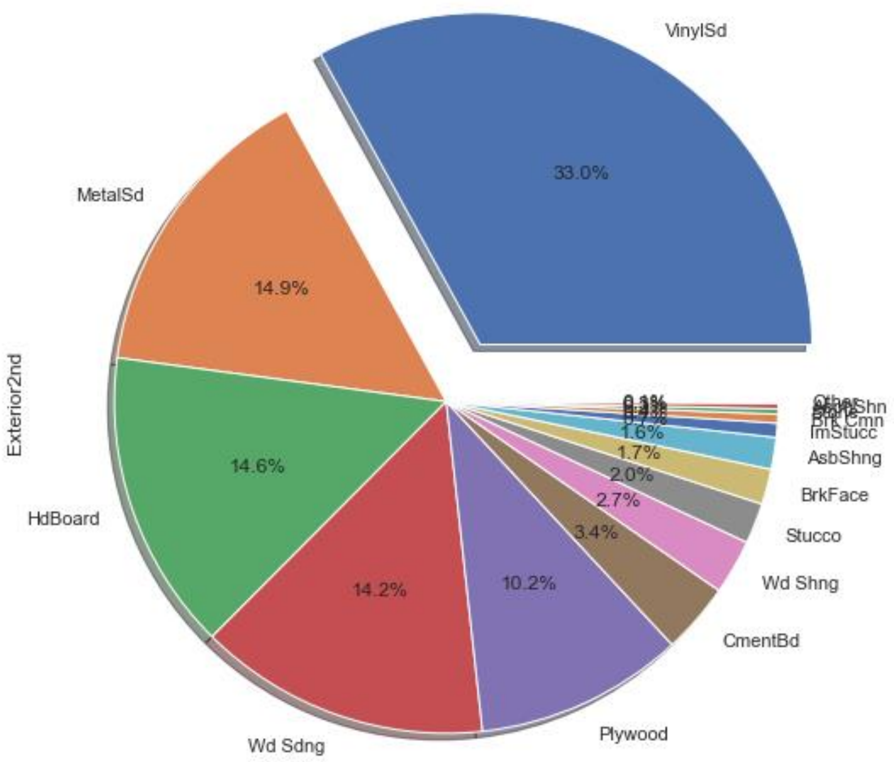
`RoofMatl`



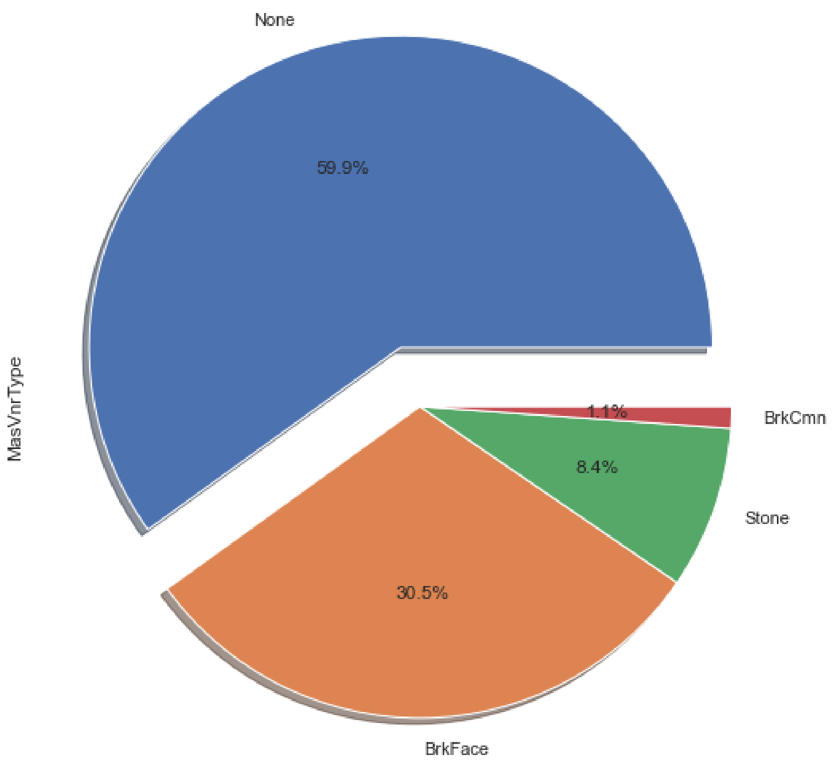
`Exterior1st`



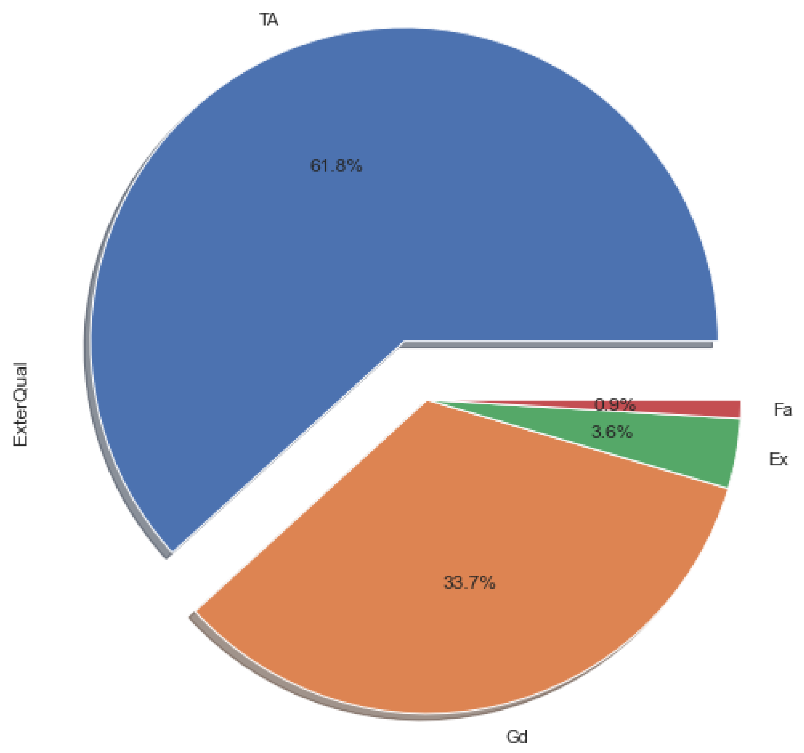
```
Exterior2nd
```



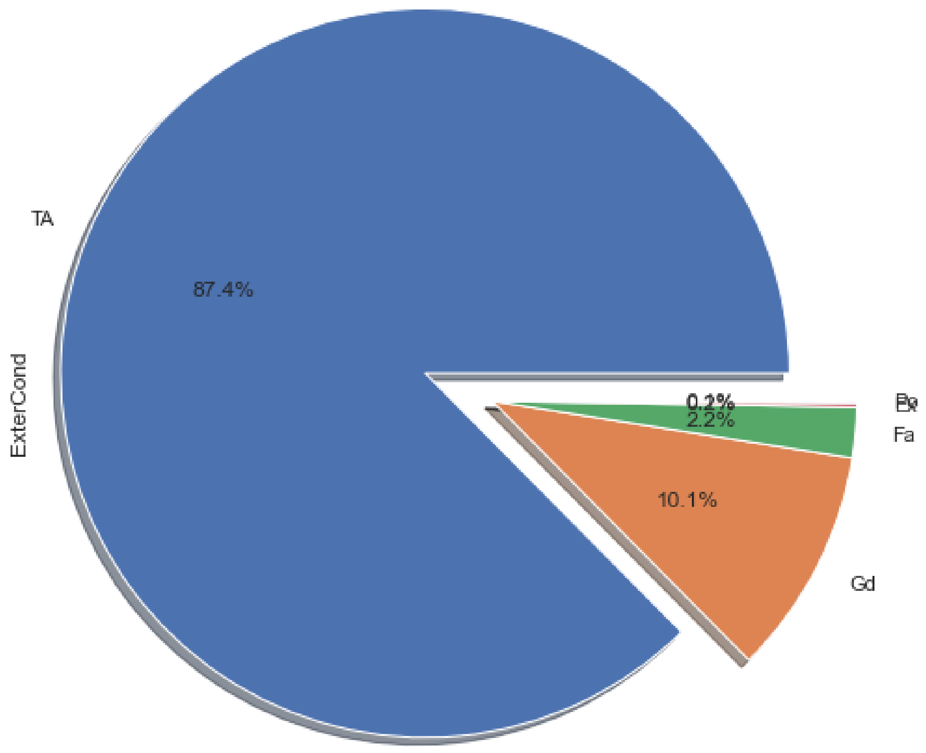
```
MasVnrType
```



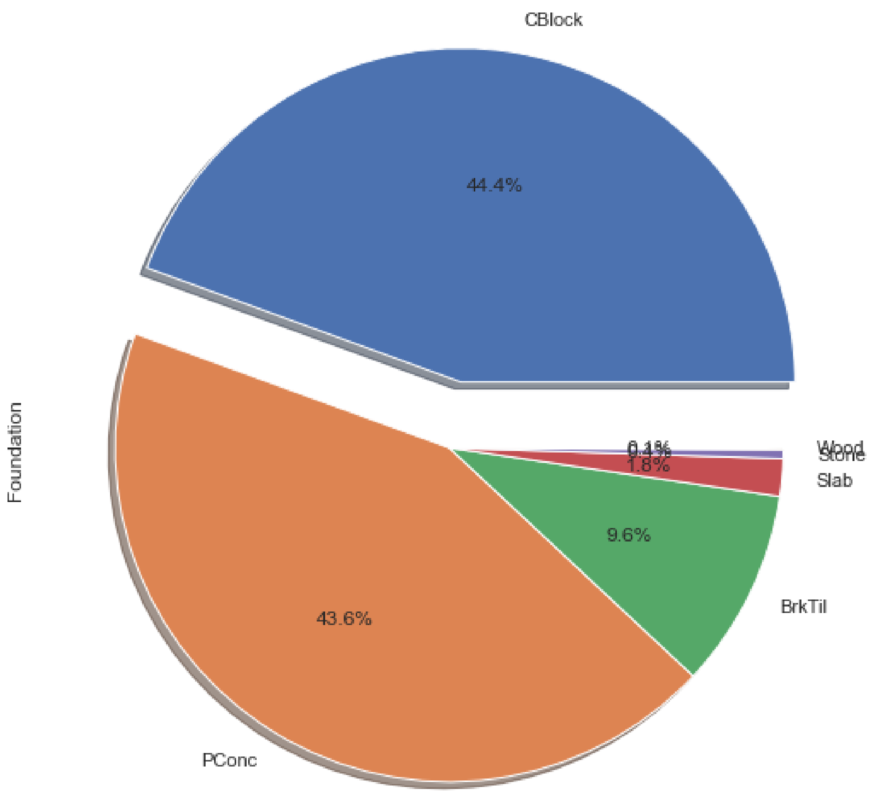
ExterQual



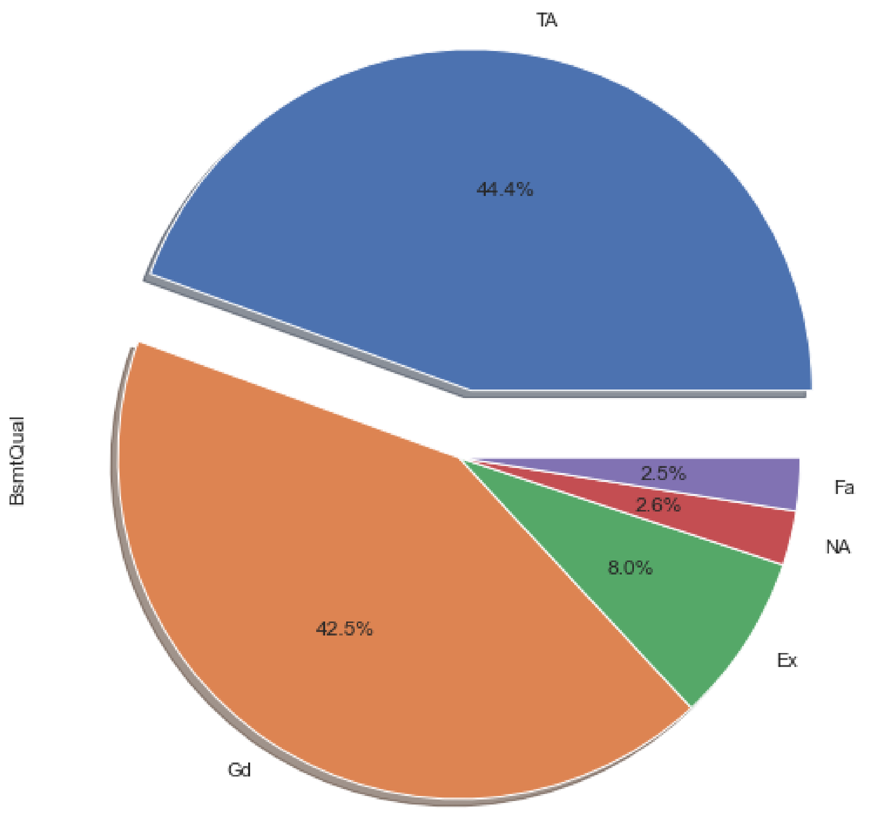
ExterCond



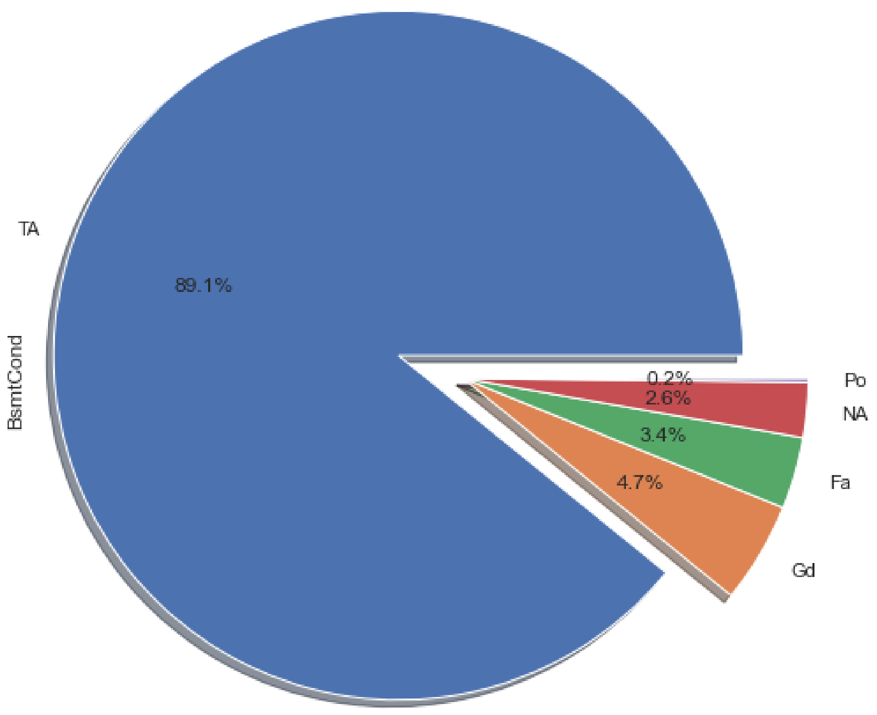
Foundation



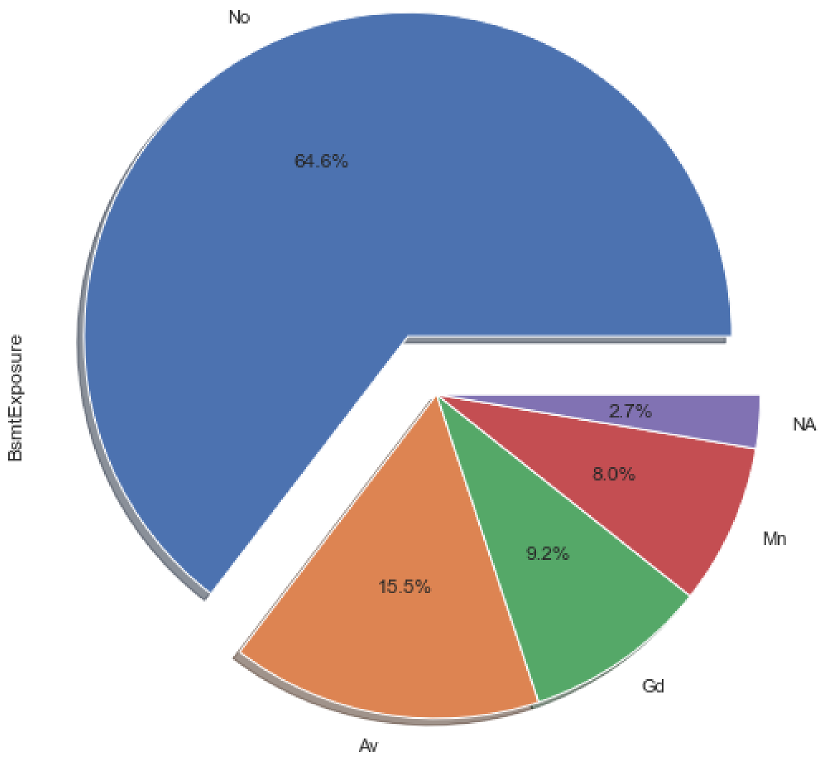
BsmtQual



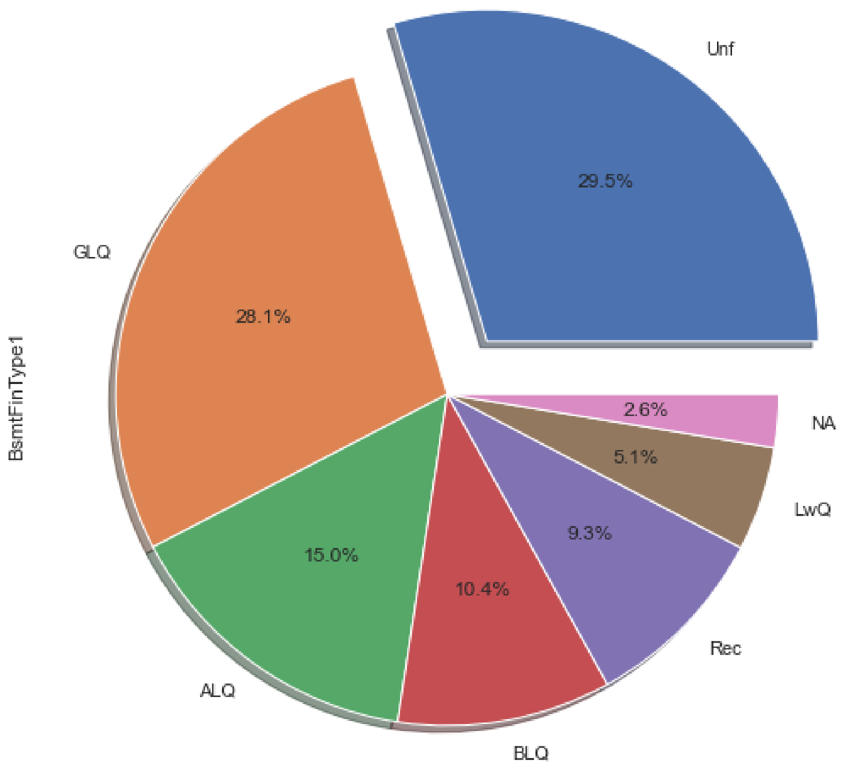
```
BsmtCond
```



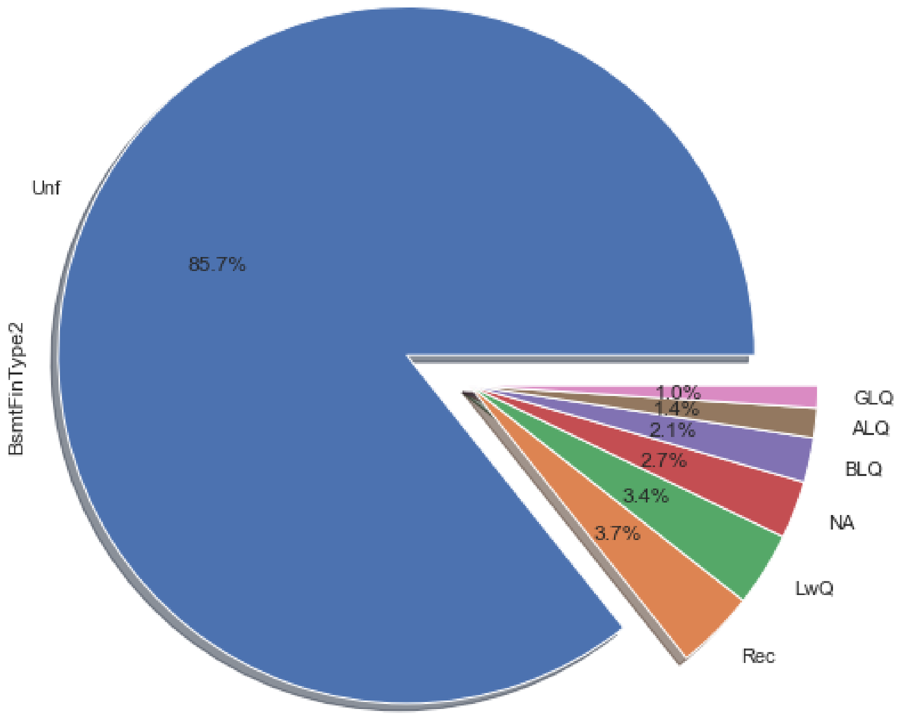
```
BsmtExposure
```



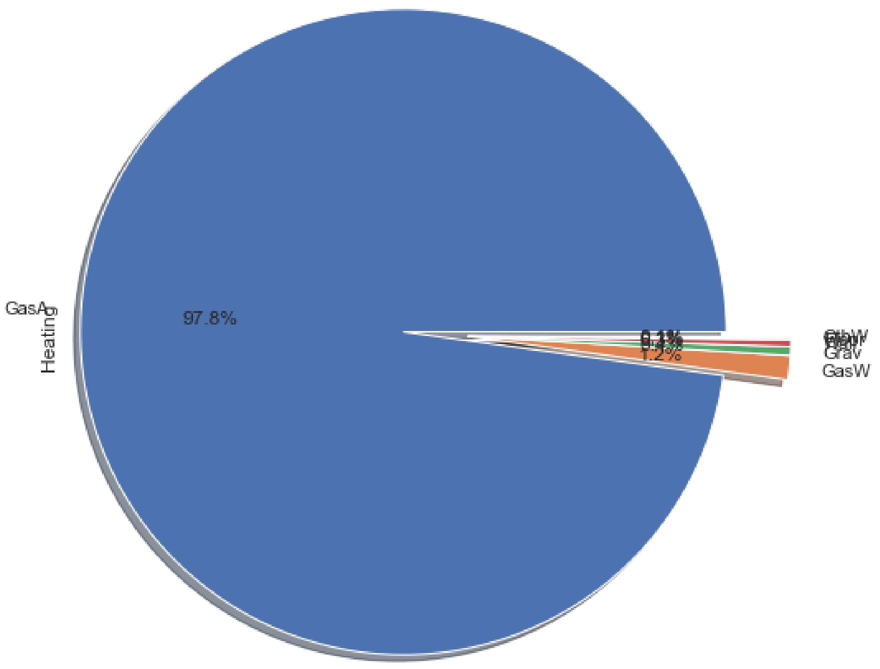
BsmtFinType1



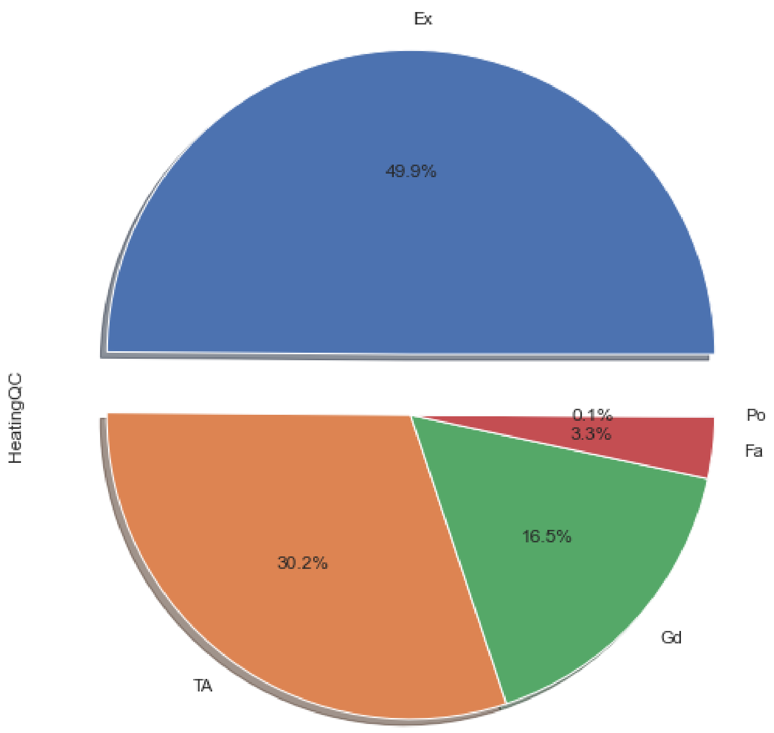
BsmtFinType2



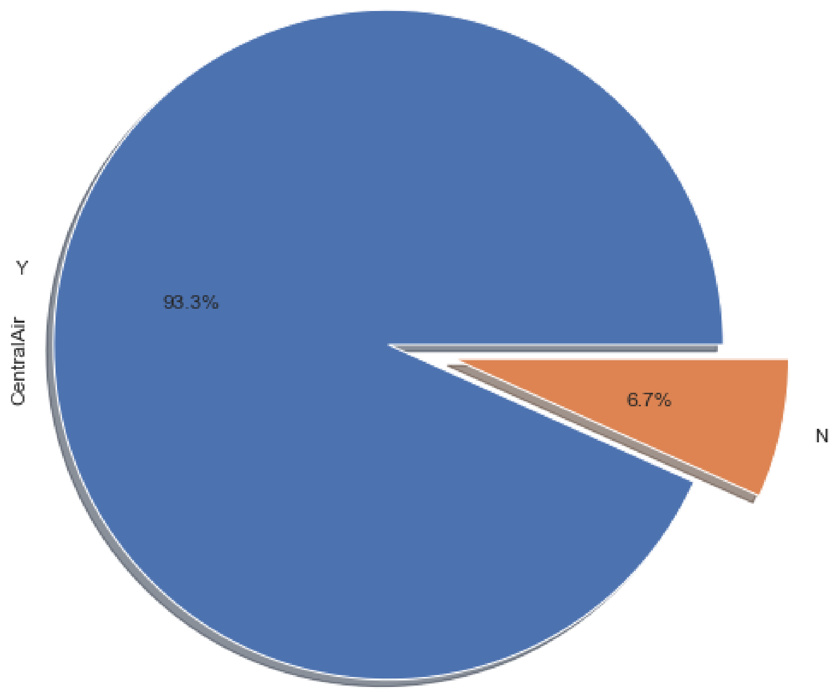
Heating



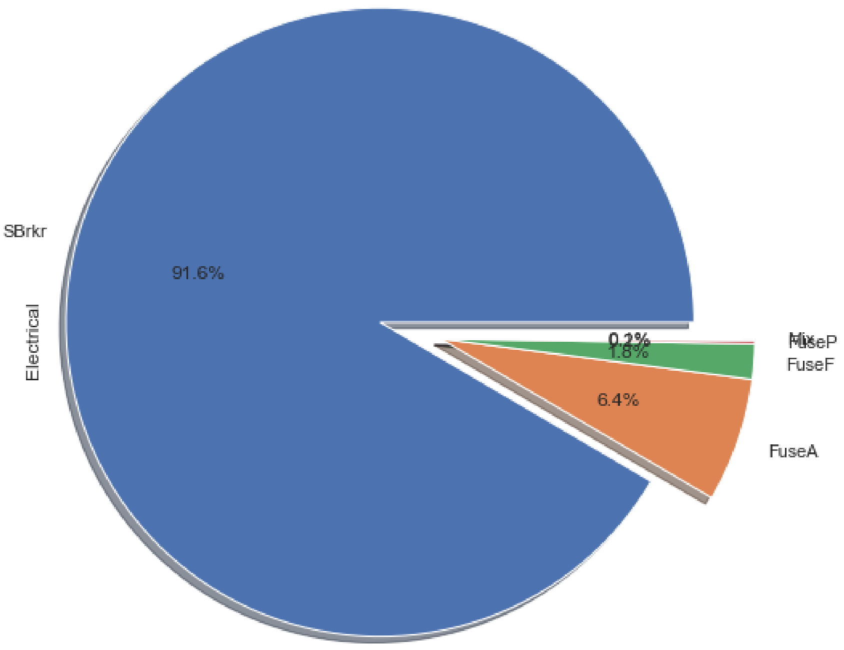
HeatingQC



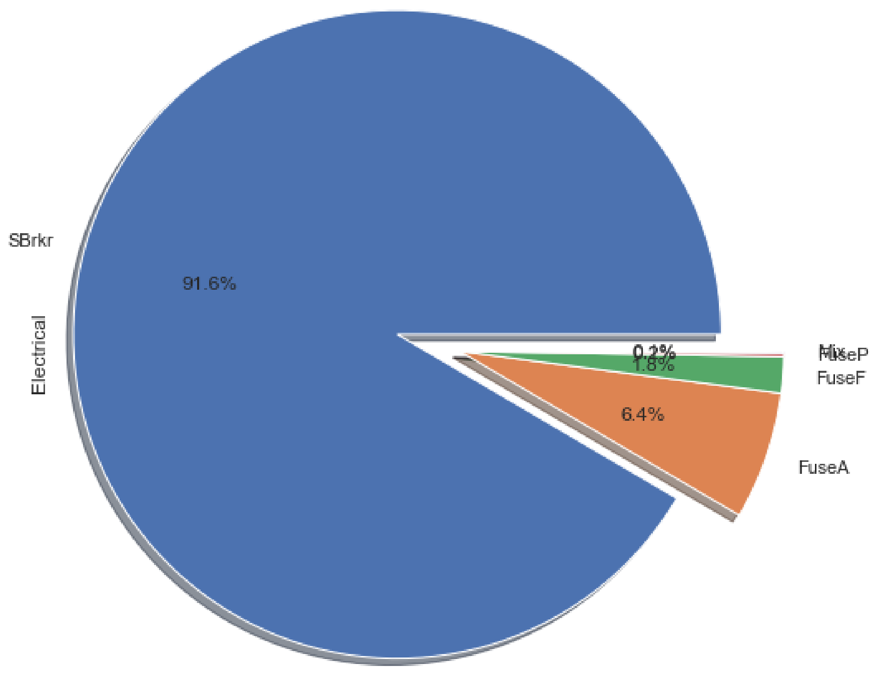
CentralAir



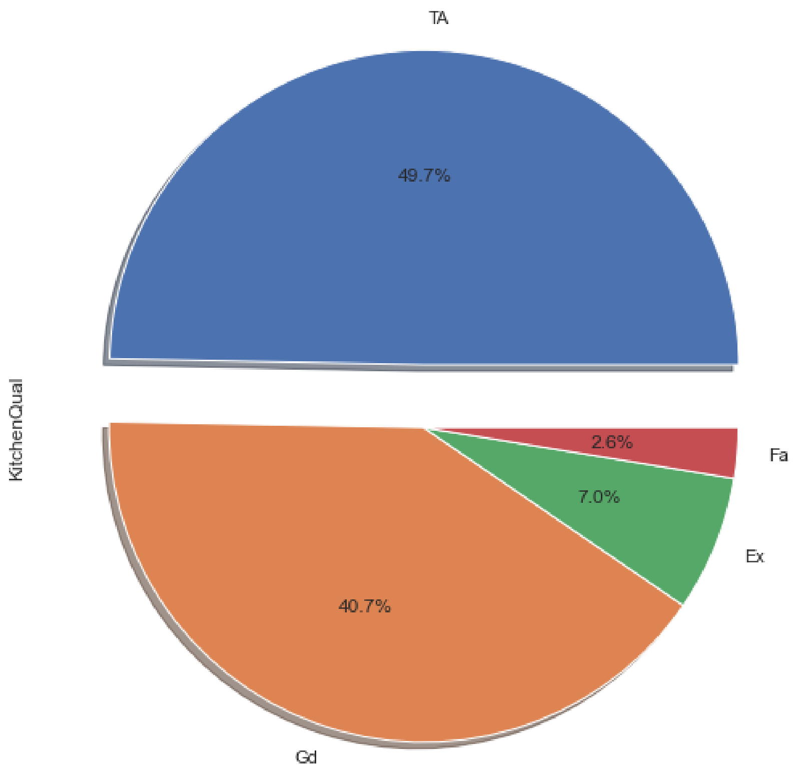
Electrical



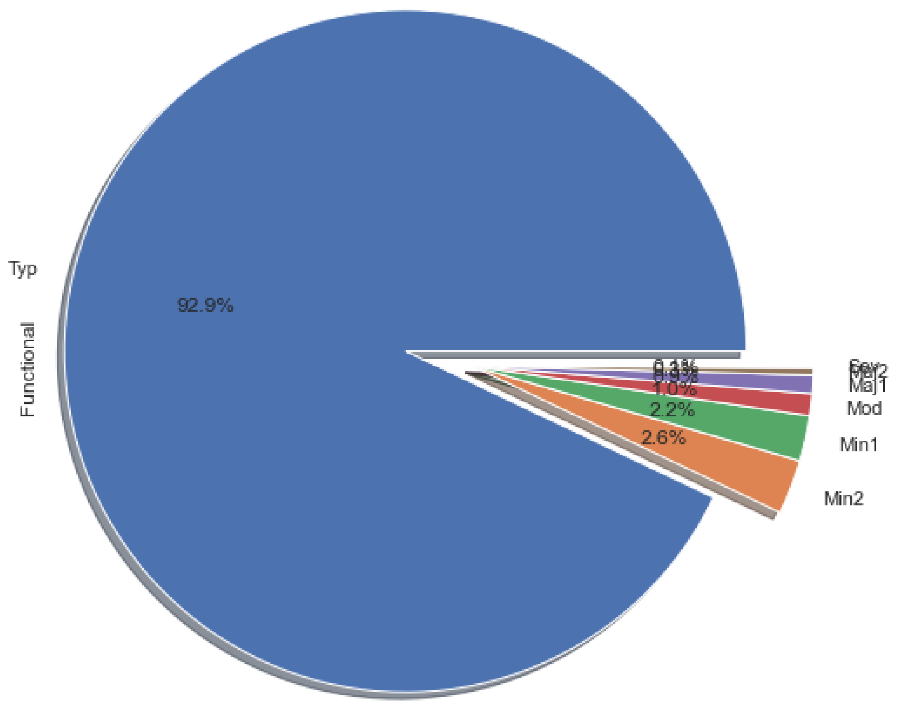
Electrical



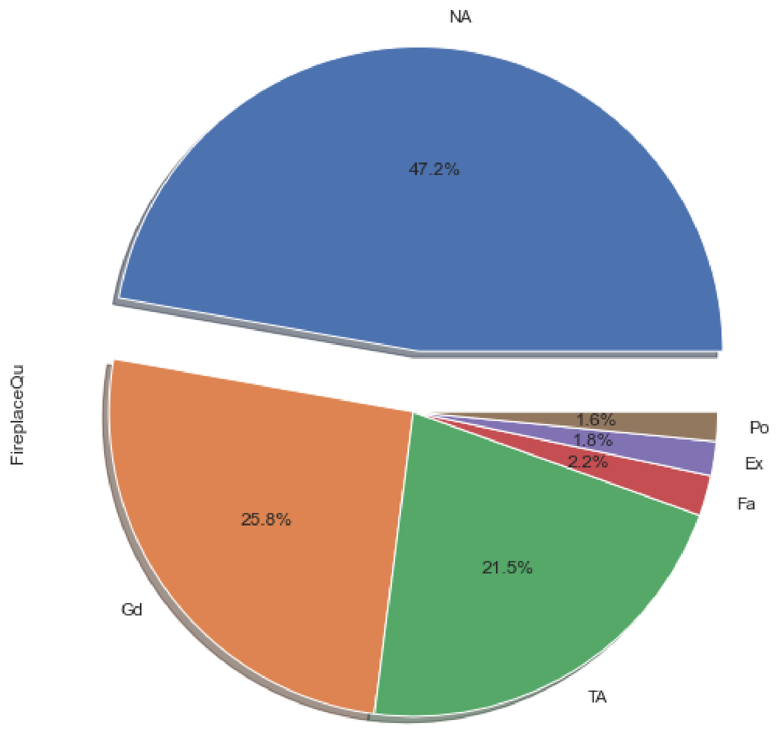
KitchenQual



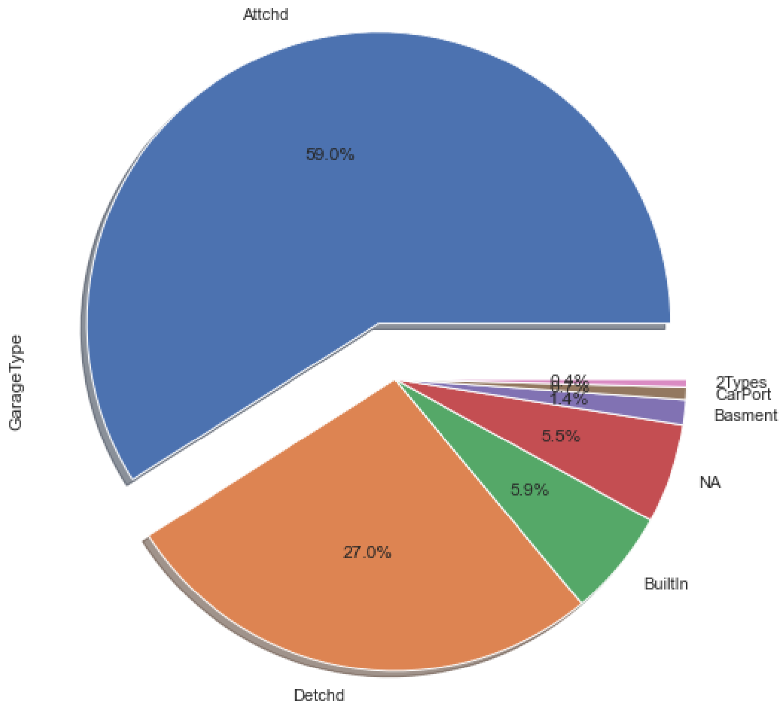
Functional

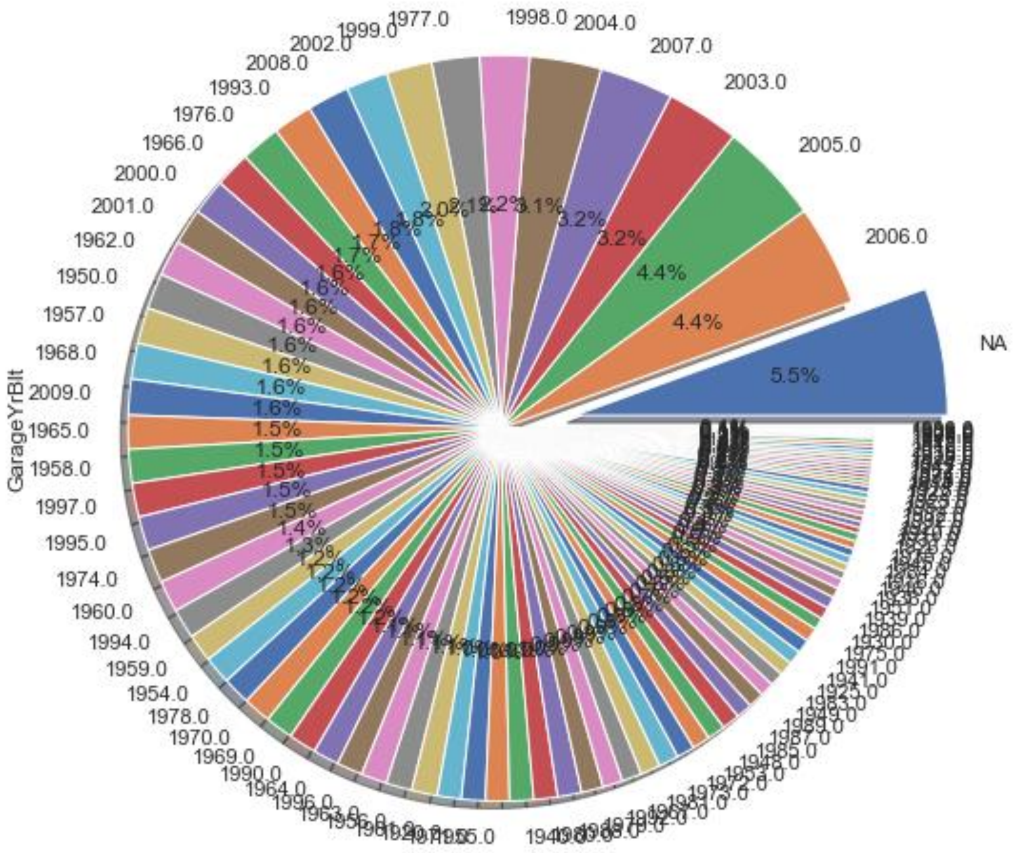


FireplaceQu

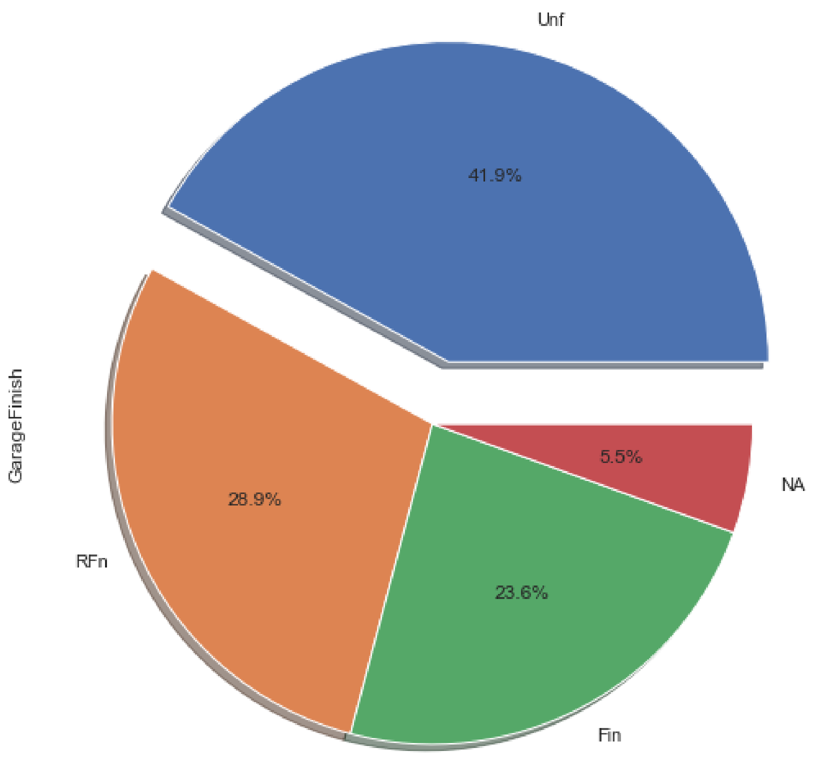


GarageType

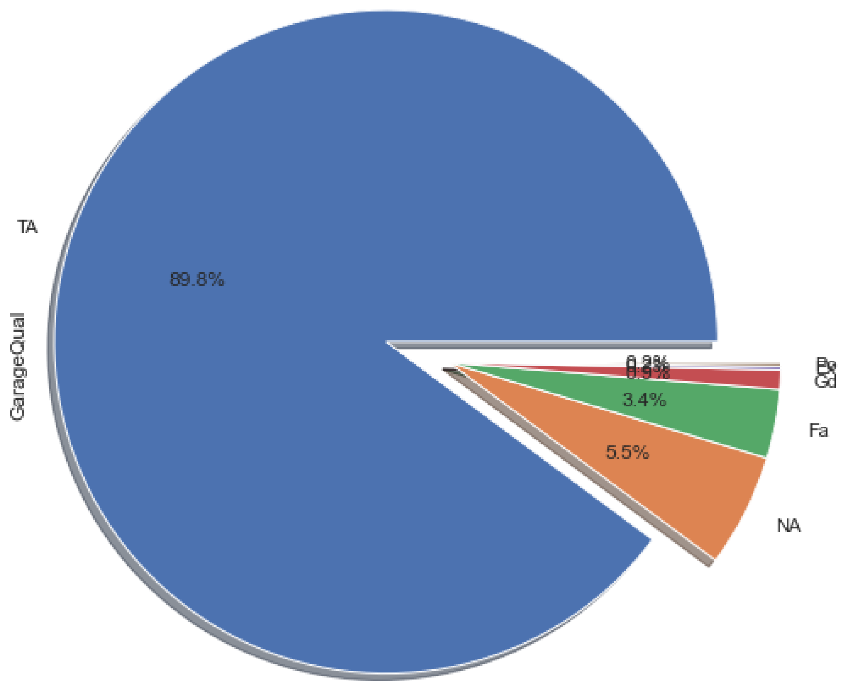

Attchd
59.0%
GarageType
0.4%
0.7%
1.4%
2Types
CarPort
Basment
5.5%
NA
5.9%
BuiltIn
27.0%
Detchd


NA
5.5%
2006.0
4.4%
2005.0
4.4%
2003.0
3.2%
2007.0
3.2%
2004.0
3.1%
1998.0
2.2%
1977.0
2.1%
1999.0
2.0%
2002.0
1.8%
2008.0
1.8%
1993.0
1.7%
1976.0
1.7%
1966.0
1.6%
2000.0
1.6%
2001.0
1.6%
1962.0
1.6%
1950.0
1.6%
1957.0
1.6%
1968.0
1.6%
2009.0
1.6%
1965.0
1.5%
1958.0
1.5%
1997.0
1.5%
1995.0
1.5%
1974.0
1.5%
1960.0
1.4%
1994.0
1.3%
1959.0
1.2%
1954.0
1.2%
1978.0
1.2%
1970.0
1.2%
1969.0
1990.0
1964.0
GarageYrBlt

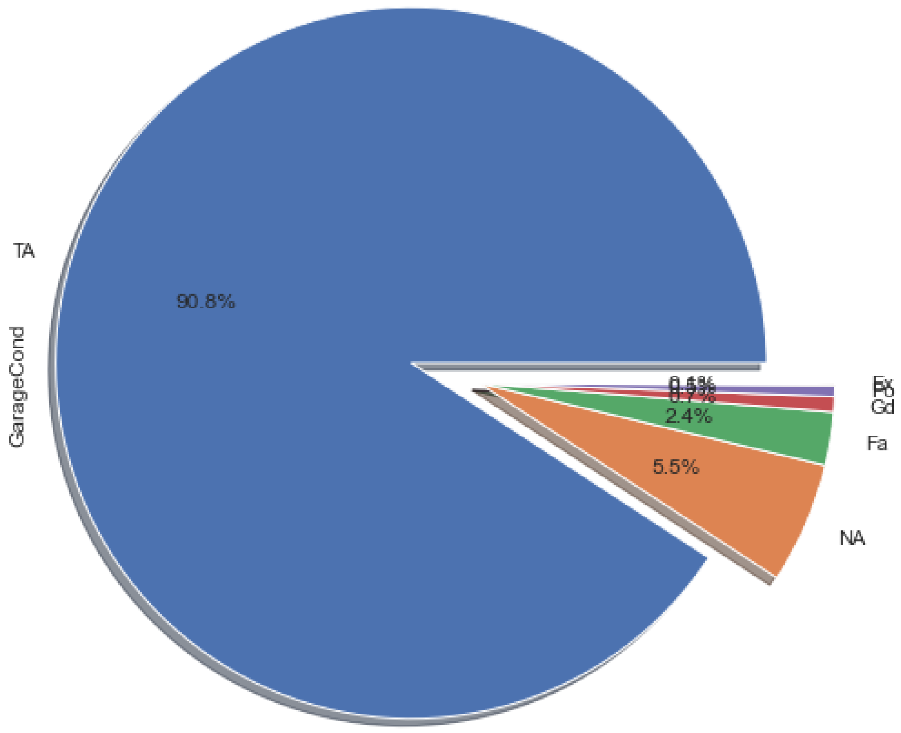
GarageFinish



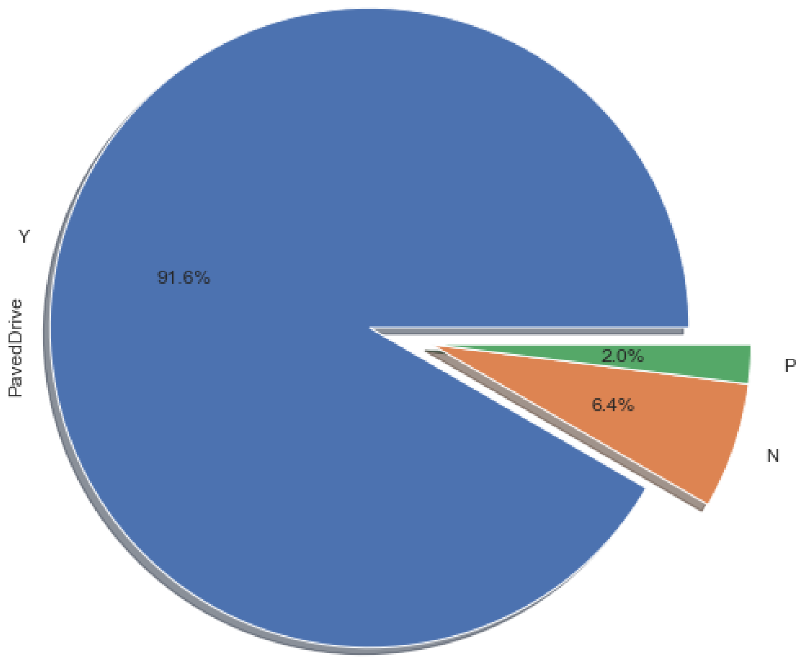
GarageQual



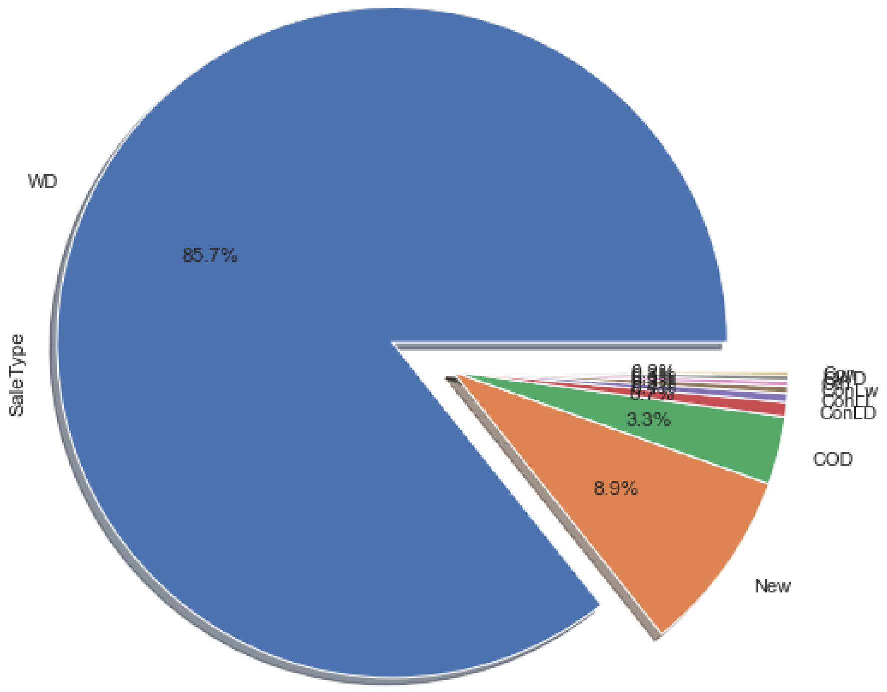
GarageCond



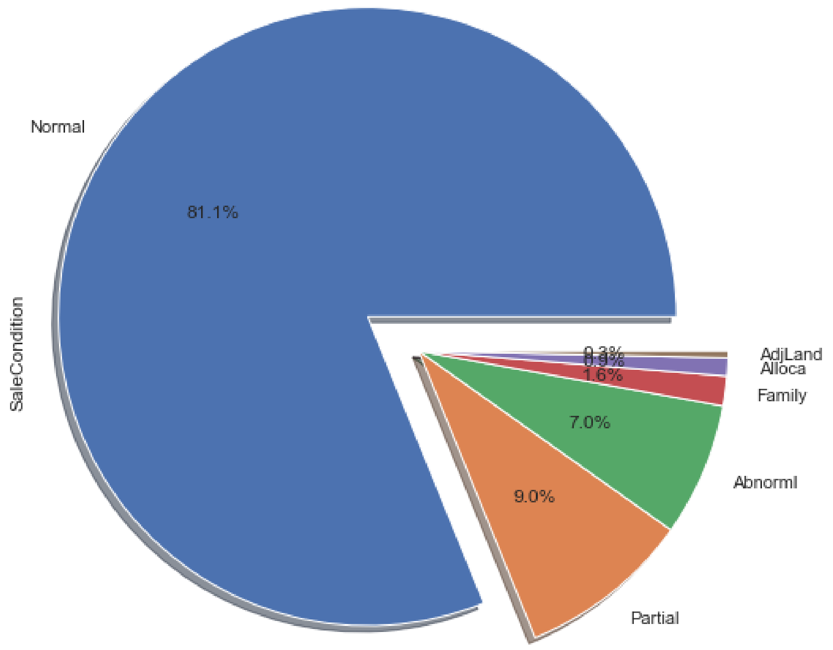
PavedDrive

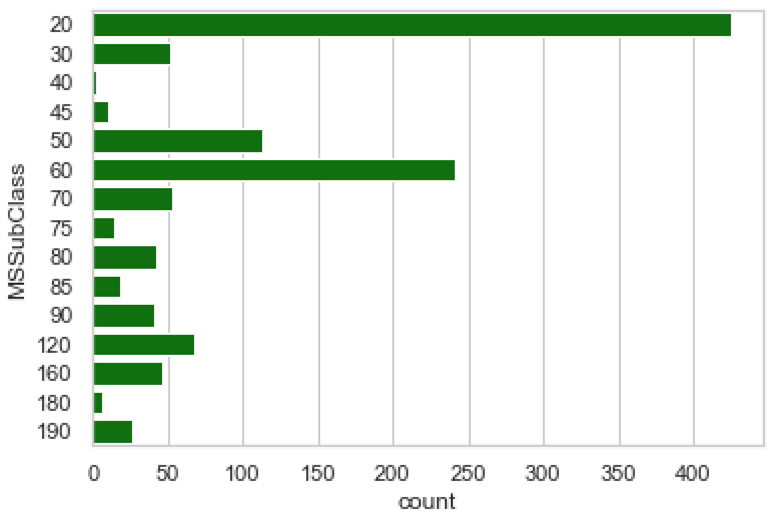


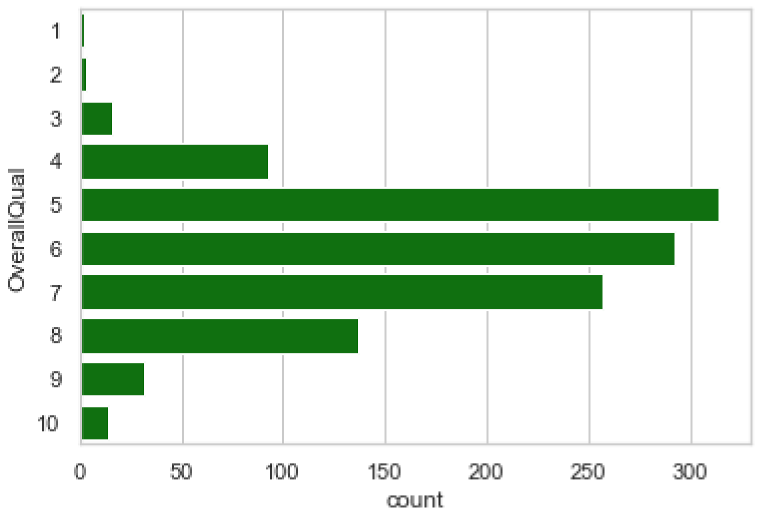
```
SaleType
```

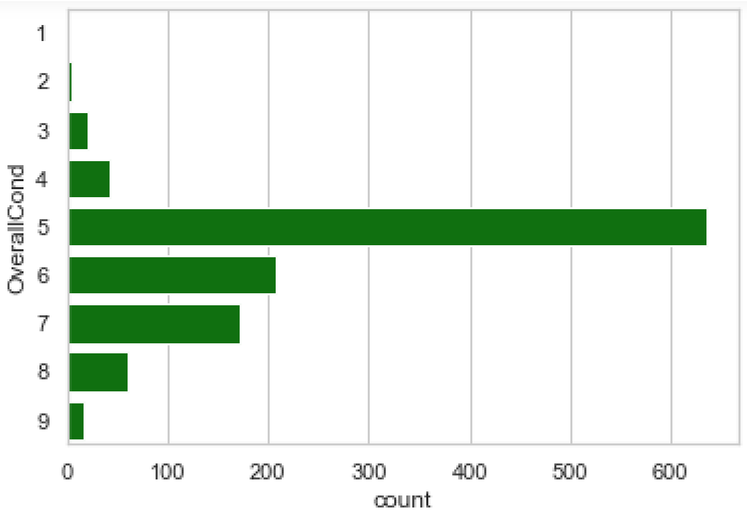


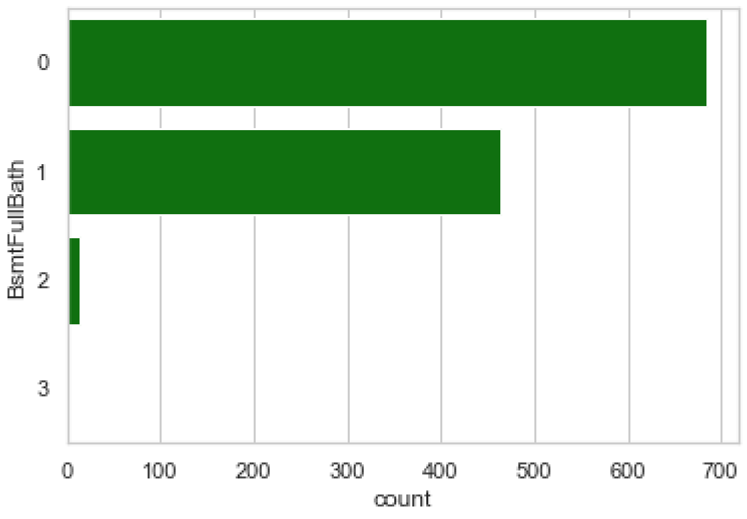
```
SaleCondition
```

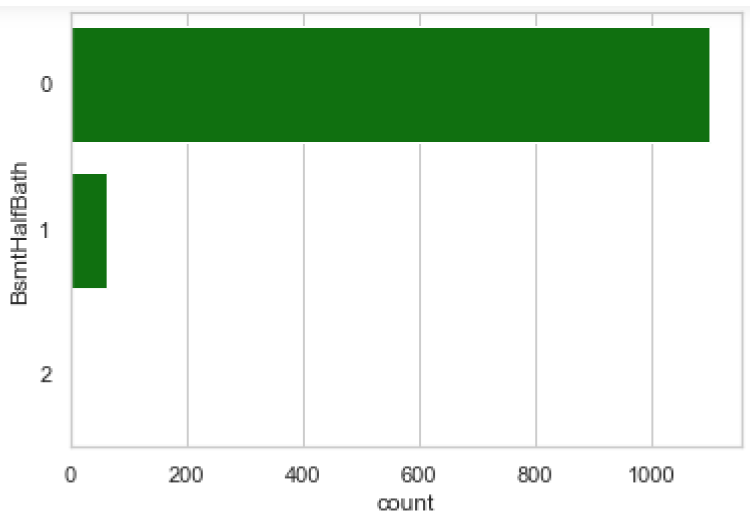


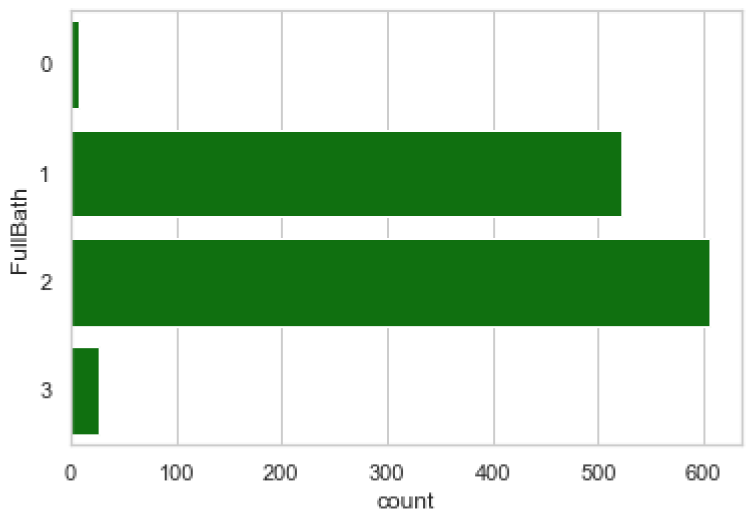

MSSubClass
20
30
40
45
50
60
70
75
80
85
90
120
160
180
190
0
50
100
150
200
250
300
350
400
count

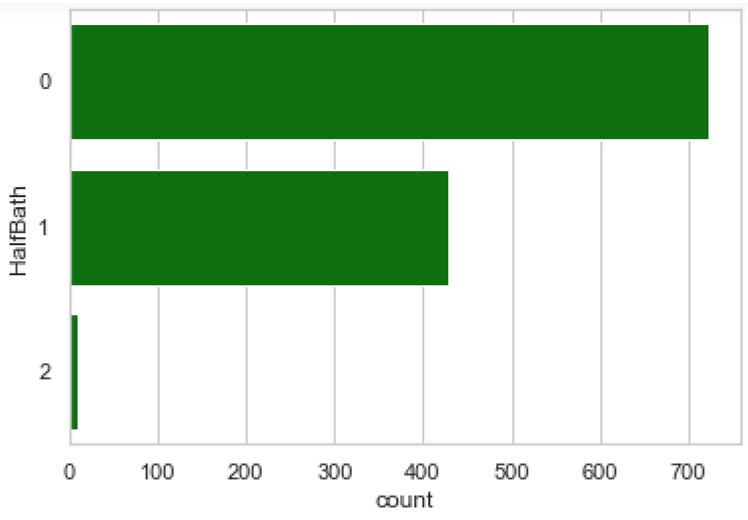

OverallQual
1
2
3
4
5
6
7
8
9
10
0
50
100
150
200
250
300
count

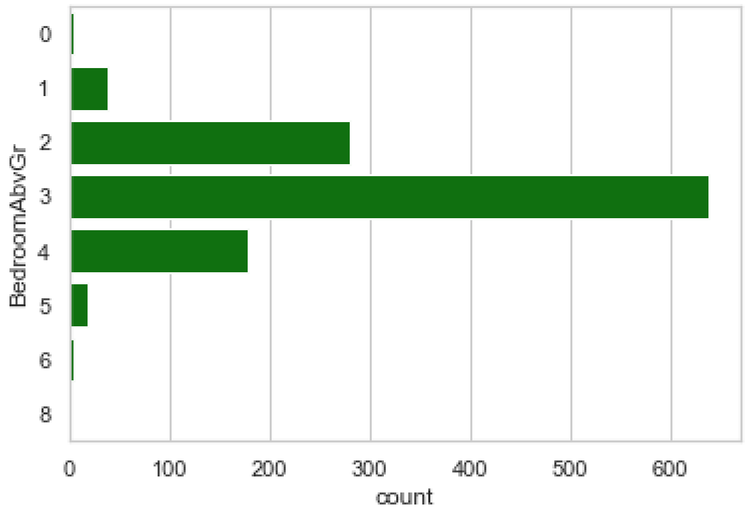

OverallCond
1
2
3
4
5
6
7
8
9
0
100
200
300
400
500
600
count

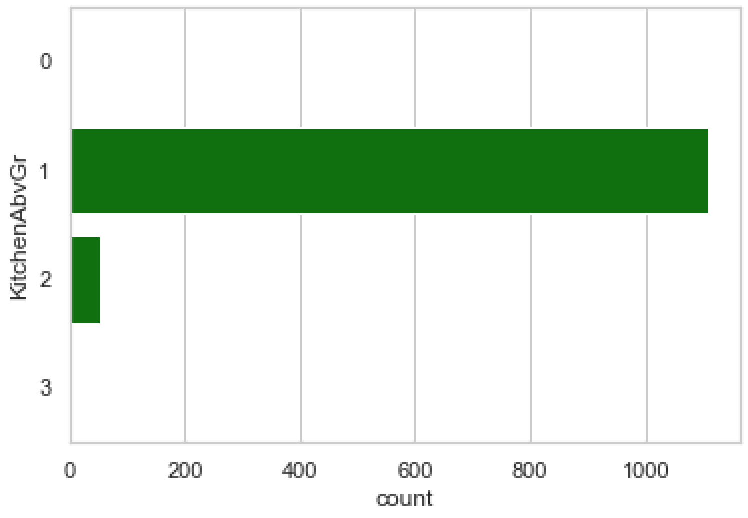

BsmtFullBath
0
1
2
3
0
100
200
300
400
500
600
700
count

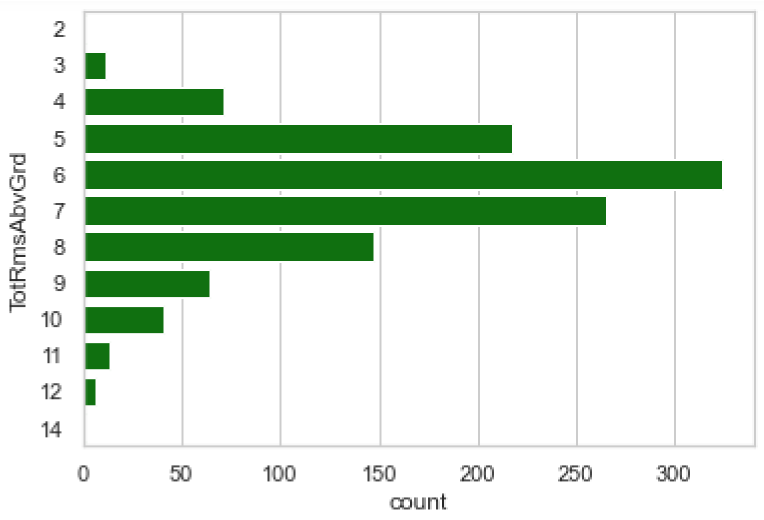

0
1
2
BsmtHalfBath
0
200
400
600
800
1000
count

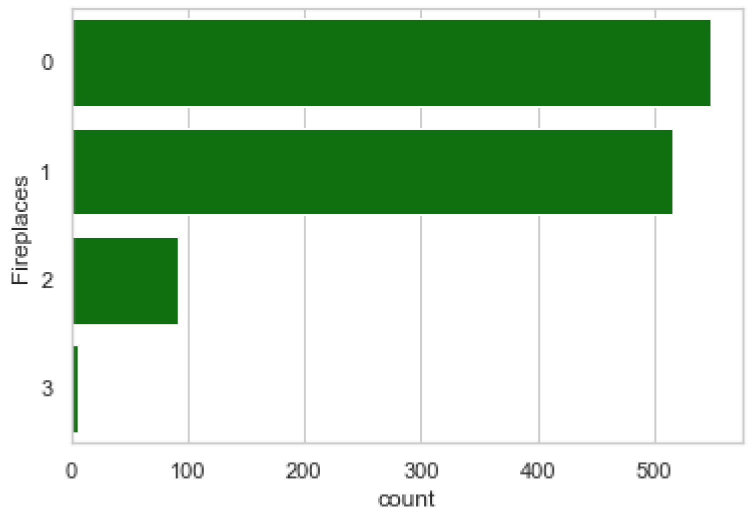

0
1
2
3
FullBath
0
100
200
300
400
500
600
count

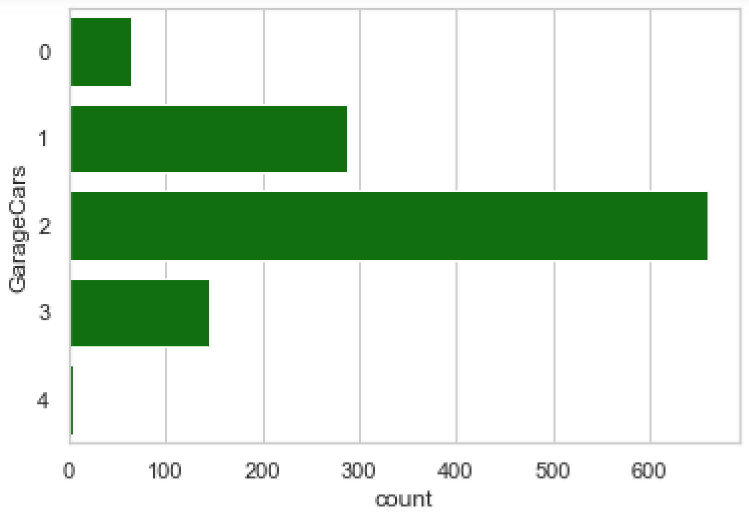

HalfBath
0
1
2
0
100
200
300
400
500
600
700
count

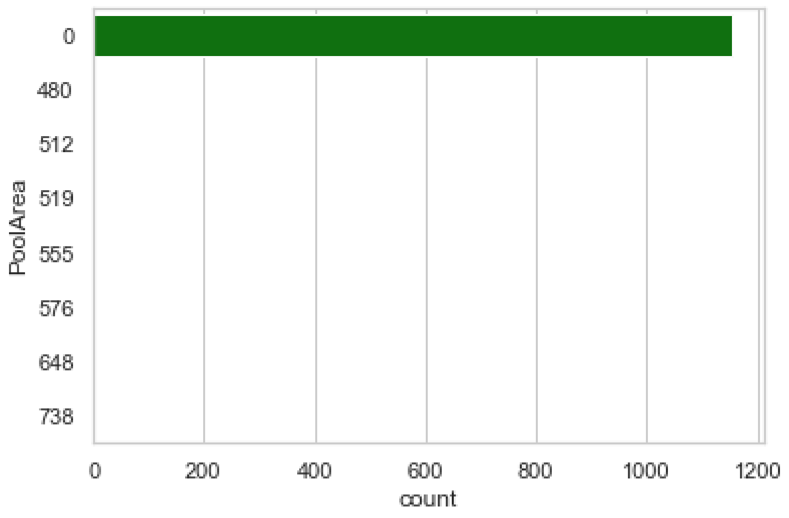

BedroomAbvGr
0
1
2
3
4
5
6
8
0
100
200
300
400
500
600
count

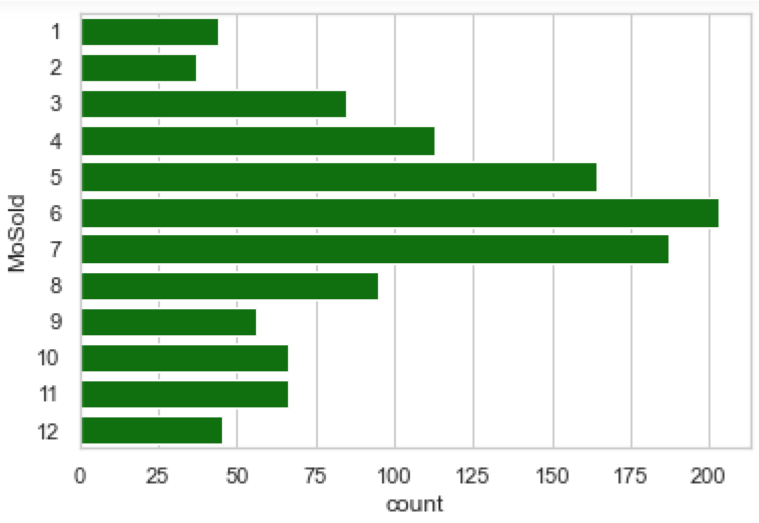

KitchenAbvGr
0
1
2
3
0
200
400
600
800
1000
count

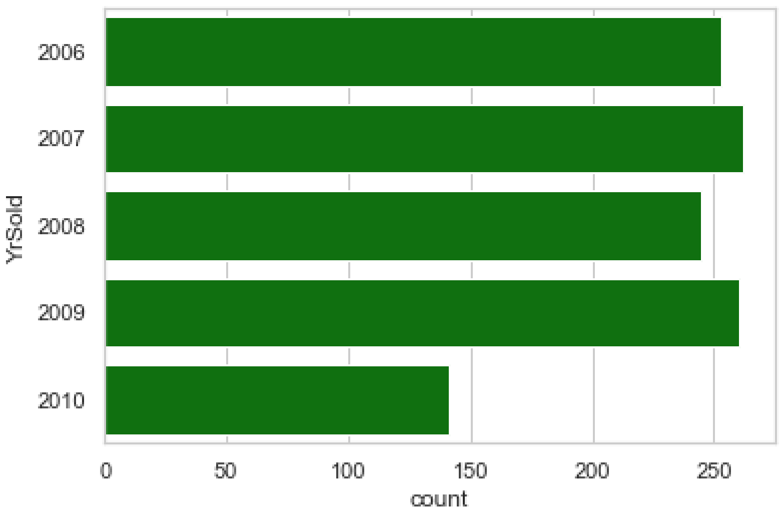

TotRmsAbvGrd
2
3
4
5
6
7
8
9
10
11
12
14
0
50
100
150
200
250
300
count


Fireplaces
0
1
2
3
0
100
200
300
400
500
count

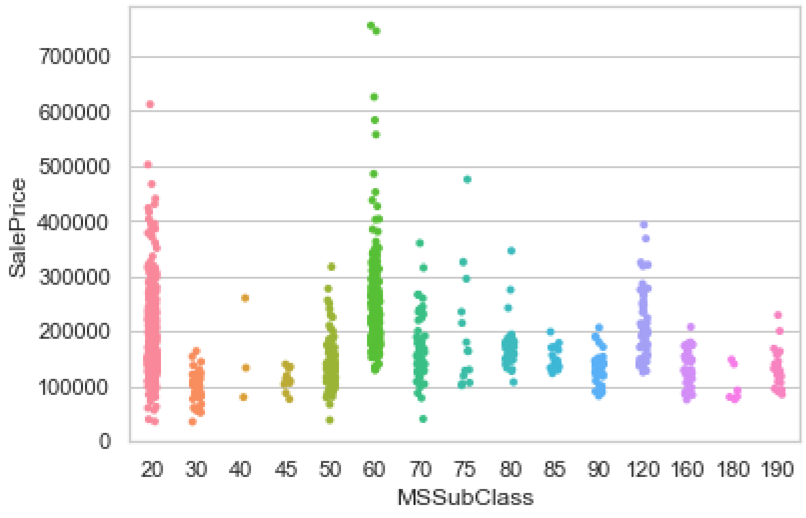

GarageCars
0
1
2
3
4
0
100
200
300
400
500
600
count

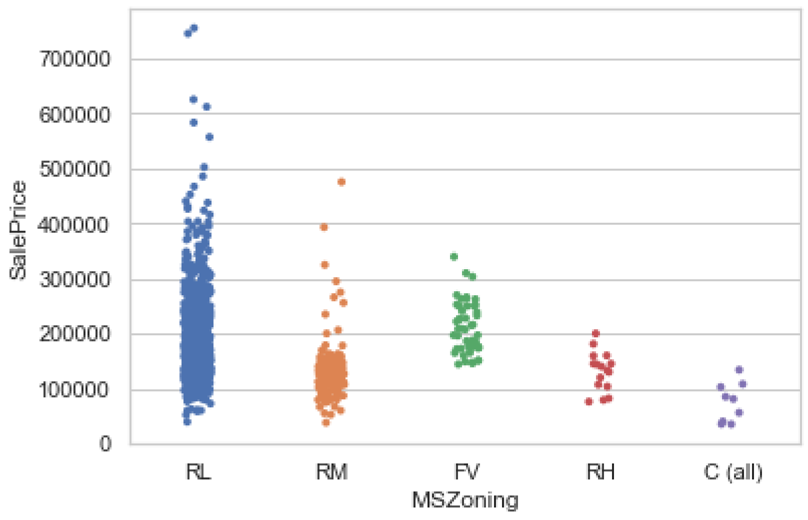

PoolArea
0
480
512
519
555
576
648
738
0
200
400
600
800
1000
1200
count



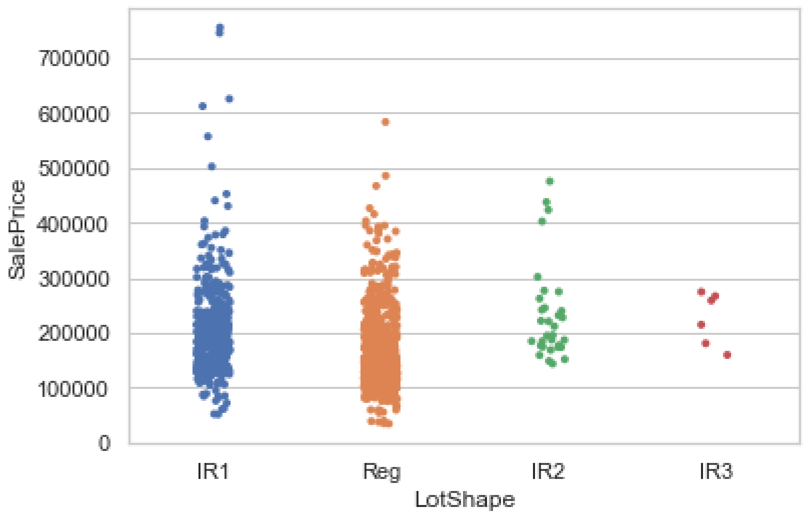


Bivariate Analysis

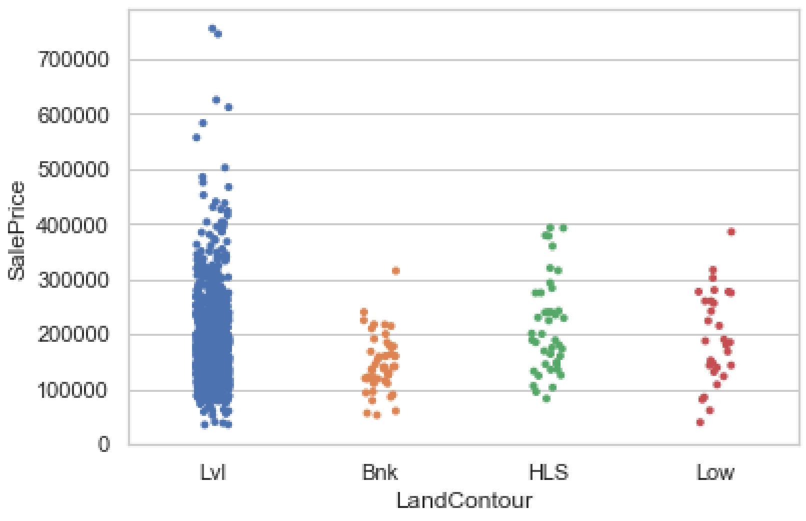




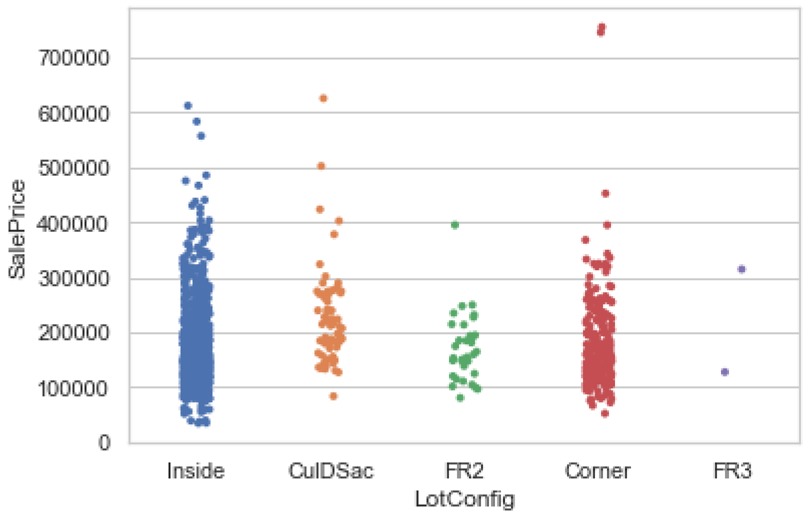
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
RL
RM
FV
RH
C (all)
MSZoning

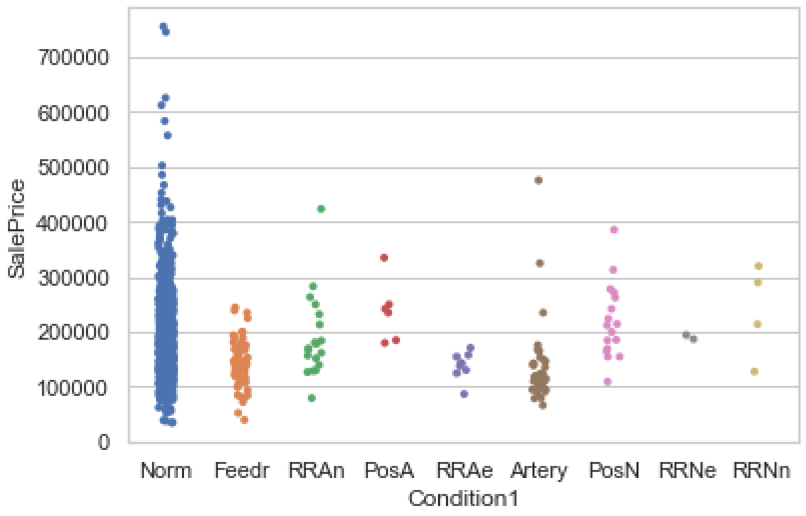


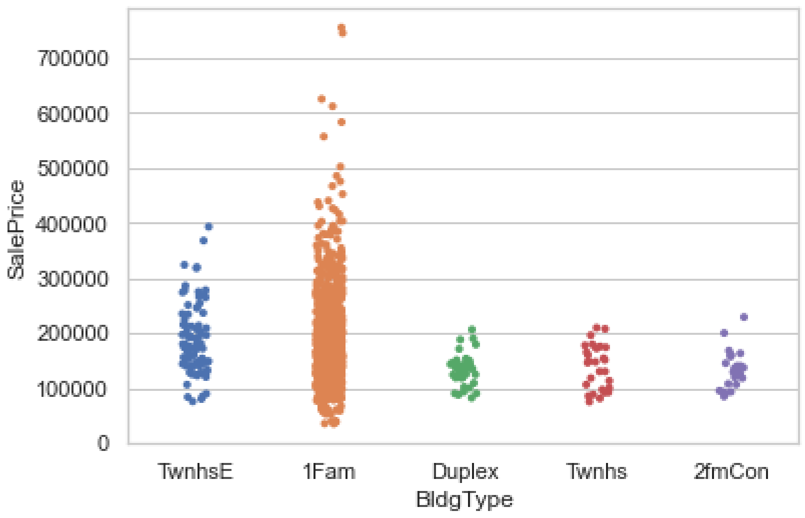
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
IR1
Reg
IR2
IR3
LotShape

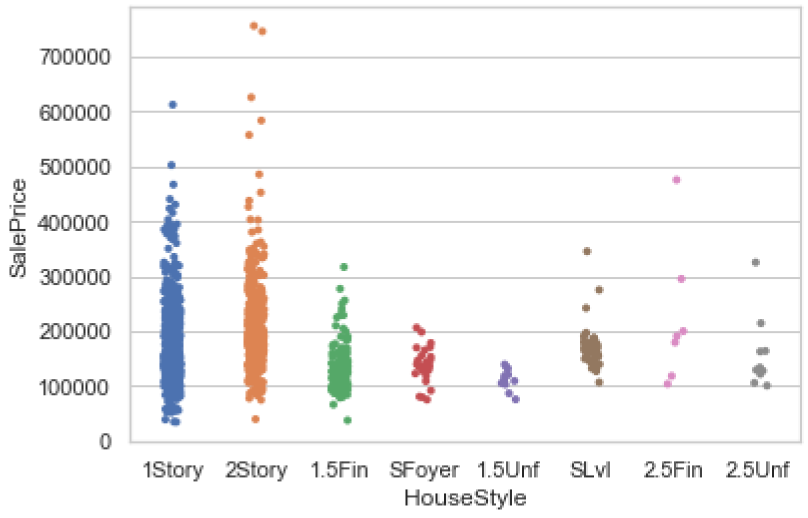


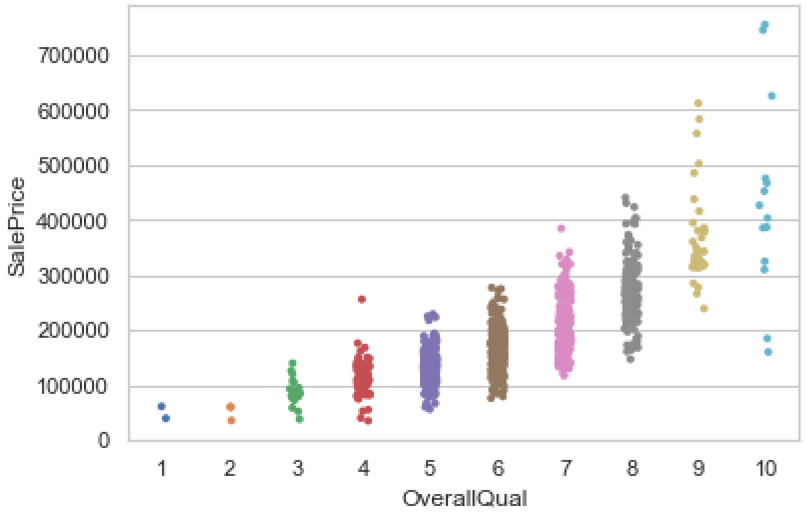
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Lvl
Bnk
HLS
Low
LandContour

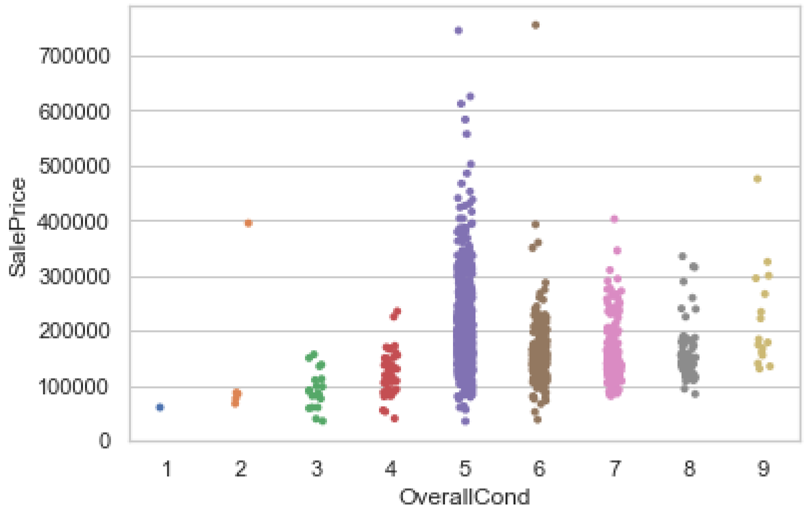

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Inside
CulDSac
FR2
Corner
FR3
LotConfig

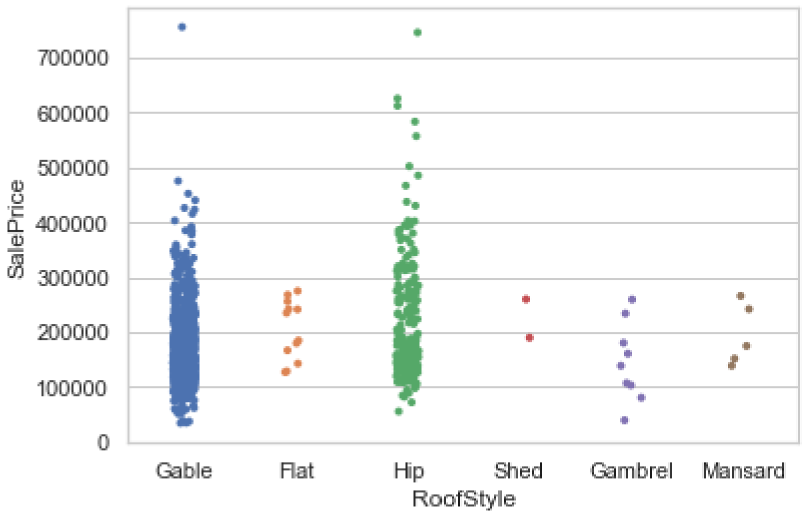

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1


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

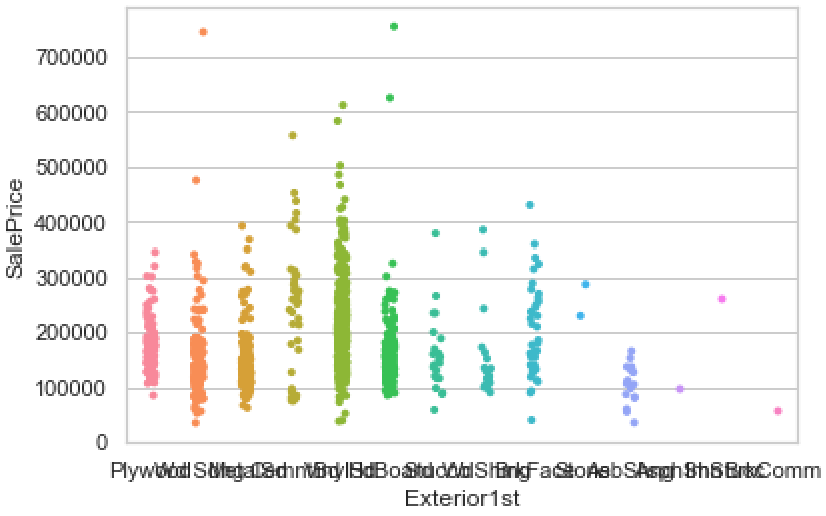

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1
2
3
4
5
6
7
8
9
10
OverallQual

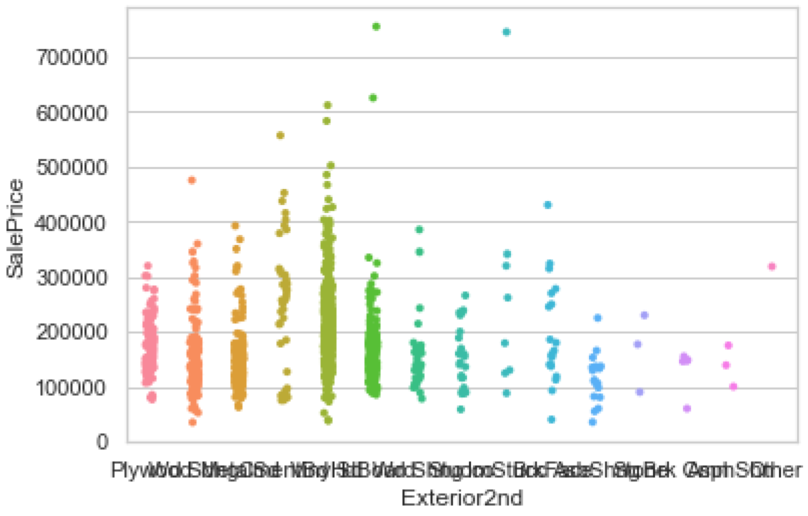

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1
2
3
4
5
6
7
8
9
OverallCond



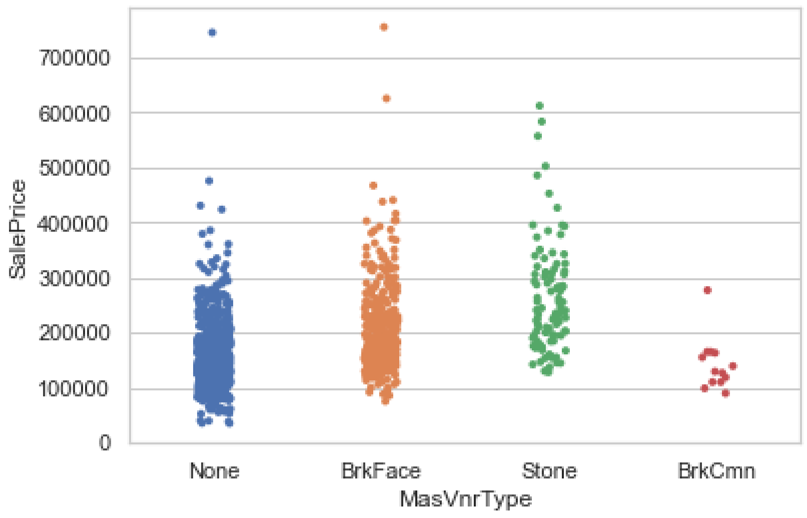
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofStyle

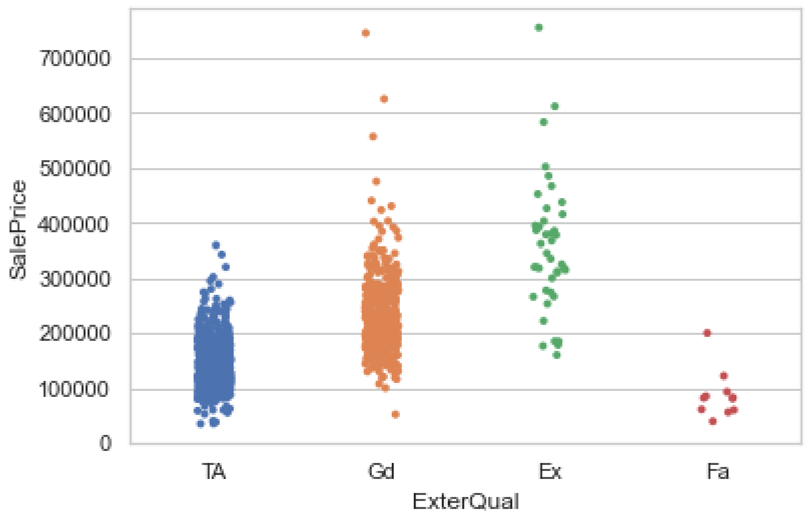


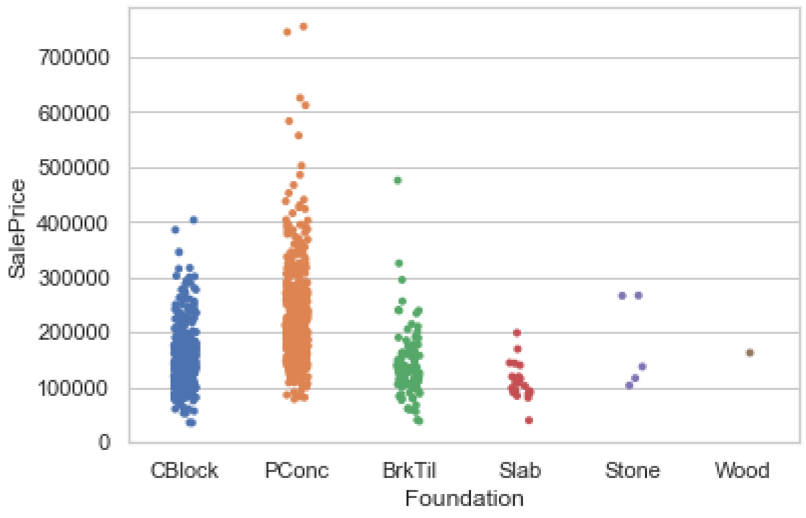
SalePrice
Exterior1st

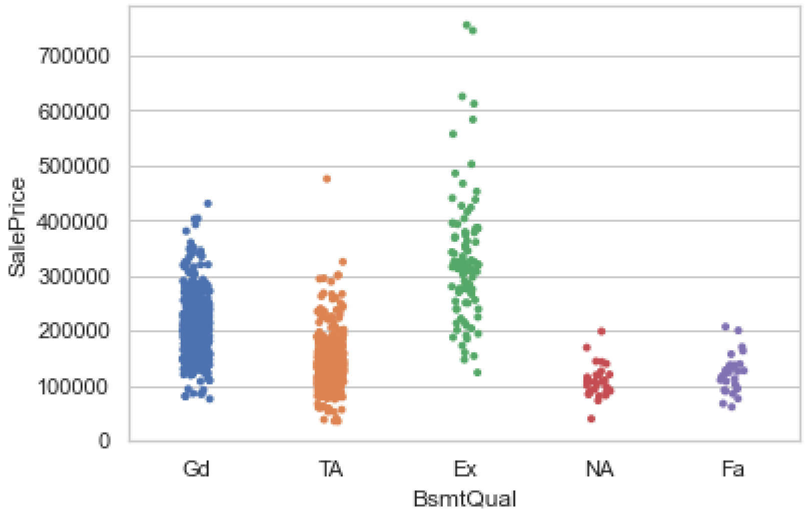


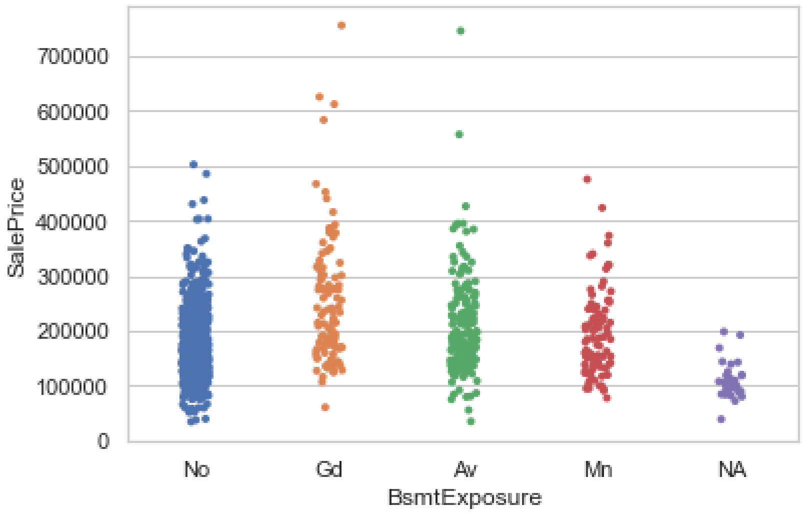
SalePrice
Exterior2nd

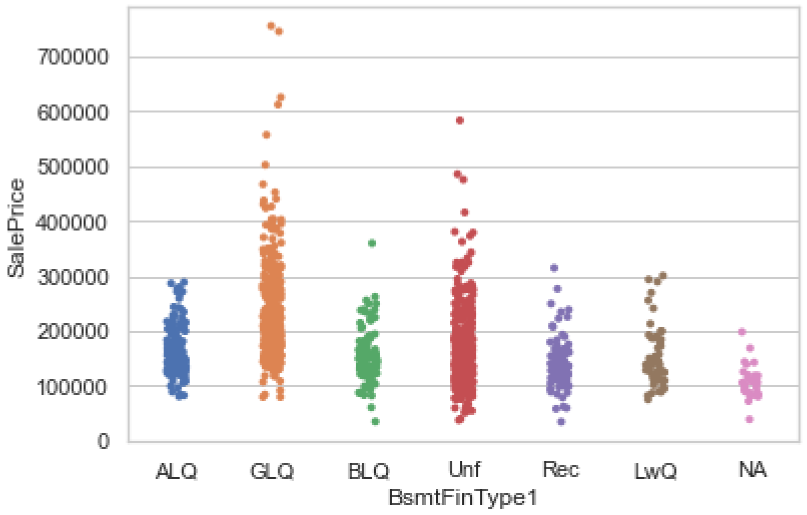

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
None
BrkFace
Stone
BrkCmn
MasVnrType

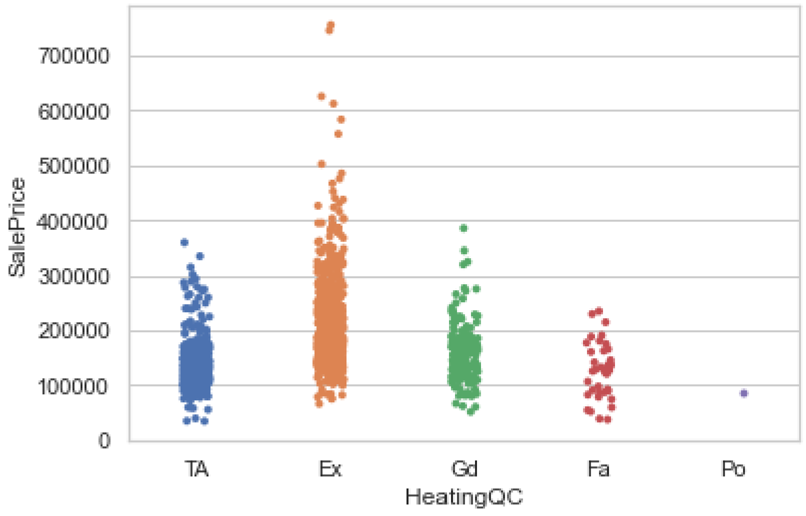

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Ex
Fa
ExterQual


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

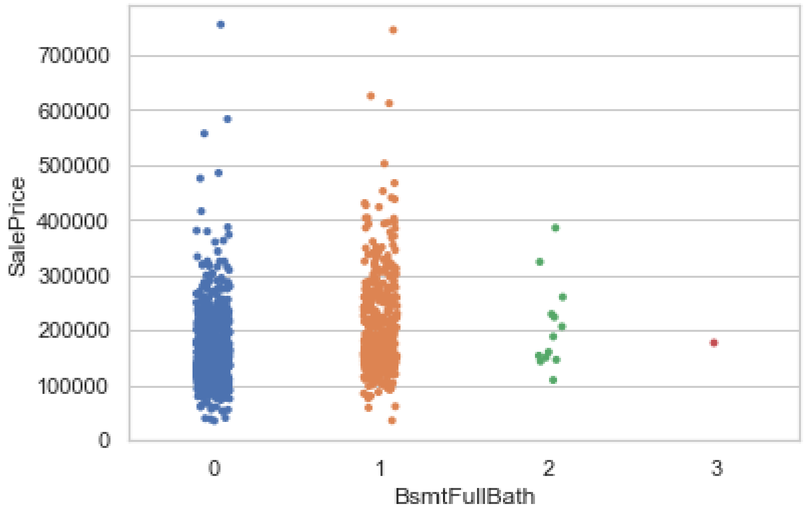

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
Gd
TA
Ex
NA
Fa
BsmtQual


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
No
Gd
Av
Mn
NA
BsmtExposure

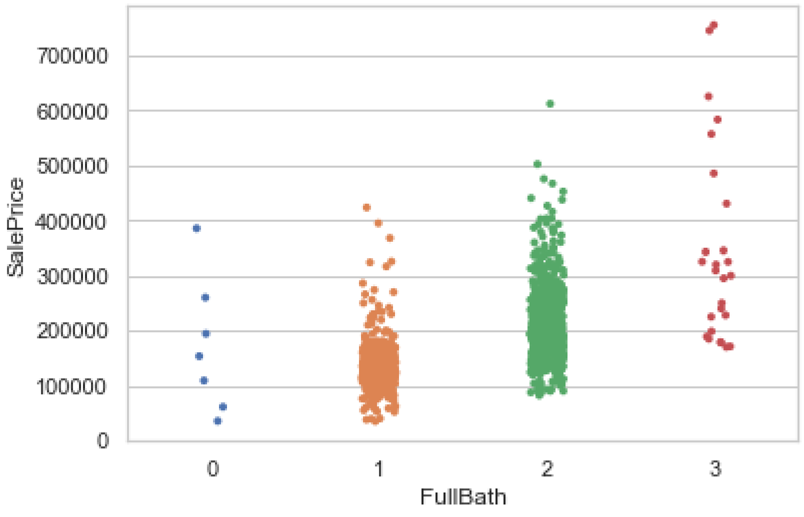

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
ALQ
GLQ
BLQ
Unf
Rec
LwQ
NA
BsmtFinType1



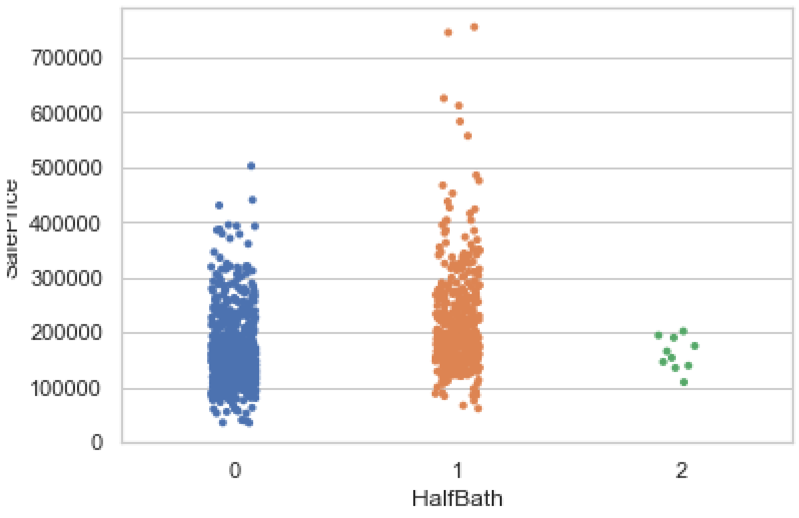
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Ex
Gd
Fa
Po
HeatingQC

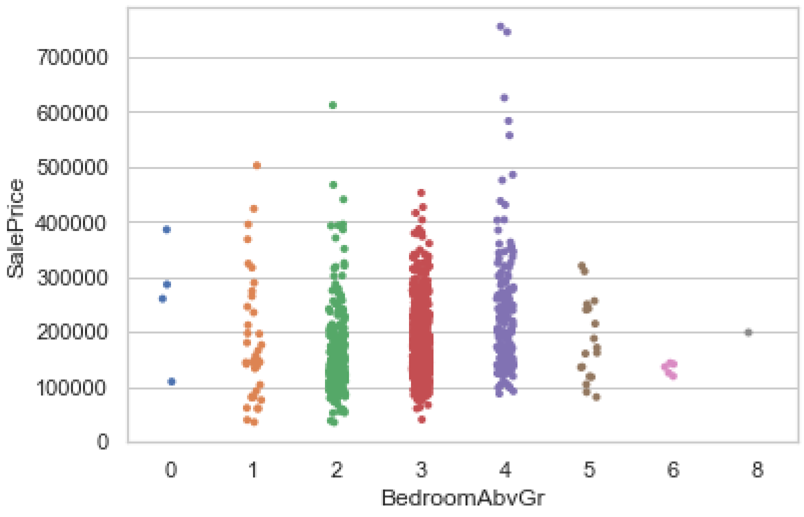


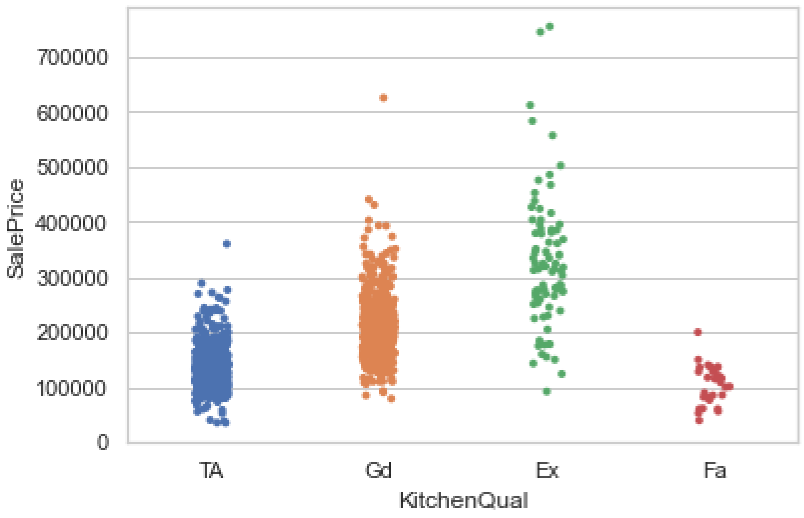
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
BsmtFullBath

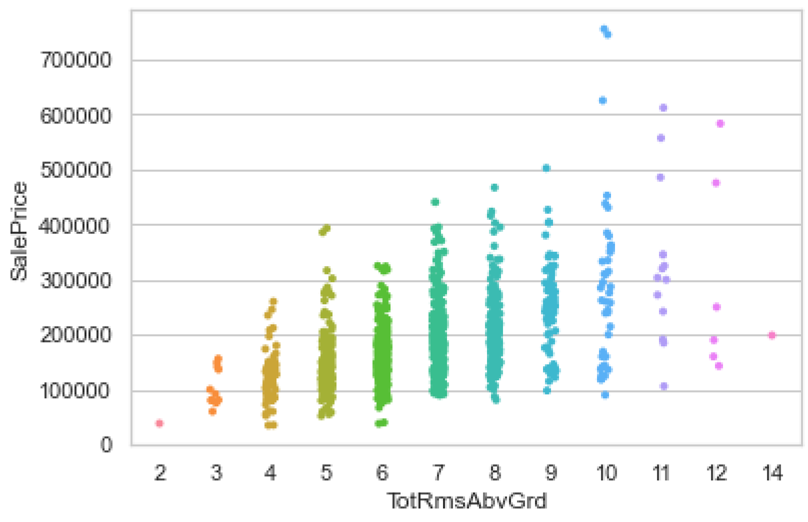


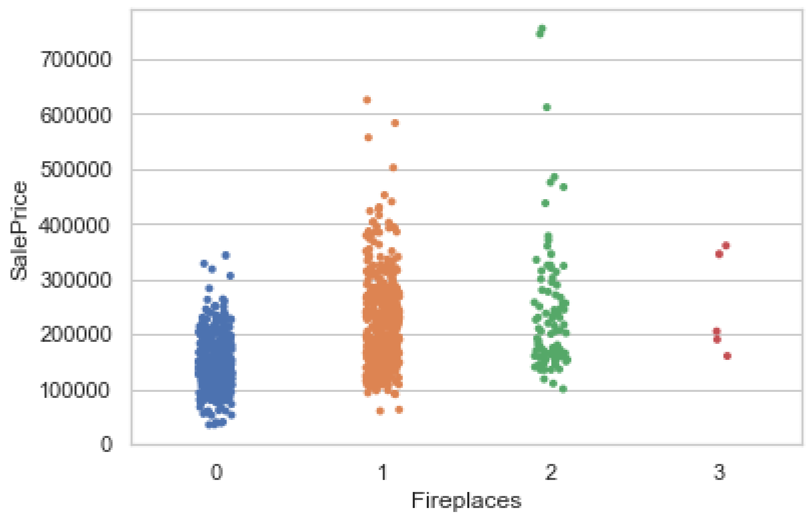
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
FullBath

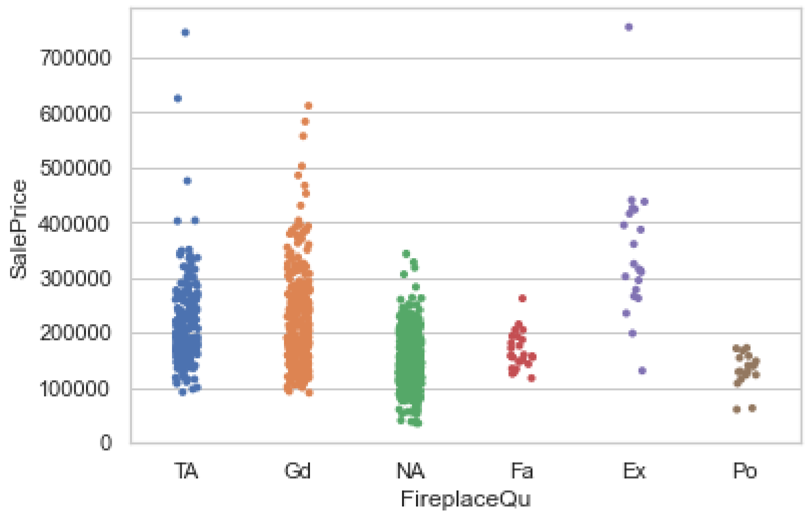

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
HalfBath

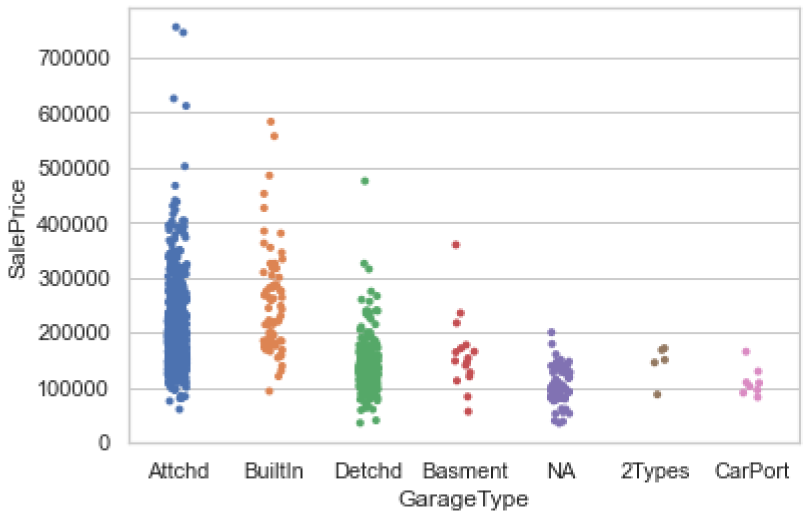

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
4
5
6
8
BedroomAbvGr

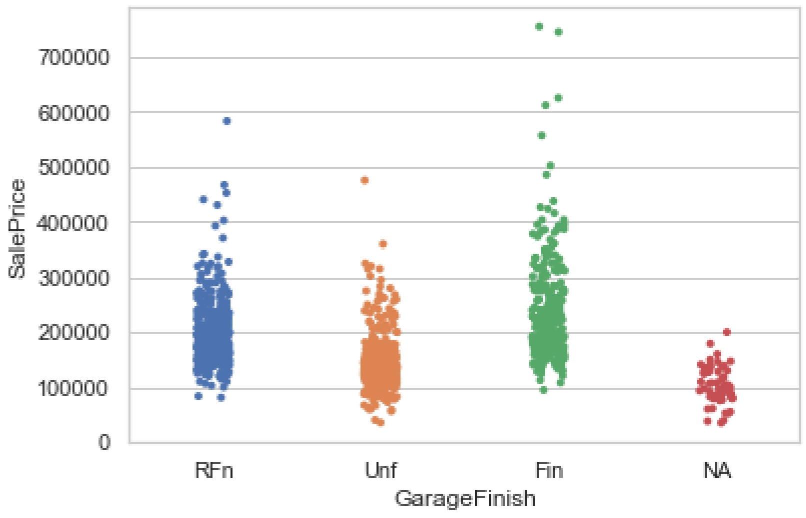

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
Ex
Fa
KitchenQual

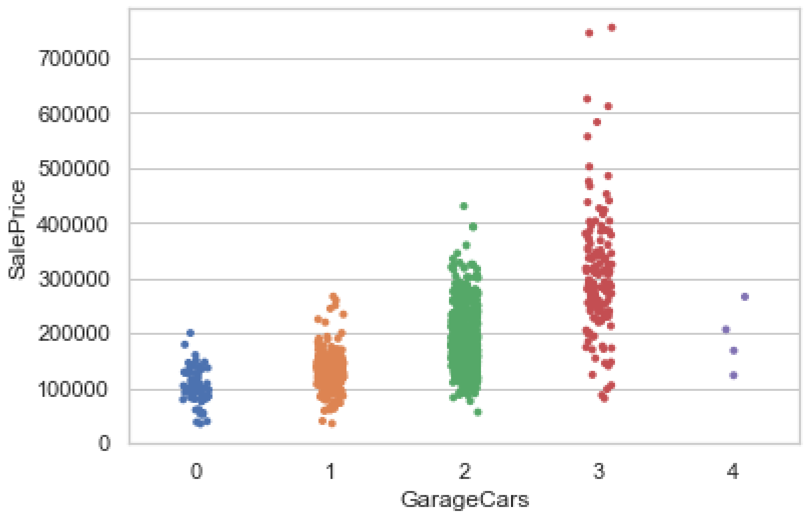

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd

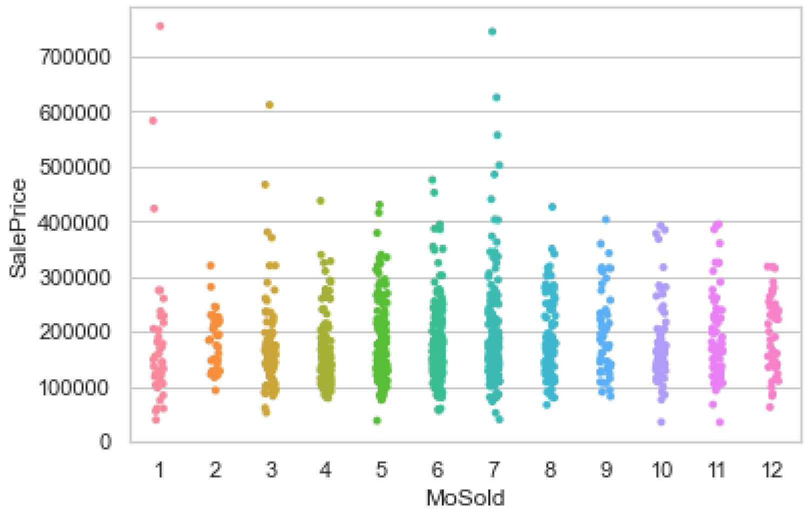

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
1
2
3
Fireplaces

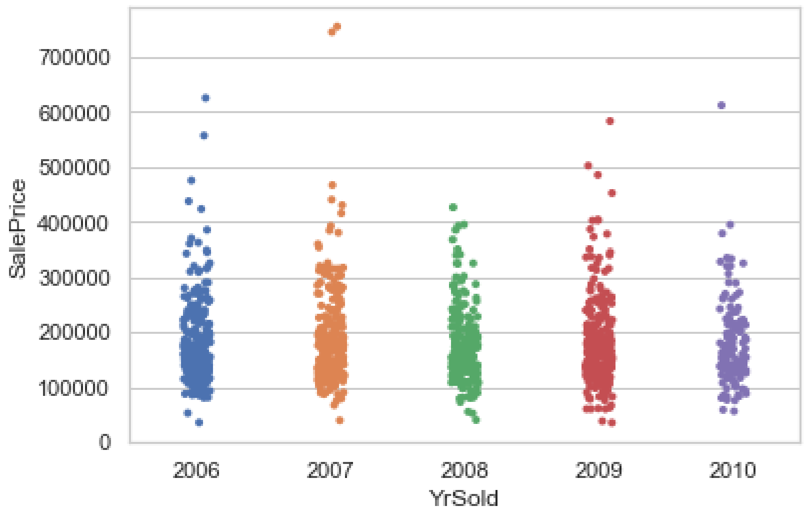

SalePrice
700000
600000
500000
400000
300000
200000
100000
0
TA
Gd
NA
Fa
Ex
Po
FireplaceQu

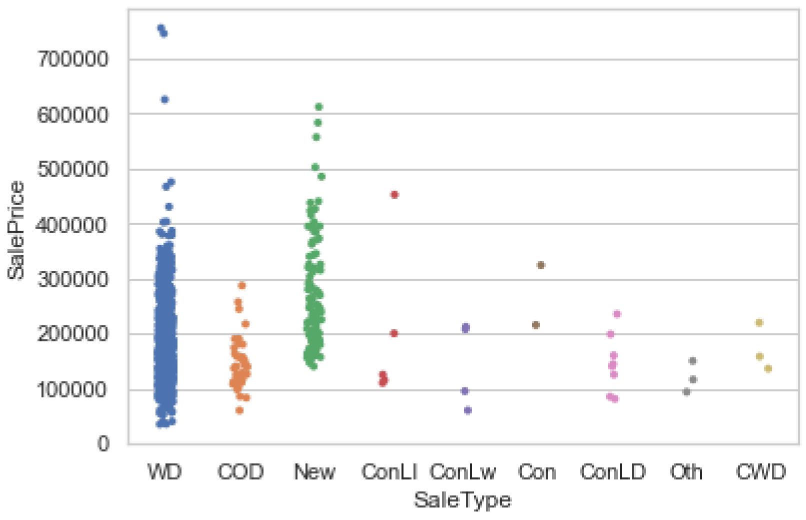

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Attchd
BuiltIn
Detchd
Basment
NA
2Types
CarPort
GarageType

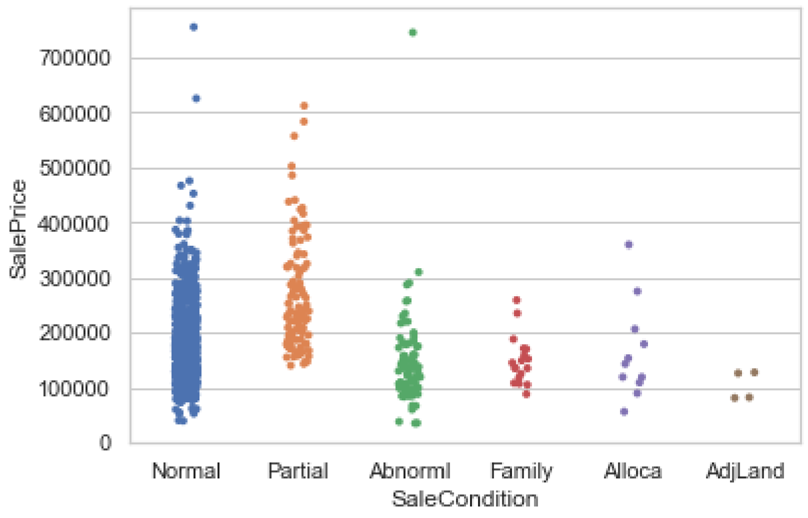

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
RFn
Unf
Fin
NA
GarageFinish


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
0
1
2
3
4
GarageCars


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
1
2
3
4
5
6
7
8
9
10
11
12
MoSold


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
2006
2007
2008
2009
2010
YrSold


SalePrice
700000
600000
500000
400000
300000
200000
100000
0
WD
COD
New
ConLI
ConLw
Con
ConLD
Oth
CWD
SaleType



Univariate Analysis Observation:

1.Low resident Density is dominating in MSZoning column and Commercial the least.

2.Almost all of the road access to property has Pavel than gravel.

3.Most Lot Shape has Regular, barely any have complete irregularity.

4.Majority of Land Contour is Levelled.

5.For all the house, all the public utilities are available.

6.Most of the Lots are Inside Lot barely any lot has more than one frontage side.

7.Almost all the houses have Gentle slope, very few have moderate or severe slope

8.All the neighbourhoods are in the limits of Ames city and Names neighbourhood have a greater number of houses.

9.Almost every house has Normal proximity condition.

10.Except a few negligible houses have normal proximity for condition type 2.

11.Half of the building are 1 Story type building followed by 2 Story. 2.5 Finished Story building are a rarity

12.Gable Type Roof Style seems to be most accustomed in Ames city

13.Standard Shingle is standard for Roof Material

14.Vinyl Sliding is used for exterior covering of 1/3rd houses

15.Most of the house doesn’t have any masonry veer and negligible amount of house have Brick Common

16.Almost every house Exterior quality is either Average. Fair and Excellent are a rarity

17.Majority of the house have Typical Exterior condition and excellent conditions are rare

18.Cinder Block and Poured Concrete are commonly used for House Foundation.

19.Almost all the available basement of the houses are of typical quality and condition

20.More than half of the houses have no basement exposure

21.All most 1/3 houses have good quality of finished basements and another 1/3 are unfinished for 1st basement type and for 2 it is almost most unfinished

22.Every other house uses Gas forced warm air furnace

23.Half of the houses have excellent heating quality condition followed by average

24.Almost every house has central air conditioning

25.Standard Circuit Breakers & Romex are used in majority of the houses.

26.Houses have a typical type of kitchen

27.Typical Functionality for most of the houses very few have deductions in them

28.Half of the house doesn't have fire place.

29.3/5th of all the house have an attached garage

30.Strangly few garages of have no mention of year built in

31.About half of the houses have unfinished garage

32.Most of the garages are of typical quality and condition

33.Majority of the houses have paved drive

34.Maximum houses that for sale have Warranty Deed - Conventional type sale and normal sale condition

35.1/3rd of the MSSubClasses is of 1-STORY 1946 & NEWER ALL STYLES.

36.Most of the house have typical overall quality and condition

37.Majority of the House have no Full Bath in the basement followed 1 full bath and zero half bath

38.Most of the houses have at least 1 bathroom.

39.Most houses have 3 Bedrooms and 1 Kitchen above grade.

40.Majority houses have 6 room above grade

41.Typically most have houses have space for parking at least 1 car in the garage.

42.Rarely any house has a pool area

43.June Month has most number house sales and except for year 2010 every other every 2006-09 have good number sales

Bi-variate Analysis Observation:

1. 2 Story above house have higher starting price.
2. MSZoning doesn't much of difference in house price, although RL has highest price
3. Irregular lot shape have higher starting price
4. Building type, Landcontour and Lot Config doesn't affect the price pretty much
5. 2 Story building has highest house price
6. As the quality of house increases the price of the house also increases.
7. If condition is above average it fetches good price
8. RoofStyle has hardly any effect
9. Brick face and Stone Masonry veneer type have high price
10. Good and Excellent quality exterior fetch good price
11. Brick and tiles and slab are the cheapest house foundation
12. Basement quality,Full Bath,Bedroom above grade, HeatingQC and sale price are directly proportional
13. Kitchen Quality, Total rooms above grade and Fireplaces is directly proportional to sale price
14. Attached garages, Garage Capacity and Fully Finished garages house are much costlier
15. July month is month when costlier houses are sold

Results:

Except for Decision Tree Regressor all the other Regressor have performed well, XGB Regressor has some over fitting compared all the other model i.e., 2% more, Gradient Boosting is best model for our data as it has best r2 cross validation score for every fold evaluate and at an average of 84 % with minimum standard deviation even when the values of random forest and gradient boosting random forest are coinciding because Gradient Boosting usually outperforms Random Forest.

Conclusion:

In the whole dataset Overall Quality of the house has most influence in the sale price of the house. Total Basement Area, Garage Area, Number of fire places in the house and Lot Area are the other factors which have good influence on the sale price prediction.

Outcome of the Study:

Visualizing data helped to negotiate few outliers and biased data. Data Cleaning helps in minimizing the overfitting created during model training and improves the model performance. Decision Tree Algorithm doesn’t have much impact when dataset is small, Random Forest and Gradient Boosting can neglect outliers even when the data is fed with outliers to the machine learning model. When there isn’t much time for training KNN Regressor is best suited for this model. Arresting of the outliers was the most challenge part in this project but it overcome if minimal domain knowledge is exploited into this project.

Limitations of this work and Scope for Future Work:

The type sale price predicted are only limited to particular city and neighbourhood. When we try to price of different city the model will fail. When can avoid this if can introduce GDP of the city and type of neighbourhood it is i.e., urban, suburban and rural. Also, distance from the house to main road, highway and city complex. The prediction can also be improved by adding the type of community the house is located in etc.