HOUSING: PRICE PREDICTION

Submitted by:

Hiral Baluja

INTRODUCTION

- ## *Business Problem Framing*

  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.

  The company is looking at prospective properties to buy houses to enter the market.

  This Project will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

  Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain.

- ## *Conceptual Background of the Domain Problem*

  It is helpful for those who have Domain knowledge of real estate and property. It will help in analysing the data as there are 81 columns in the dataset the domain knowledge and experience can help to understand the correlated things and changing trends over the period of time.

- ## *Review of Literature*

  Affordable housing is, unfortunately, a topic that's largely misunderstood by the public. Myths and misconceptions about affordable housing developments are based on fear around negative stereotypes, property values, and the change it brings to neighbourhoods—all of which are common arguments in opposition of a new affordable housing community

  Real estate investment plays a significant role in home appreciation.

  The process in which investors procure a property initiates a chain of events that stimulates the economy. Every step in rehabbing and/or

flipping a house involves the exchange of money in return for goods and services.

- ***Motivation for the Problem Undertaken***

A positive externality occurs when a third party benefits from the production or consumption of a good. In many cases, building the right kind of housing can have benefits to the rest of society. Therefore, the social benefit of good quality housing can be greater than the private benefit that property developers gain.

The model will be a good way for the management to understand the pricing dynamics of a new market.

Analytical Problem Framing

- ***Mathematical/ Analytical Modelling of the Problem***

Data consist of 1163 rows and 81 columns, because of the large number of columns it is difficult to analyse easily also to find significant columns that are helpful to predict the price of the house.
EDA part takes more time for this problem also as per the domain we can not eliminate any observation for its odd value. Also, many columns have many null values.

- ***Data Sources and their formats***

The sample data is provided by the US-based housing company named Surprise Housing 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 in csv format. First I have read the csv file and converted it into a data frame. This is a regular regression problem statement.

Importing Important Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Loading Train Dataset

In [2]:
```
# To display maximum rows and columns in the dataset
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [3]:
```
# Loading Train file
train_df = pd.read_csv('train.csv')
train_df
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm
5	1197	60	RL	58.0	14054	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm
6	561	20	RL	NaN	11341	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm
7	1041	20	RL	88.0	13125	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	Sawyer	Norm
8	503	20	RL	70.0	9170	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	Edwards	Feedr

- ***Data Pre Processing***

  Pre-processing Pipe Line is an important step towards the data modelling, To make data ready for prediction.
  Following Steps I followed for Pre Processing:

- ***Handling the null values***

  I have dropped columns which have null values more than 70%, filling them is meaningless.

In [20]:
```
# Dropping the columns containing more than 70% of the null values
train_df.drop(['PoolQC', 'MiscFeature', 'Alley', 'Fence'], axis = 1, inplace = True)
```

For treating the null values of the columns, I have used mean and mode method I.e. I have replaced null values with the mode in the categorical columns and with the mean for numerical columns.

Treating null values

1) The columns LotFrontage, MasVnrArea and GarageYrBlt are numeric in nature, lets fill the null values using mean values as there ar in these columns.

2) The remaining columns seems to be categorical so filling the null values using mode method.

In [21]:
```
# Filling null values of numerical columns with their mean values
col1 = train_df.columns.values
for i in range(0, len(col1)):
    if train_df[col1[i]].dtypes != 'object':
        train_df[col1[i]].fillna(train_df[col1[i]].mean(), inplace = True)
```

In [22]:
```
# Filling null values of the object data columns with their mode values
col2 = train_df.columns.values
for i in range(0, len(col2)):
    if train_df[col2[i]].dtypes == 'object':
        train_df[col2[i]].fillna(train_df[col2[i]].mode()[0], inplace = True)
```

- ***Handling outliers and skewness***

  Considering the domain of the problem the outliers can be the real values and which leads to the skewness of the columns .By analysing the data in

the columns, it is clear that the number of zeros is the main reason for the skewness. So for both issues, I have used power transformation.

```
# Removing skewness using yeo-johnson  method to get better prediction
skew1 = ["MSSubClass", "LotArea", "MasVnrArea", "BsmtHalfBath", "BsmtUnfSF", "2ndFlrSF", "GrLivArea", "KitchenAb

from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method = 'yeo-johnson')
'''
parameters:
method = 'box-cox' or 'yeo-johnson'
'''
```

```
"\nparameters:\nmethod = 'box-cox' or 'yeo-johnson'\n"
```

```
train_df[skew1] = scaler.fit_transform(train_df[skew1].values)
train_df[skew1].head()
```

	MSSubClass	LotArea	MasVnrArea	BsmtHalfBath	BsmtUnfSF	2ndFlrSF	GrLivArea	KitchenAbvGr	HalfBath	Fireplaces	WoodDeckSF	OpenPorchSF	A
0	1.370435	-1.306083	-0.822896	-0.238775	0.916764	-0.871789	-1.281768	0.0	-0.782707	0.779453	-0.960144	1.423871	
1	-1.167999	1.356458	-0.822896	-0.238775	1.042594	-0.871789	1.377184	0.0	-0.782707	0.779453	0.783831	1.428474	
2	0.490047	0.113089	-0.822896	-0.238775	-0.510359	1.177246	1.060650	0.0	1.268421	0.779453	1.047285	1.208580	
3	-1.167999	0.530989	1.385487	-0.238775	1.178365	-0.871789	0.775546	0.0	-0.782707	0.779453	-0.960144	1.178672	
4	-1.167999	1.497522	1.140684	4.188040	-0.230032	-0.871789	0.322952	0.0	-0.782707	0.779453	1.139237	-1.062308	

- ***Encoding***

  For the encoding of all the categorical columns, I used Ordinal Encoder.

Encoding Categorical Variables

```
column = ['ExterQual', 'ExterCond', 'BsmtQual', 'BsmtCond', 'HeatingQC', 'KitchenQual', 'FireplaceQu', 'GarageQu
for i in column:
    train_df[i] = train_df[i].replace({'Ex':5, 'Gd':4, 'TA':3, 'Fa':2, 'Po':1, 'None':0})
```

```
# Encoding train dataframe using OrdinalEncoder
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
for i in train_df.columns:
    if train_df[i].dtypes == "object":
        train_df[i] = enc.fit_transform(train_df[i].values.reshape(-1, 1))
```

```
train_df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType
0	1.370435	3.0	70.98847	-1.306083	1.0	0.0	3.0	4.0	0.0	13.0	2.0	2.0	4.0
1	-1.167999	3.0	95.00000	1.356458	1.0	0.0	3.0	4.0	1.0	12.0	2.0	2.0	0.0
2	0.490047	3.0	92.00000	0.113089	1.0	0.0	3.0	1.0	0.0	15.0	2.0	2.0	0.0
3	-1.167999	3.0	105.00000	0.530989	1.0	0.0	3.0	4.0	0.0	14.0	2.0	2.0	0.0
4	-1.167999	3.0	70.98847	1.497522	1.0	0.0	3.0	2.0	0.0	14.0	2.0	2.0	0.0

- ***Data Inputs- Logic- Output Relationships***

To analysis the relation between input and output first I have separated the numerical, categorical, and temporal columns.

```
# Checking for categorical columns
categorical_col = []
for i in train_df.dtypes.index:
    if train_df.dtypes[i] == 'object':
        categorical_col.append(i)

print('Categorical columns are :', categorical_col)
print('\n')

# Checking for numerical columns
numerical_col = []
for i in train_df.dtypes.index:
    if train_df.dtypes[i] != 'object':
        numerical_col.append(i)

print('Numerical columns are :', numerical_col)
```

```
Categorical columns are : ['MSZoning', 'Street', 'LotShape', 'LandContour', 'LotConfig', 'LandSlope', 'Neighborhood
', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'Ma
sVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFin
Type2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType
', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition']


Numerical columns are : ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'MasVnrArea', 'BsmtF
inSF1', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath'
, 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF
', 'OpenPorchSF', 'MoSold', 'YrSold', 'SalePrice', 'AgeBuilt', 'AgeRemod', 'AgeGarage']
```

Model's Development and Evaluation

- ***Identification of possible problem-solving approaches (methods)***

To prepare the data for data modelling it needs to be clean and scale.The size of this dataset is 1163 rows and 81 columns.
The number of columns is more therefore to handle this number of columns it is important to analyse the significance of the column.Two

columns Utilities and Id have all the unique values and PoolQC and MiscFeatures have almost null values therefore I have deleted these four columns. Moreover, many columns have many zeros observations which increase the skewness of the columns also taking the domain of the problem into the consideration it is possible that the values may vary for each column. So we can not remove the outliers also.

- ***Testing of Identified Approaches (Algorithms)***

Feature Selection Algorithms:
RFE , SelectFromModel.
Some Standard Regression algorithms:
Linear Regression, Decision Tree Regressor, Support Vector Regressor, Kneigbors Regressor.
Ensemble Techniques:
Random Forest Regressor, AdaBoost Regressor, Gradient BoostRegressor
Hyper Parametric Tuning of Gradient Boost Regressor

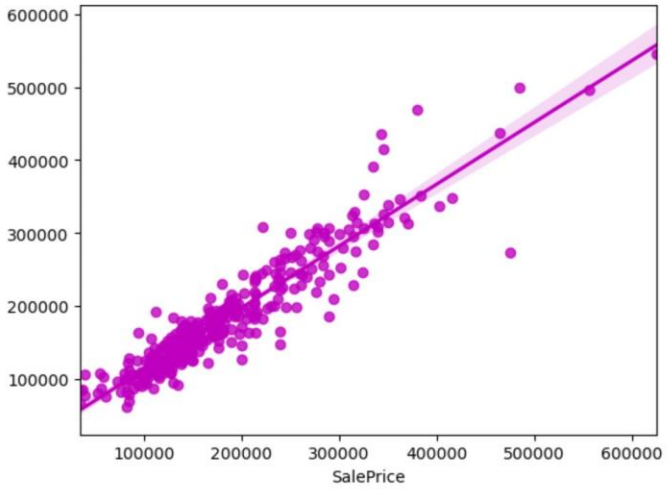
- ***Run and Evaluate selected models***

As mentioned above I have applied 7 to 8 algorithms to predict the label. But I selected Bagging Regressor as the best model on the basis of the difference between cross-validation score and r2 score and mean square root value and the plot between the predicted value and actual values.

```
 Checking R2 score for BaggingRegressor
R = BaggingRegressor()
R.fit(x_train, y_train)

 prediction
redBR = BR.predict(x_test)
rint('R2_Score:', r2_score(y_test, predBR))

 Metric Evaluation
rint('MAE:', metrics.mean_absolute_error(y_test, predBR))
rint('MSE:', metrics.mean_squared_error(y_test, predBR))
rint("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, predBR)))

 Checking cv score for BaggingRegressor
core = cross_val_score(BR, train_df_x, y, cv = 5)
rint(score)
rint("cross validation score:  ", score.mean())
rint("Difference between R2 score and cross validation score is - ", r2_score(y_test, predBR)-abs(score.mean()))


 Visualizing the predicteed values
ns.regplot(y_test, predBR, color = "m")
lt.show()
```

```
R2_Score: 0.8741821886004756
MAE: 20881.52678062678
MSE: 892088347.0002563
RMSE: 29867.848047695974
[0.84624409 0.81501101 0.77557908 0.84219191 0.83468349]
cross validation score:   0.8227419158952533
Difference between R2 score and cross validation score is -  0.051440272705222334
```

```
MSE: 892088347.0002563
RMSE: 29867.848047695974
[0.84624409 0.81501101 0.77557908 0.84219191 0.83468349]
cross validation score:   0.8227419158952533
Difference between R2 score and cross validation score is -  0.051440272705222334
```



The Bagging Regressor model is getting 87.41% R2 score with cross validation score 82.27%.The difference between R2 score and cross validation score is 0.051.

From the difference between R2 score and Cross Validation Score I can conclude that Bagging Regressor as best fitting model as it is giving less difference compare to other models. Let's perform Hyperparameter tuning to increase the model accuracy.

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

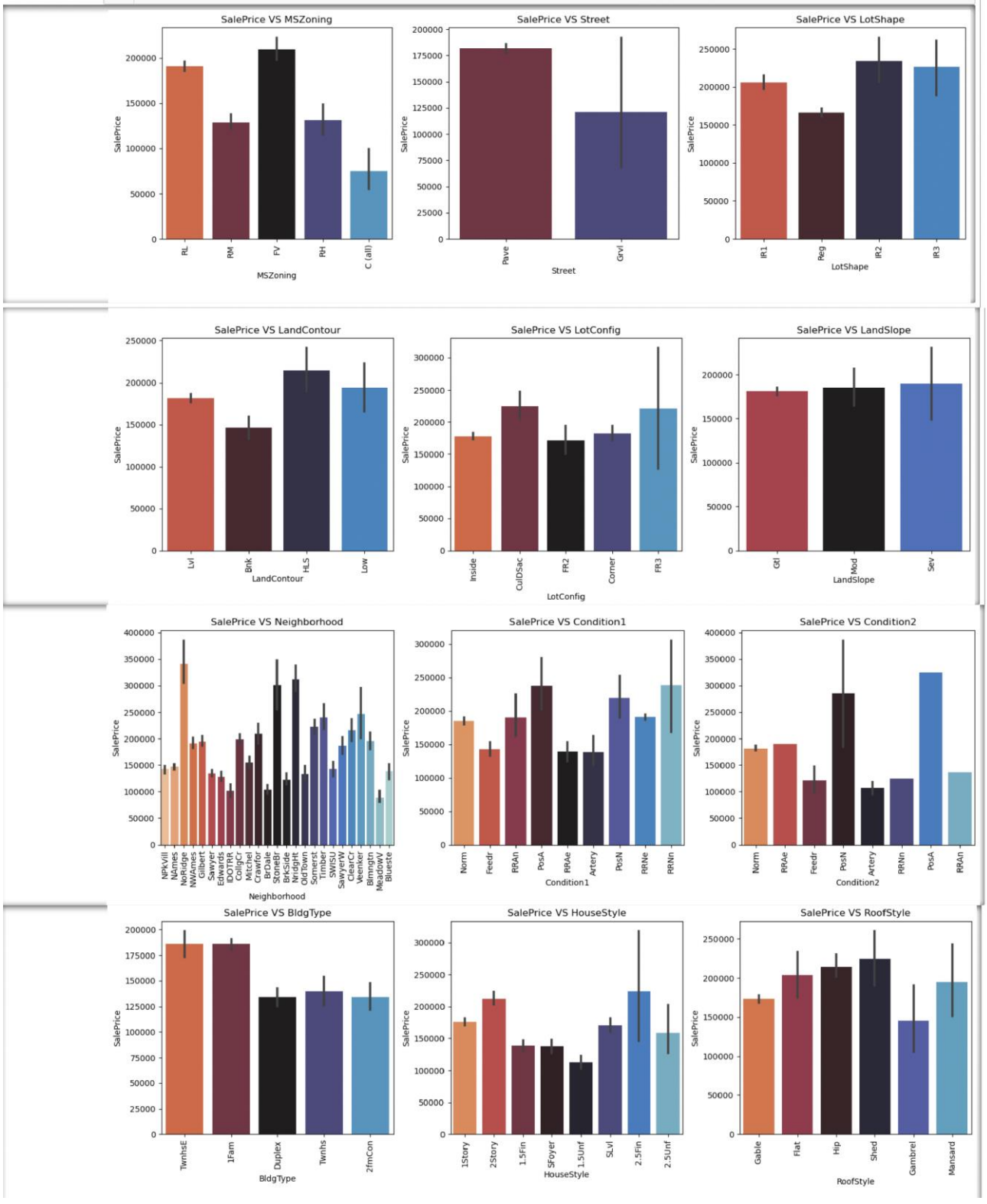
Key Metrics that I have used in this project are r2 score, Mean Square Error(MSE), Root Mean Square Error(RMSE), and for checking the fitting of the model I have used cross-validation score
R Square measures how much variability in the dependent variable can be explained by the model.
Mean Square Error is an absolute measure of the goodness for the fit.
Cross-validation is a statistical method used to estimate the skill of machine learning models.
Also, I have used the regplot to analyze the best fitted line of the selected model.

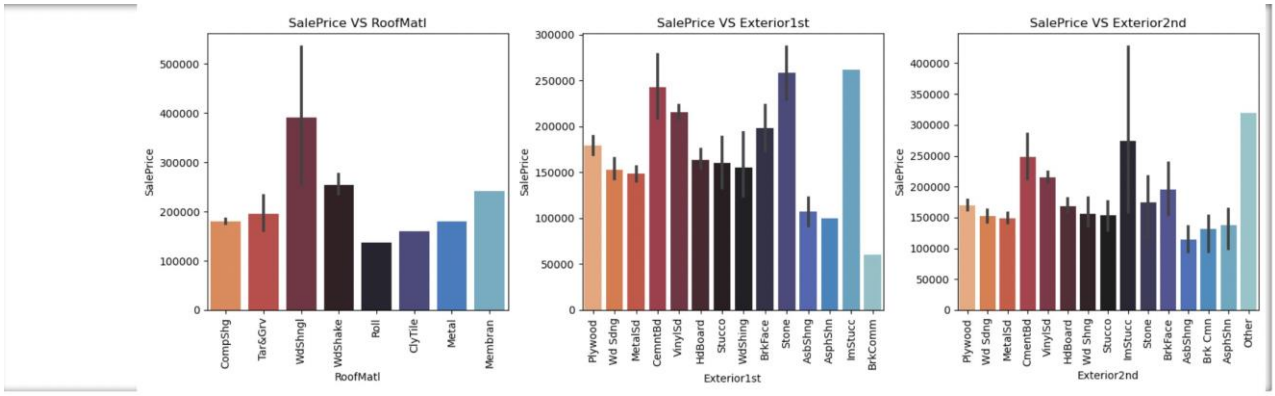
· *Visualizations*

1. Plotting Nominal Variables VS SalePrice

In [34]:

```
# Checking relation between nominal categorical variable and target variable
nominal_data1 = ['MSZoning', 'Street', 'LotShape', 'LandContour', 'LotConfig', 'LandSlope', 'Neighborhood', 'Con
plt.style.use('default')
plt.figure(figsize = (15, 25))
for i in range(len(nominal_data1)):
    plt.subplot(5, 3, i+1)
    sns.barplot(y = train_df['SalePrice'], x = train_df[nominal_data1[i]], palette = 'icefire_r')
    plt.title(f'SalePrice VS {nominal_data1[i]}', fontsize = 12)
    plt.xticks(rotation = 90)
    plt.tight_layout()
```

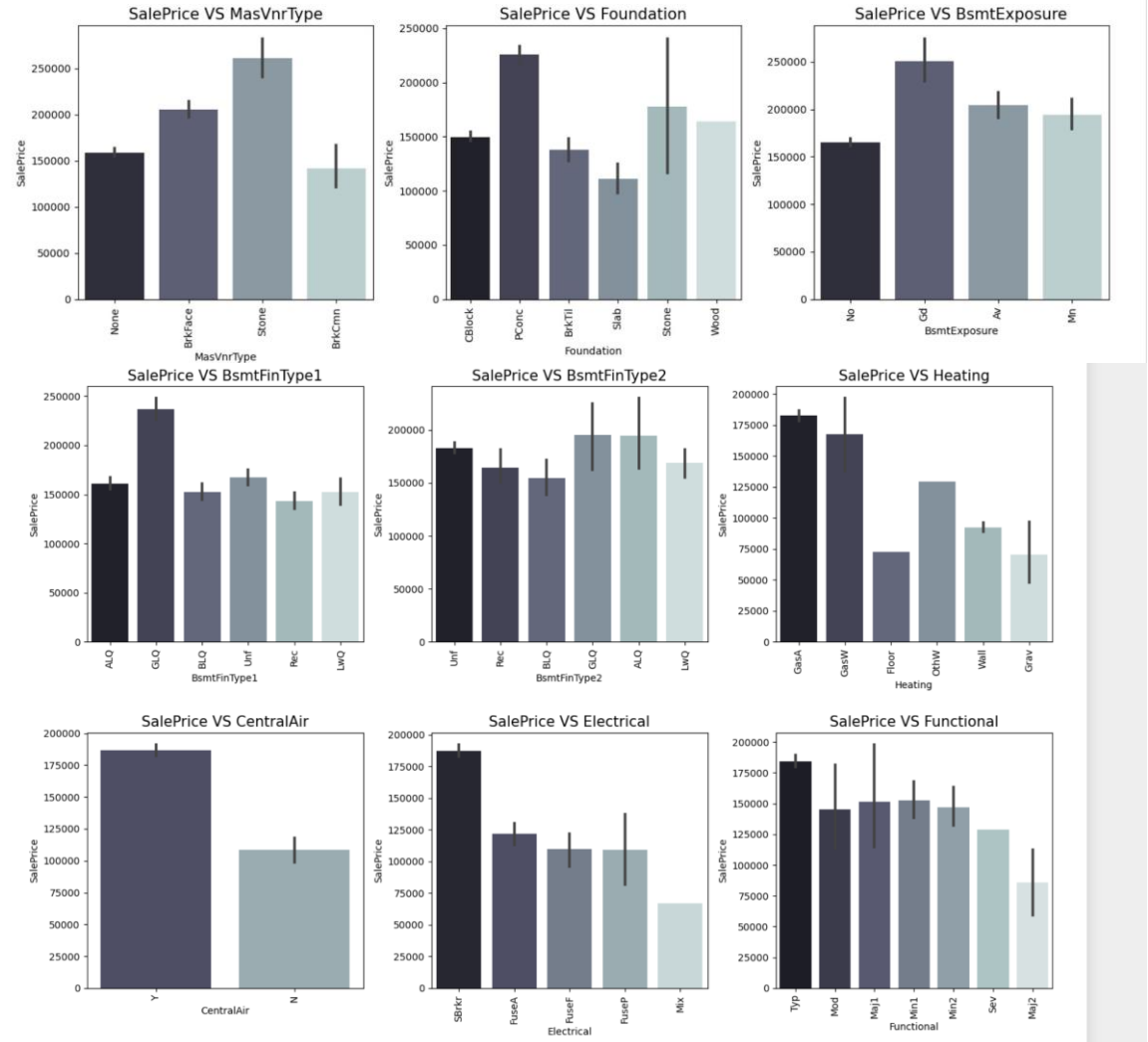


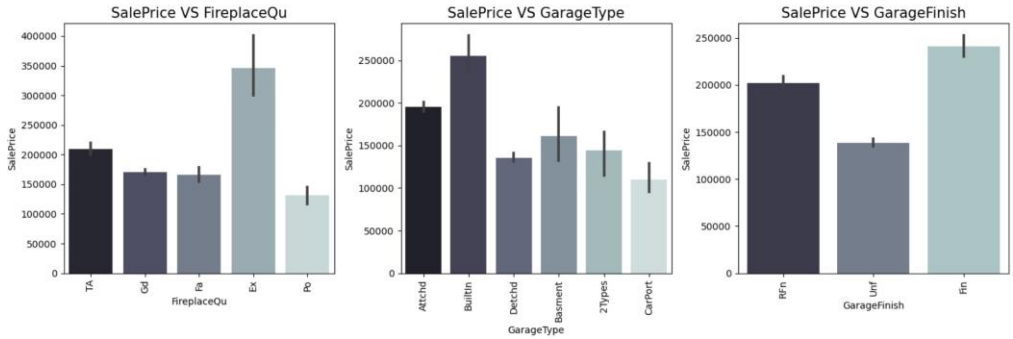


2. Plotting Nominal Categorical Variable VS SalePrice

In [35]:

```
# Checking relation between nominal categorical variables and target variables
nominal_data2 = ['MasVnrType', 'Foundation', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'Central
plt.style.use('default')
plt.figure(figsize = (15, 25))
for i in range(len(nominal_data2)):
    plt.subplot(5, 3, i+1)
    sns.barplot(y = train_df['SalePrice'], x = train_df[nominal_data2[i]], palette = 'bone')
    plt.title(f'SalePrice VS {nominal_data2[i]}', fontsize = 15)
    plt.xticks(rotation = 90)
    plt.tight_layout()
```

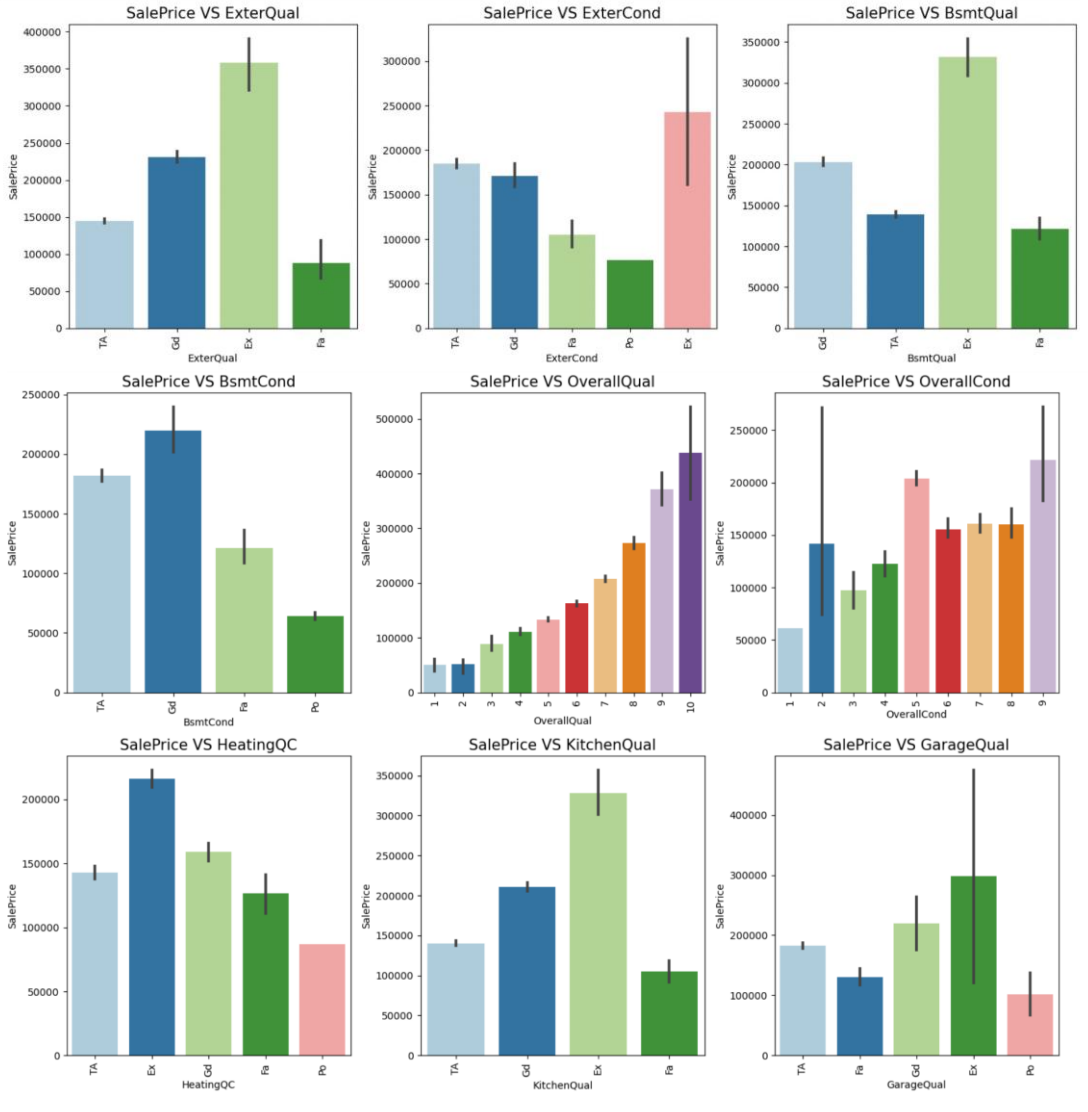


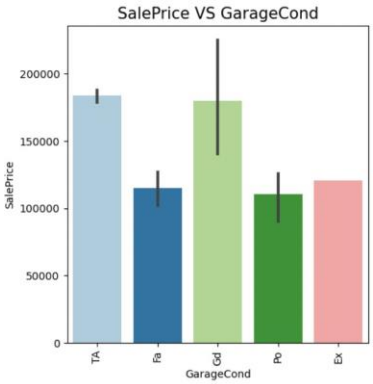


3. Plotting Ordinal Categorical Variables VS SalePrice

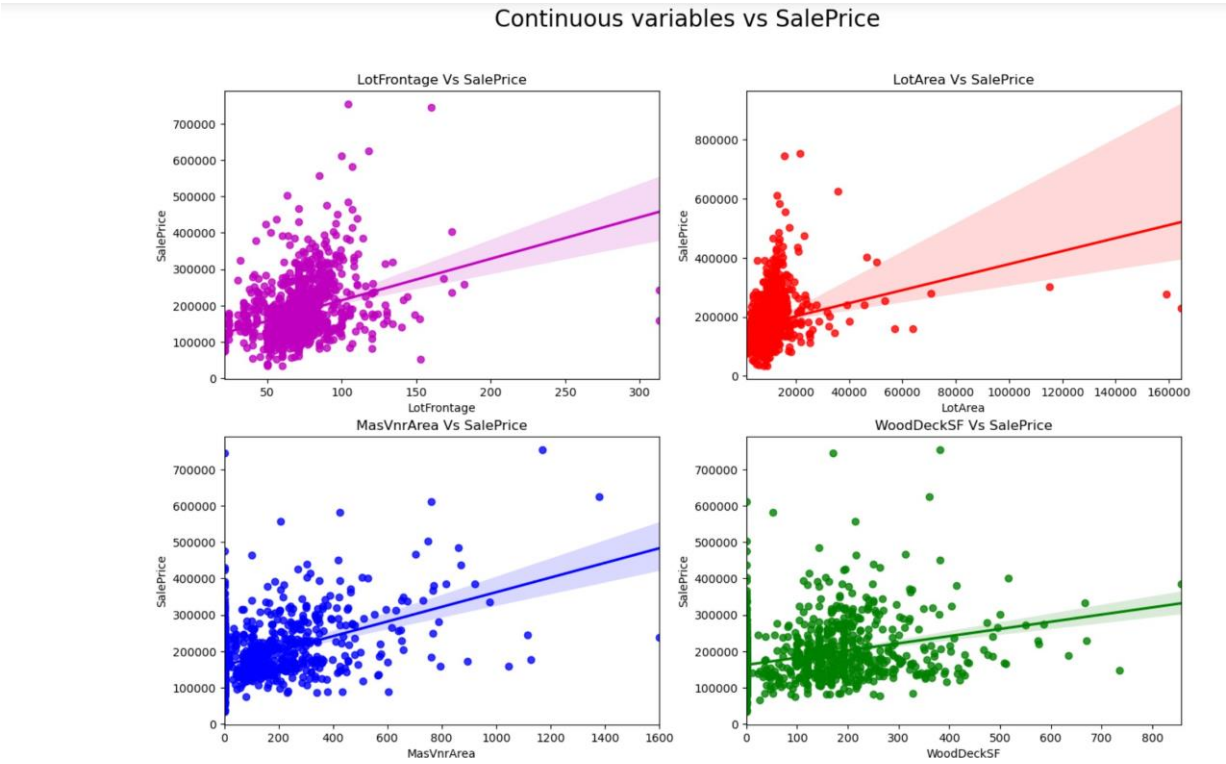
In [36]:

```
between ordinal categorical variables and target variable
erQual', 'ExterCond', 'BsmtQual', 'BsmtCond', 'OverallQual', 'OverallCond', 'HeatingQC', 'KitchenQual', 'GarageQ
ilt')
: (15, 25))
rdinal_data)):
, i+1)
train_df['SalePrice'], x = train_df[ordinal_data[i]], palette = "Paired")
Price VS {ordinal_data[i]}", fontsize = 15)
ion = 90)
()
```

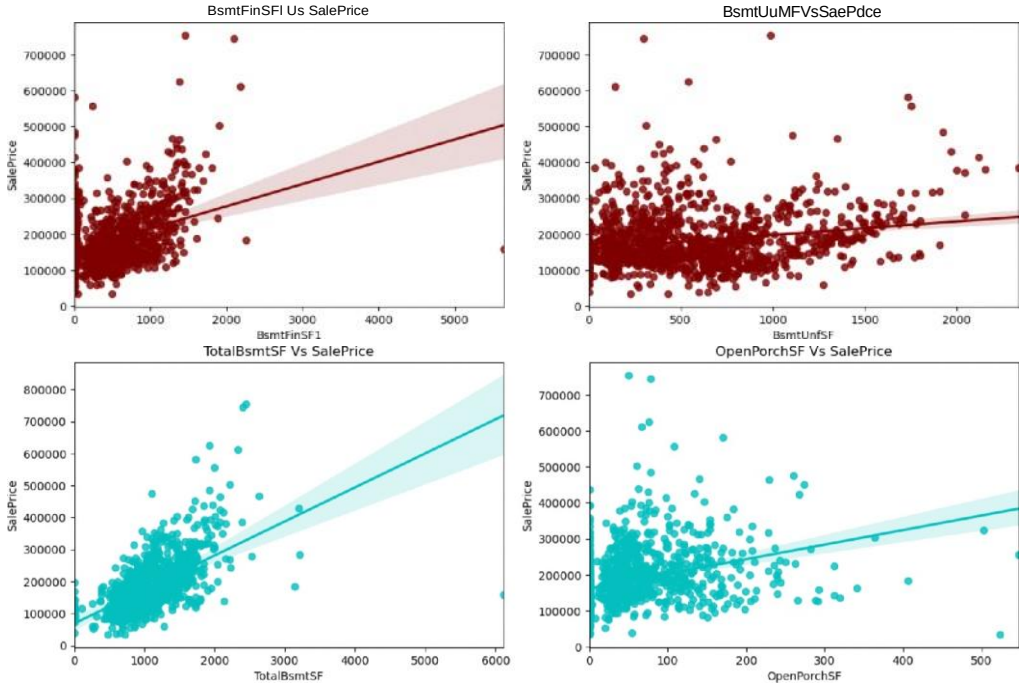




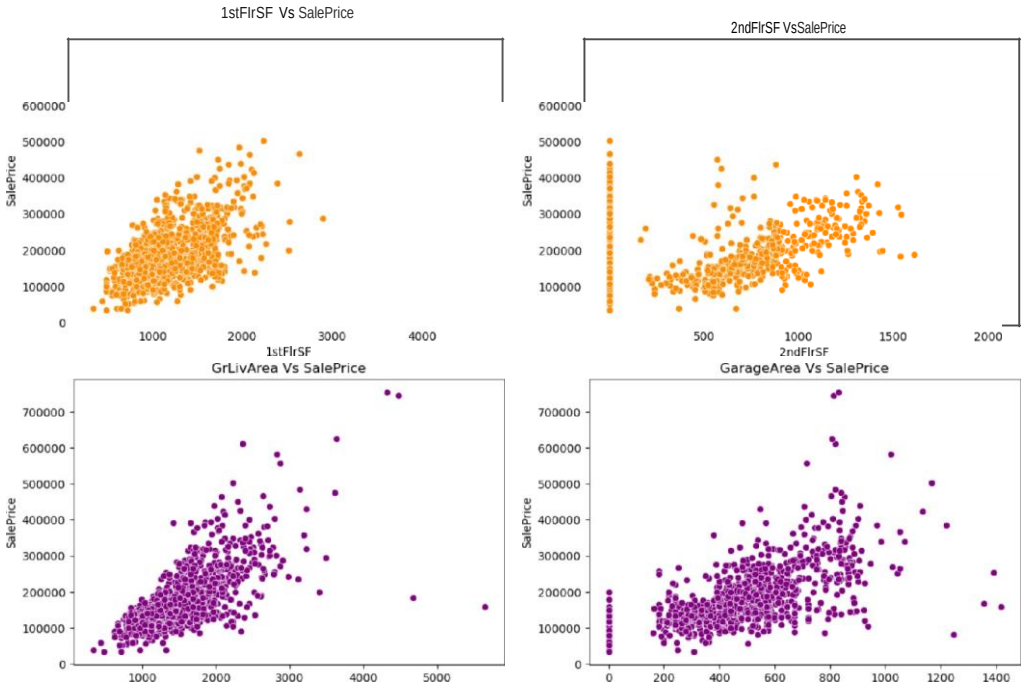
4. Plotting Continous Variable VS SalePrice

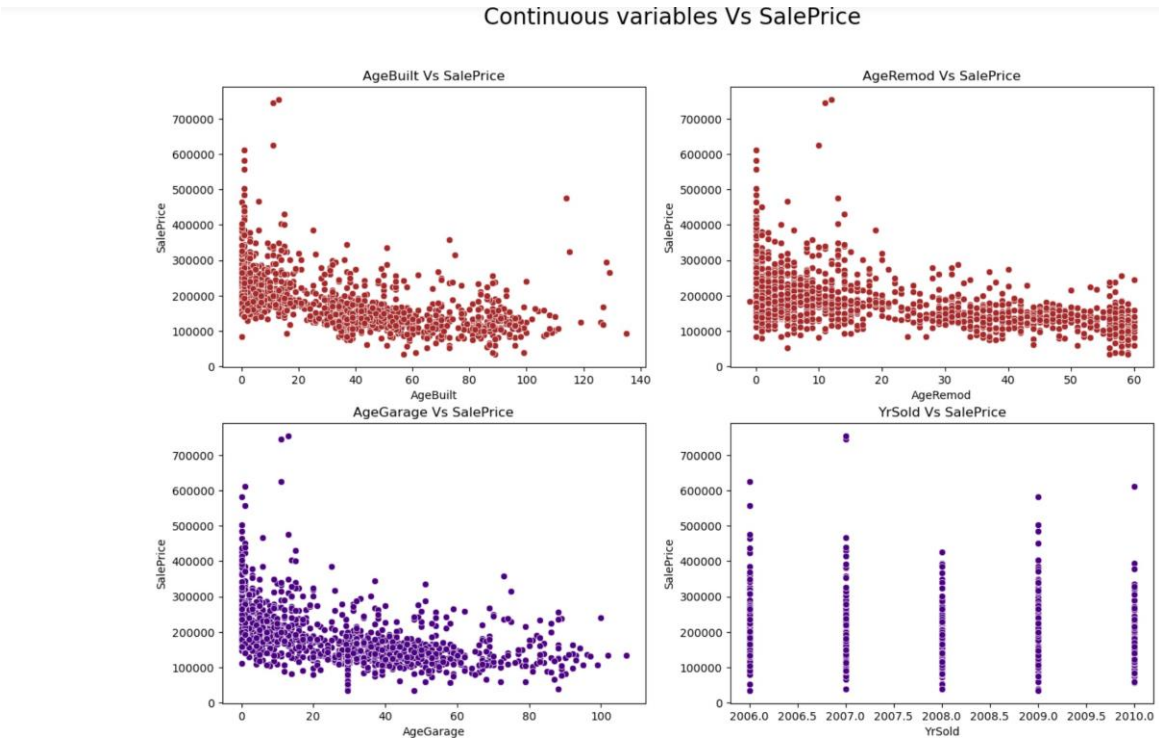


Continuous variables Vs SalePrice

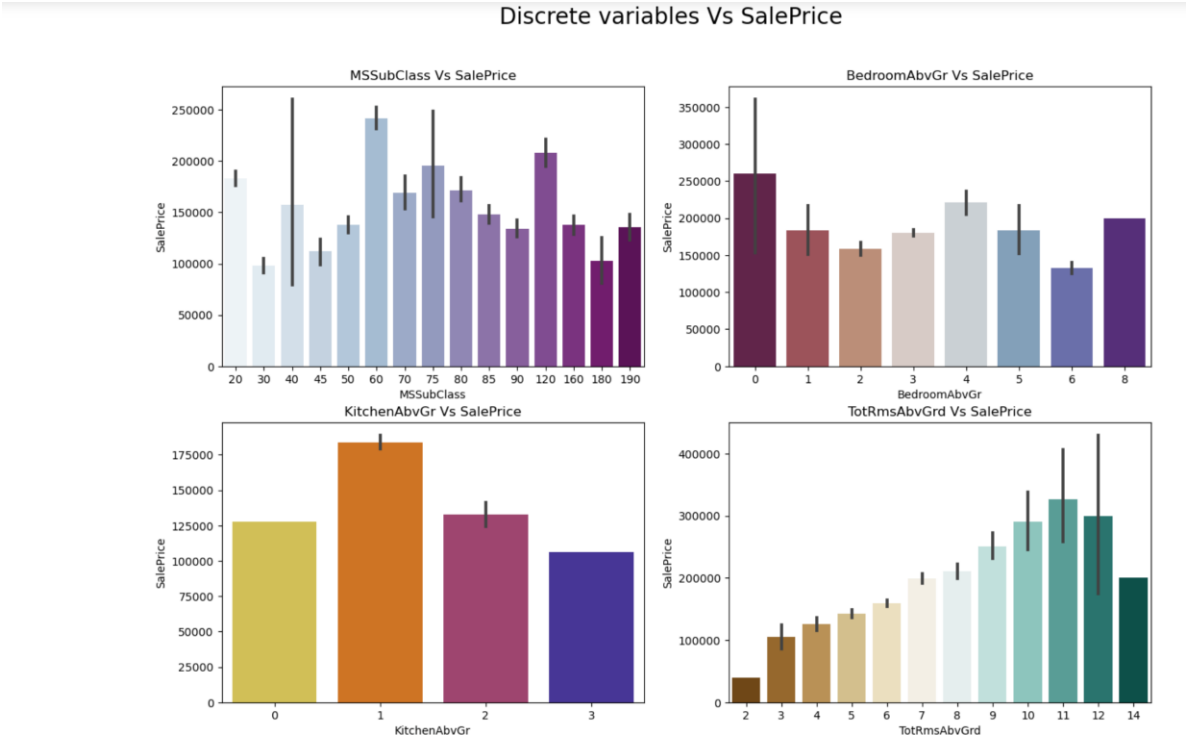


Continuous variables Vs SalePrice

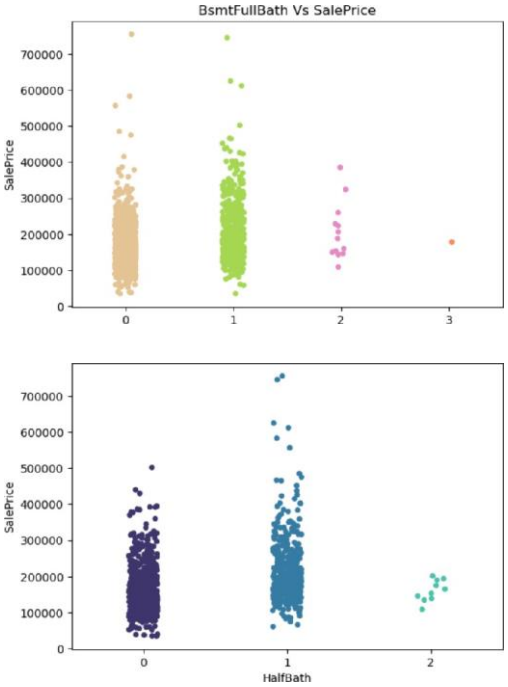


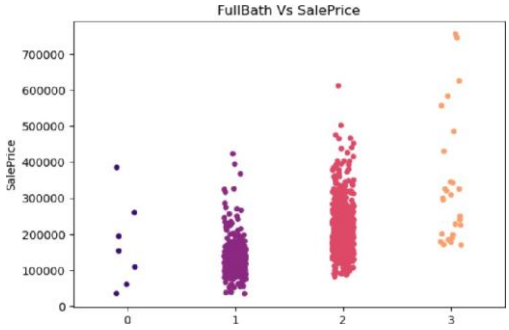


5. Plotting Discrete Variable VS SalePrice

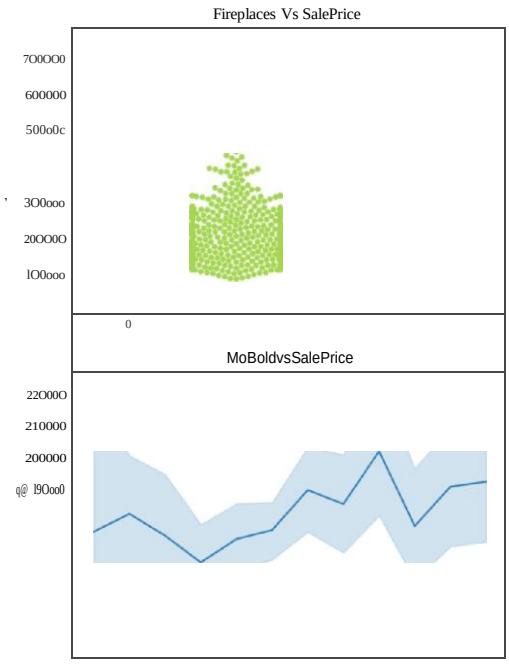


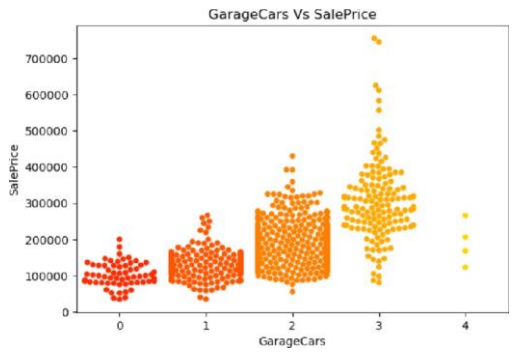
Discrete variables Vs SalePrice





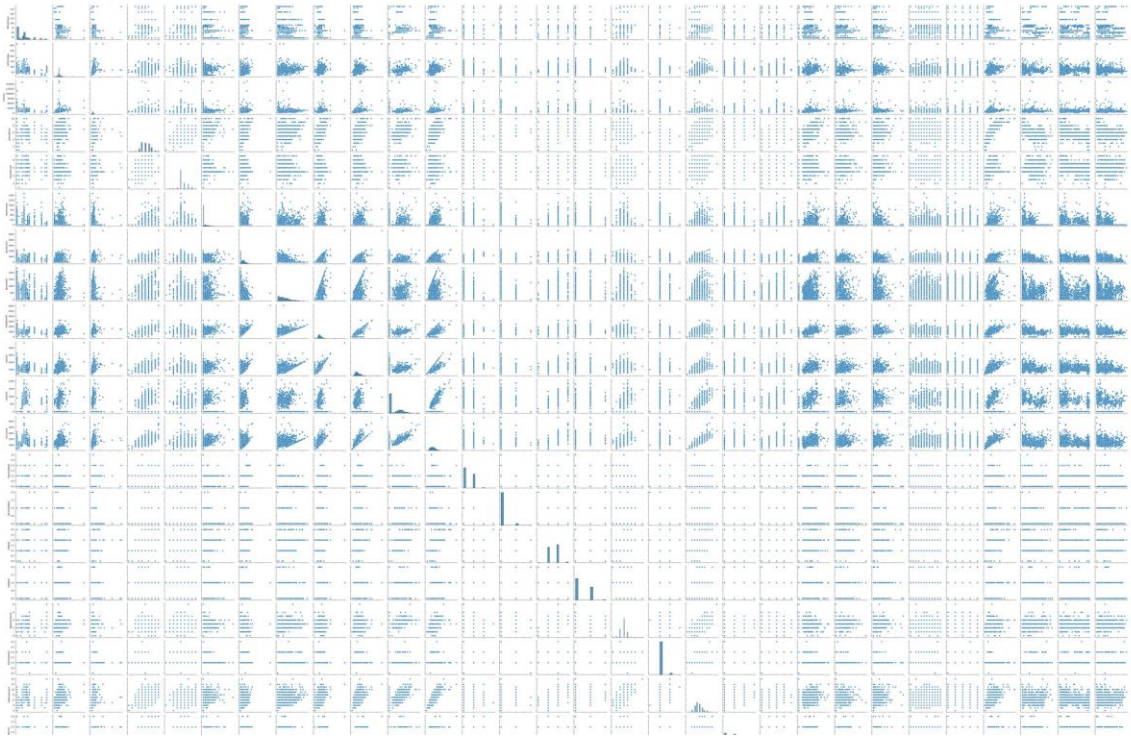
Discrete variables Vs SalePrice

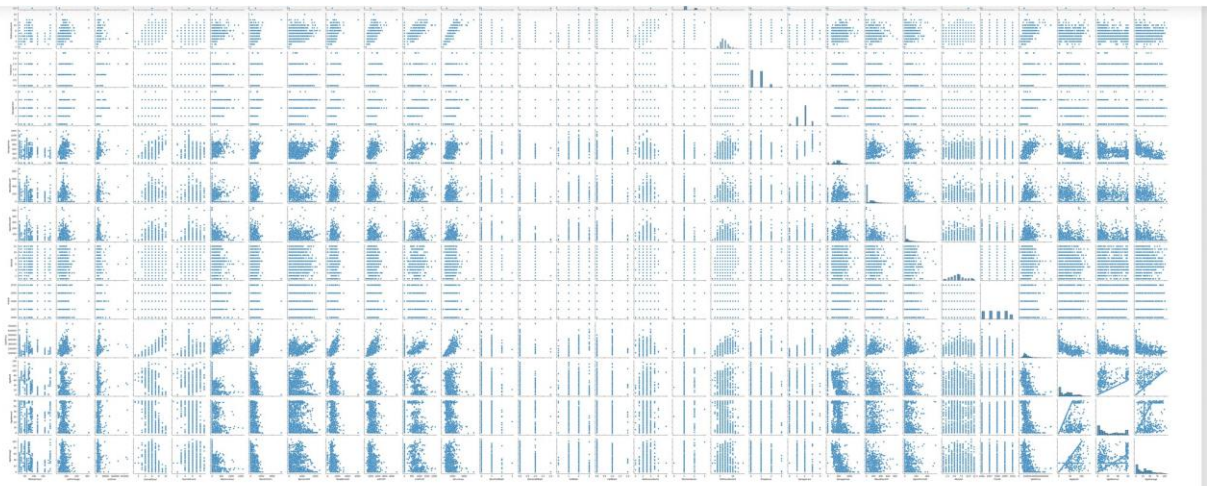




14. Multivariate Analysis

```
In [44]: 1 sns.pairplot(train_df, palette = 'husl')
Out[44]: <seaborn.axisgrid.PairGrid at 0x7fe5da1f0c70>
```





1) This pairplot gives the pairwise relation betweeen the columns which is plotted on the basis of target variable "SalePrice". Here we can observe the relation between the features and the label.

2) Here we can observe the correlation between the features and on the diagonal we can notice the distribution plot which shows whether the columns has skewness or not.

3) We can also see the linear relation between the features.

- ***Observations:***

- The range of the target value is 100000 to 500000
  and frequency of the sale price is between 100000 to 200000
  - Houses sold in the year 2007 were highest and lowest in the year 2010 which seems practically not possible.
  - Houses built in the year 1990 were sold at high prices.
  - On observing on the basis of different parameters :
  - Houses with Ex Basement Quality, PCond Foundation, Ex kitchen quality, 2nd story is expensive.
  - Excellent categories are of high prices.

  - ***Interpretation of the Results***

  - All the independent columns (categorical and numerical) are imbalanced. In categorical columns, only one or two categories are of high frequency, and in numerical columns, zeros shrink the graphs.
  - Categories that have excellent category show high prices.
  - Looking at the temporal columns shows impractical records saying the prices of the houses reduced with the year sold.
  - As the number of independent columns is more it is difficult to analyse each column. Also, the number of rows is less as compared to independent columns.
  - Many columns have null values which I have treated using two techniques:
    creating new categories for more null values and replacing them with mode for less null values columns.
  - For encoding the categorical variables I used an ordinal encoder.
    Also, the columns have outliers and skewness reason being the more number of zeros in numerical features for which I have used power transform.
  - The full bath is highly correlated with the sale price and the year sold is least correlated.
  - Following are the values of r2 scores for each algorithm:
    1. Linear Regression:
       r2 score: 0.84
    2. Lasso Regressor:
       r2 score: 0.84
    3. Ridge Regressor:
       r2 score: 0.84
    4. Random Forest Regressor:
       r2 score: 0.89
    5. Extra Trees Regressor:

r2 score: 0.90

6. Gradient Boost Regressor :
   r2 score: 0.91
7. Bagging Regressor
   r2 score: 0.87
8. Grid Search CV:
   r2 score: 0.89

CONCLUSION

- ***Key Findings and Conclusions of the Study***

All the columns are imbalanced and the major drawback of the dataset is the number of columns.
The key challenges of the problem were handling null values and skewed data .Also, some categorical columns have many categories from which we can not conclude results because of imbalance and null values. For numerical columns, more zeros are there.

- ***Learning Outcomes of the Study in respect of Data Science***

As the dataset contains 81 columns it is good practice to drop the column with no correlation with the target variable or with most null values, So I dropped PoolQC, misFeatures, Utilities, and Id.
Because of 81 columns, I have tried two techniques for feature selection to eliminate the non-significant features.
Some standard classification algorithms I have used are: Linear Regression, Support Vector regressor, Decision Tree Regressor, Kneigbors Regressor, and some ensemble techniques I have used are Random Forest Regressor and Ada Boost Regressor, Gradient Boost Regressor
I have also used a cross-validation score to validate the overfitting of the model.
EDA is time consuming as we need to do analysis between numerical and Numerical variable, categorical and numerical column and also the number
Of columns are also a big factor.
Predicting target variable after feature selection 2% score is less, therefore, I have not used that model because of the number of rows

If for 1163 rows only it is affecting the accuracy for more rows it may not give a precise prediction.

- ***Limitations of this work and Scope for Future Work***

If the data can be balanced and all the values were there the model can easily predict the data. The number of dimensions is one of the limitations of the dataset. Also treating null values affect the precision of the model as many columns have null values. EDA of this type of data took time because of the number of dimensions. If fewer null values could be there and it will be good for predicting the Sales Price.