FLIP ROBO

HOUSING: Pricing Prediction

Submitted by:

NITIN SINGH TATRARI

Batch no. : 1825

ACKNOWLEDGMENT

I have studied on various social-economical and technical factors on predicting the house price. I have read articles online on house prediction. I have also studied different models from Kaggle on House prediction model. I have understood each variables and it importance in a house.

INTRODUCTION

- Business Problem Framing

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 are 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.

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. Thus we 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.

- Conceptual Background of the Domain Problem

The actual price of housing properties depends on many variables. We want to find which variables are important for price prediction and how these variables describe the price of a house.

Generally, House prices depends on many factors like location and its proximity to different utilities, house size and usable space, Age and condition of property and upgrades like garage, swimming pool, etc added to the properties.

Some of the socio-economical factors also affect the pricing like economic growth which indicates income, unemployment rates, availability of properties (supply), loans and interest rates.

In this project, we will not consider the socio-economical factors.

- Review of Literature

  I have studied on various social-economical and technical factors on predicting the house price.

  Some of the socio-economical factors also affect the pricing like economic growth which indicates income, unemployment rates, availability of properties (supply), loans and interest rates.

  House prices depends on technical factors like location and its proximity to different utilities, house size and usable space, Age and condition of property and upgrades like garage, swimming pool, etc added to the properties.

- Motivation for the Problem Undertaken

  Own House is something which an individual always dreams to have at some point in his life. So it is one of the evergreen industries. The challenge is faced to predict the correct price of a house. Thus, by finding out all important parameters upon which pricing depends, we can make the machine learn it and apply the model to predict the price of the property. It will help in purchasing houses at a price below their actual values and flip them at a higher price everywhere. That’s why I decided to conduct my project around this topic with Machine Learning.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

1) Data is normalised by removing outlier in it by applying z-score function. The data is keep between -3 standard deviation to +3 standard deviation of the curve. Z-score is applied on continuous variables only. If we try to normalise categorical variables, then it could change the meaning of the data.

```
from scipy.stats import zscore
z=np.abs(zscore(df[{'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
       'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
       'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
       'BsmtFullBath', 'BsmtHalfBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars',
       'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'MiscVal','SalePrice'}]))
new_df=df[(z<3).all(axis=1)]
```

2) The correlation of both continuous and categorical variables was studied with respect to Target variable. We have kept all variables having correlations more than 15% with the target and remove the rest.

3) We have changed all values in categorical variables into numerical as show below:-

```
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
```

```
df2=df2.apply(le.fit_transform)
```

4) We have standardised the independent variables as show below:-

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

```
x=sc.fit_transform(x)
```

- ## Data Sources and their formats

The data was provided in csv format file. We upload it using pandas library and store it in 'data' as shown below:-

```
data=pd.read_csv('housing_train.csv')
data
```

The data information is as shown below:-

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1168 non-null   int64  
 1   MSSubClass     1168 non-null   int64  
 2   MSZoning       1168 non-null   object 
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64  
 5   Street         1168 non-null   object 
 6   Alley          77 non-null     object 
 7   LotShape       1168 non-null   object 
 8   LandContour    1168 non-null   object 
 9   Utilities      1168 non-null   object 
 10  LotConfig      1168 non-null   object 
 11  LandSlope      1168 non-null   object 
 12  Neighborhood   1168 non-null   object 
 13  Condition1     1168 non-null   object 
 14  Condition2     1168 non-null   object 
 15  BldgType       1168 non-null   object 
 16  HouseStyle     1168 non-null   object 
 17  OverallQual    1168 non-null   int64  
 18  OverallCond    1168 non-null   int64  
 19  YearBuilt      1168 non-null   int64  
 20  YearRemodAdd   1168 non-null   int64  
 21  RoofStyle      1168 non-null   object 
 22  RoofMatl       1168 non-null   object 
 23  Exterior1st    1168 non-null   object 
 24  Exterior2nd    1168 non-null   object 
 25  MasVnrType     1161 non-null   object 
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object 
 28  ExterCond      1168 non-null   object 
 29  Foundation     1168 non-null   object 
 30  BsmtQual       1138 non-null   object 
 31  BsmtCond       1138 non-null   object 
 32  BsmtExposure   1137 non-null   object 
 33  BsmtFinType1   1138 non-null   object 
 34  BsmtFinSF1     1168 non-null   int64  
 35  BsmtFinType2   1137 non-null   object 
 36  BsmtFinSF2     1168 non-null   int64  
 37  BsmtUnfSF      1168 non-null   int64  
 38  TotalBsmtSF    1168 non-null   int64  
 39  Heating        1168 non-null   object 
 40  HeatingQC      1168 non-null   object 
 41  CentralAir     1168 non-null   object 
 42  Electrical     1168 non-null   object 
 43  1stFlrSF       1168 non-null   int64  
 44  2ndFlrSF       1168 non-null   int64  
 45  LowQualFinSF   1168 non-null   int64  

 46  GrLivArea      1168 non-null   int64  
 47  BsmtFullBath   1168 non-null   int64  
 48  BsmtHalfBath   1168 non-null   int64  
 49  FullBath       1168 non-null   int64  
 50  HalfBath       1168 non-null   int64  
 51  BedroomAbvGr   1168 non-null   int64  
 52  KitchenAbvGr   1168 non-null   int64  
 53  KitchenQual    1168 non-null   object 
 54  TotRmsAbvGrd   1168 non-null   int64  
 55  Functional     1168 non-null   object 
 56  Fireplaces     1168 non-null   int64  
 57  FireplaceQu    617 non-null    object 
 58  GarageType     1104 non-null   object 
 59  GarageYrBlt    1104 non-null   float64
 60  GarageFinish   1104 non-null   object 
 61  GarageCars     1168 non-null   int64  
 62  GarageArea     1168 non-null   int64  
 63  GarageQual     1104 non-null   object 
 64  GarageCond     1104 non-null   object 
 65  PavedDrive     1168 non-null   object 
 66  WoodDeckSF     1168 non-null   int64  
 67  OpenPorchSF    1168 non-null   int64  
 68  EnclosedPorch  1168 non-null   int64  
 69  3SsnPorch      1168 non-null   int64  
 70  ScreenPorch    1168 non-null   int64  
 71  PoolArea       1168 non-null   int64  
 72  PoolQC         7 non-null      object 
 73  Fence          237 non-null    object 
 74  MiscFeature    44 non-null     object 
 75  MiscVal        1168 non-null   int64  
 76  MoSold         1168 non-null   int64  
 77  YrSold         1168 non-null   int64  
 78  SaleType       1168 non-null   object 
 79  SaleCondition  1168 non-null   object 
 80  SalePrice      1168 non-null   int64  
dtypes: float64(3), int64(35), object(43)
memory usage: 739.2+ KB
```

```
df=pd.DataFrame(data)
```

1) There are 1168 entries and 81 columns in the dataset.
2) The data type comprises of 35 integer type columns, 3 float type columns and 43 string type columns.
3) Columns- LotFrontage, Alley, MasVnrType, MasVnrArea, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2 have Nan values.

Then the data is loaded into data Frame 'df'

```
df=pd.DataFrame(data)
```

- Data Pre-processing Done

1) We try to find all unique values and its count in the variables.

```
for i in df.columns:
    print('\n')
    print(i)
    print(df[i].unique())
    print('The Total no. of unique value:',len(df[i].unique()))
    print(df[i].value_counts())
```

```
Id
[127 889 793 ... 196  31 617]
The Total no. of unique value: 1168
1460    1
501     1
476     1
477     1
478     1
       ..
959     1
961     1
962     1
963     1
1       1
Name: Id, Length: 1168, dtype: int64
```

```
Street
['Pave' 'Grvl']
The Total no. of unique value: 2
Pave    1164
Grvl       4
Name: Street, dtype: int64
```

```
Alley
[nan 'Grvl' 'Pave']
The Total no. of unique value: 3
Grvl    41
Pave    36
Name: Alley, dtype: int64
```

```
Utilities
['AllPub']
The Total no. of unique value: 1
AllPub    1168
Name: Utilities, dtype: int64
```

```
PoolArea
[  0 555 576 738 519 480 648 512]
The Total no. of unique value: 8
0      1161
738       1
648       1
576       1
555       1
519       1
512       1
480       1
Name: PoolArea, dtype: int64
```

```
Fence
[nan 'MnPrv' 'GdPrv' 'GdWo' 'MnWw']
The Total no. of unique value: 5
MnPrv    129
GdPrv     51
GdWo      47
MnWw      10
Name: Fence, dtype: int64
```

```
PoolQC
[nan 'Ex' 'Gd' 'Fa']
The Total no. of unique value: 4
Gd    3
Ex    2
Fa    2
Name: PoolQC, dtype: int64
```

```
MiscFeature
[nan 'Shed' 'Gar2' 'TenC' 'Othr']
The Total no. of unique value: 5
Shed    40
Gar2     2
Othr     1
TenC     1
Name: MiscFeature, dtype: int64
```

a) All rows in column- ID have different values, thus it show no relevance to house price.
b) Since only 4 houses have gravel road to access out of 1168, we will only consider paved road.
c) Since column- Alley has 1091 missing data out of 1168, it provides no help in house price prediction.
d) All rows in column- Utilities have same value 'AllPub'. Thus we consider all houses having all public utilities.
e) Since column – Fence has 931 missing value out of 1168, thus can not to develop any relation with target.
f) Since column – MiscFeature has 1124 missing value out of 1168, thus can not to develop any relation with target.
g) Only 7 houses out of 1168 have swimming pool, we should avoid making relation with the target variable
h) Since no swimming is considered, PoolQC (pool quality) can be ignored too.

Since above columns cannot be treated, we have to drop them.

```
df.drop({'PoolQC','PoolArea','Alley','MiscFeature','Fence','Utilities','Id','Street'},axis=1,inplace=True)
df.head()
```

```
print(df['LotFrontage'].corr(df['LotArea']))
```

```
0.5572571226801905
```

Since columns 'LotFrontage' & 'LotArea' have above 0.55 correlation and columns 'LotFrontage' have 214 missing values(18.4%), we cannot drop 216 rows. Thus, we drop the column 'LotFrontage'.

```
df.drop('LotFrontage',axis=1,inplace=True)
```

I decide to drop column- LotArea as it have 214 missing value which is difficult to fill up. It also has 55% correlation with LotArea.

2) Repairing the data

```
df['FireplaceQu']=df['FireplaceQu'].mask(df['Fireplaces']==0,'NA')
```

```
df['FireplaceQu'].value_counts()
```

```
NA    551
Gd    301
TA    252
Fa     25
Ex     21
Po     18
Name: FireplaceQu, dtype: int64
```

In all those entries where Fireplaces=0, there will be no data for fire place quality. Thus we fill those places with 'NA 'in colum-FireplaceQU.

```
for i in ['BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2']:
    df[i]=df[i].mask(df['TotalBsmtSF']==0,'NA')
```

```
for i in ['BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2']:
    print(i,df[i].isnull().sum())
```

```
BsmtQual 0
BsmtCond 0
BsmtExposure 1
BsmtFinType1 0
BsmtFinType2 1
```

Since n all those entries where TotalBsmtSF=0, there will be no data in columns related to basement, thus we fill those places with 'NA'.

```
for i in ['GarageType','GarageFinish','GarageQual','GarageCond','GarageYrBlt']:
    df[i]=df[i].mask(df['GarageArea']==0,'NA')
```

```
for i in ['GarageType','GarageFinish','GarageQual','GarageCond','GarageYrBlt']:
    print(i,df[i].isnull().sum())
```

```
GarageType 0
GarageFinish 0
GarageQual 0
GarageCond 0
GarageYrBlt 0
```

```
df['GarageYrBlt']=df['GarageYrBlt'].replace('NA',2006.0)
```

Since in all those entries where TotalBsmtSF=0, there will be no data in columns related to basement, thus we fill those places with 'NA'.

```
df['MasVnrType']=df['MasVnrType'].mask(df['MasVnrArea']==0,'None'
print('MasVnrType','\n',df['MasVnrType'].value_counts())
print('No. of Nan value:',df['MasVnrType'].isnull().sum())
```

```
MasVnrType
 None        697
BrkFace      353
Stone         98
BrkCmn        13
Name: MasVnrType, dtype: int64
No. of Nan value: 7
```

Since in all those entries where MasVnrArea =0, MasVnrType will be 'None ', thus we fill those places with 'None'.

```
df=df.dropna()
```

```
df.shape
```

```
(1159, 72)
```

We drop any other rows we contains nan values.

The new shape of Data frame 'df' is (1159, 72).

3) Removing outlier

We apply zscore function on numerical columns

```
from scipy.stats import zscore
z=np.abs(zscore(df[{'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
       'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
       'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
       'BsmtFullBath', 'BsmtHalfBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars',
       'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'MiscVal','SalePrice'}]))
new_df=df[(z<3).all(axis=1)]
```

New_df shape is (809,72)

We again try to find all unique values and its count in the variables.

```
for i in new_df.columns:
    print('\n')
    print(i)
    print(new_df[i].unique())
    print('The Total no. of unique value:',len(new_df[i].unique()))
    print(new_df[i].value_counts())
```

```
Condition2
['Norm' 'Feedr']
The Total no. of unique value: 2
Norm     805
Feedr      4
Name: Condition2, dtype: int64
```

```
RoofMatl
['CompShg' 'WdShngl' 'Tar&Grv' 'WdShake']
The Total no. of unique value: 4
CompShg    805
Tar&Grv      2
WdShake      1
WdShngl      1
Name: RoofMatl, dtype: int64
```

```
Heating
['GasA' 'Floor' 'GasW' 'Wall' 'Grav']
The Total no. of unique value: 5
GasA     798
Grav       5
GasW       4
Wall       1
Floor      1
Name: Heating, dtype: int64
```

a) Out of 809 entries column- Condition2 has 805 entries of norm(Normal proximity to various condition). Thus, we can consider all houses to have normal proximity.
b) Out of 809 entries column-RoofMat1 has 805 entries of CompShg(Standard (Composite) Shingle). Thus, we can consider all houses to Standard (Composite) Shingle.
c) Out of 809 entries column- Heating has 798 of GasA (Gas forced warm air furnace). Thus, we can consider all houses to have GasA heating system.

4) Label Encoder

We have converted the data from string to numerical in all string data type columns.

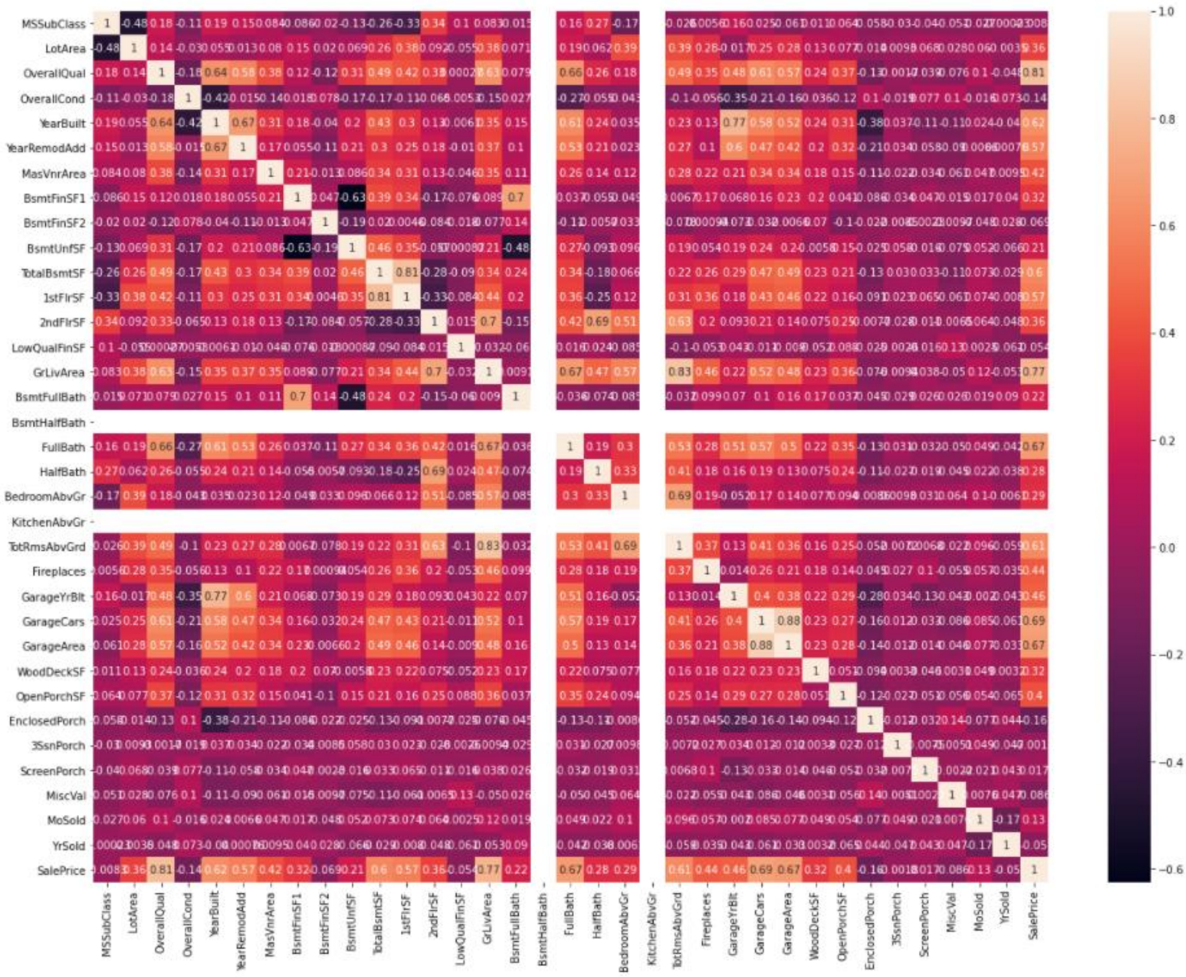
```
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
```

```
df2=df2.apply(le.fit_transform)
```

- Data Inputs- Logic- Output Relationships

1) Correlation of continuous type variables with Target variables

```
plt.figure(figsize=(20,15))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



a) Columns :- 'MSSubClass','OverallCond','BsmtFinSF2','LowQualFinSF','BsmtHalfBath','KitchenAbvGr','EnclosedPorch','3SsnPorch','ScreenPorch','MiscVal','MoSold','YrSold','GarageCars' have correlation less than 15% with the Target variable –'SalePrice'.
b) Columns:- 'GarageCars' & 'GarageArea' have high correlation of 88% with each other, thus we can drop one of them.

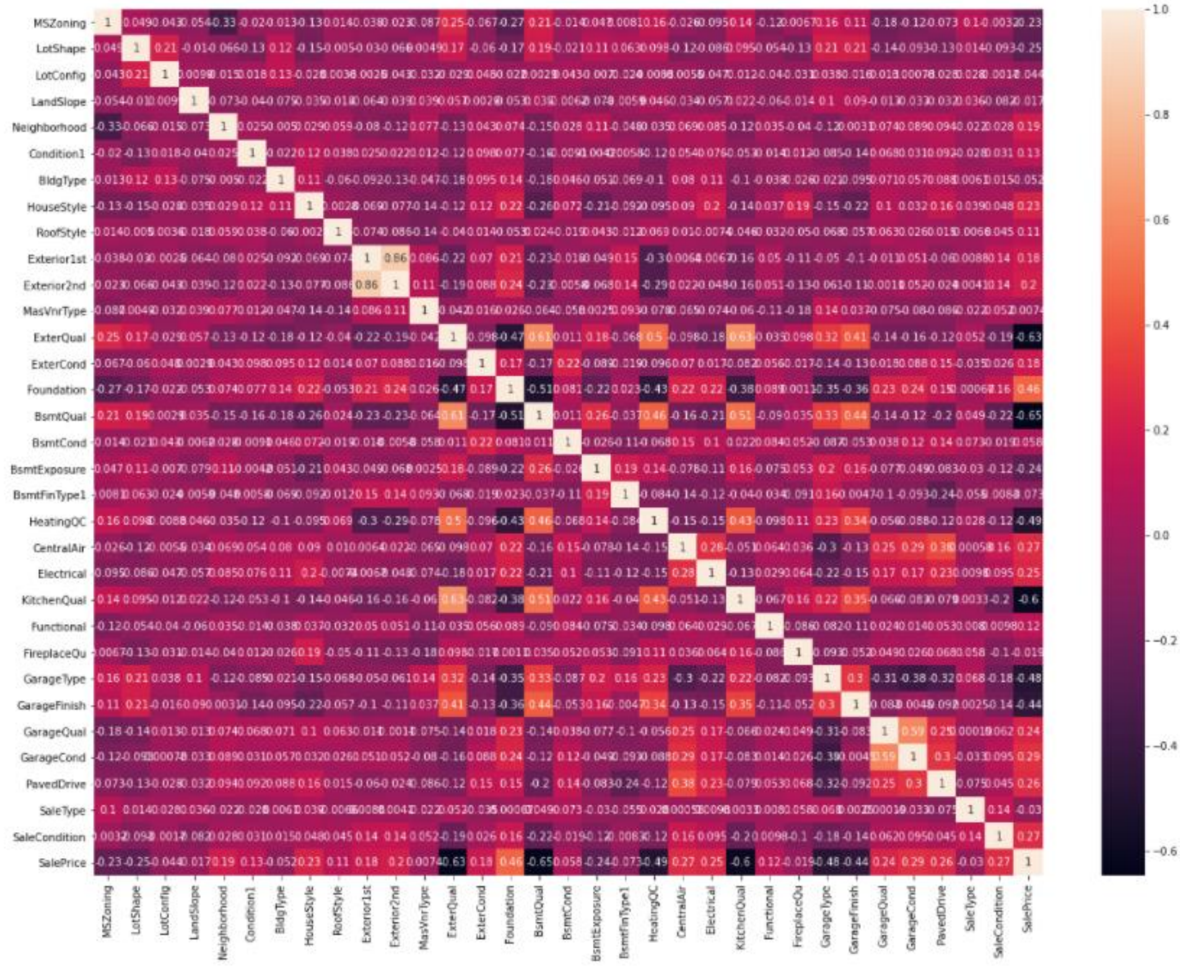
Thus we drop the columns.

```
df.drop({'MSSubClass','OverallCond','BsmtFinSF2','LowQualFinSF','BsmtHalfBath','KitchenAbvGr','EnclosedPorch','3SsnPorch',
        'ScreenPorch','MiscVal','MoSold','YrSold','GarageCars'},axis=1,inplace=True)
df.head()
```

2) Correlation of continuous type variables with Target variables

```
df2=df2.join(df['SalePrice'])
```

```
plt.figure(figsize=(20,15))
sns.heatmap(df2.corr(),annot=True)
plt.show()
```



a) Columns- 'LotConfig','Condition1','LandSlope','BldgType','MasVnrType', 'BsmtCond','BsmtFinType1','FireplaceQu' 'SaleType', 'Exterior1st', 'RoofStyle', 'SalePrice' have have correlation less than 15% with the Target variable –'SalePrice'.

Thus we drop the columns

```
df2.drop({'LotConfig','Condition1','LandSlope','BldgType','MasVnrType','BsmtCond','BsmtFinType1','FireplaceQu',
          'SaleType','Exterior1st','RoofStyle','SalePrice'},axis=1,inplace=True)
df2.head()
```

	MSZoning	LotShape	Neighborhood	HouseStyle	Exterior2nd	ExterQual	ExterCond	Foundation	BsmtQual	BsmtExposure	...	CentralAir	Electrical	Kitche
0	3	0	13	2	8	3	2	1	2	4	...	1	2	
2	3	0	15	4	6	2	2	2	2	0	...	1	2	
3	3	0	14	2	8	3	2	1	2	4	...	1	2	
5	3	0	8	4	11	2	2	2	2	0	...	1	2	
6	3	0	19	2	12	3	2	1	2	4	...	1	2	

5 rows × 21 columns

- ## State the set of assumptions (if any) related to the problem under consideration

The assumptions considered:-

1) The access to all houses is by gravel road
2) Houses have no swimming pool.
3) All houses have public utilities- Electricity, Gas, Water and Septic Tank.
4) Access to Alley, Fence and other miscellaneous feature – Elevators, Tennis court, 2nd garage, and shade are not considered to predict the House sale price.
5) All houses to have normal proximity.
6) All houses have roofs made with Standard (Composite) Shingle.
7) YearBuilt (Original construction date), YearRemodAdd (Remodel date) and GarageYrBlt (Year garage was built) are all consider continuous value.
8) Heating system in Houses is considered with Gas forced warm air furnace only.
9) Roofing material used in all houses in considered Standard (Composite) Shingle.

- ## Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

The libraries used are: pandas, numpy, matplotlib.pyplot, seaborn and scikit_learn and SciPy. The laptop used is with Intel I5 10th generation, 4GB RAM, 4GB GPU.

Model/s Development and Evaluation

- ## Identification & Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

The algorithms used for training and testing are

LinearRegression(), DecisionTreeRegressor(),

Space VectorRegressor() ,Lasso() & Ridge().

```
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import Lasso,Ridge
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

- ## Run and Evaluate selected models & Key Metrics used

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

A) MODEL SELECTION

1. Linear Regression

```
lg=LinearRegression()
lg.fit(x_train,y_train)
pred=lg.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: 0.8713028389025775
mean_squared_error: 460713430.03505886
mean_absolute_error: 15333.327115075554
Root_mean_square_error: 21464.2360692166
```

```
lg.score(x_train,y_train)
```

```
0.9045905865184293
```

```
lg.score(x_test,y_test)
```

```
0.8713028389025775
```

2. Decision Tree Regressor()

```
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
pred=dtr.predict(x_test)
print("R2_score:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print('Root_mean_square_error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_score: 0.6722824933102731
mean_squared_error: 1173171617.0123458
mean_absolute_error: 24694.777777777777
Root_mean_square_error: 34251.59291204346
```

3. Space Vector Regressor()

```
svr=SVR()
svr.fit(x_train,y_train)
pred=svr.predict(x_test)
print("R2_scorez:",r2_score(y_test,pred))
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
```

```
R2_scorez: -0.03634764616808339
mean_squared_error: 3709944140.985561
mean_absolute_error: 47338.12966528457
```

Linear Regression() shows the best score.

4. Lasso & Ridge Regressor()

```
ls=Lasso(alpha=0.0001)
ls.fit(x_train,y_train)
```

```
Lasso(alpha=0.0001)
```

```
ls.score(x_train,y_train)
```

```
0.9045757405345324
```

```
ls.score(x_test,y_test)
```

```
0.8713166376771689
```

```
Rg=Ridge(alpha=0.0001)
Rg.fit(x_train,y_train)
```

```
Ridge(alpha=0.0001)
```

```
Rg.score(x_train,y_train)
```

```
0.9045905863937509
```

```
Rg.score(x_test,y_test)
```

```
0.8713029760942608
```

Since Linear Regression() and Lasso score are same, that means already optimum variables where chosen for modelling. So no regularisation was needed.

B) CROSS VALIDATION

```
cross=cross_val_score(lg,x,y,cv=5)
print('lg')
print('Score:',cross)
print('Mean_score:',cross.mean())
print('STD_score:',cross.std())
```

```
lg
Score: [0.88879978 0.91111744 0.87370369 0.89931484 0.84434136]
Mean_score: 0.8834554237662783
STD_score: 0.023109638319315746
```

```
cross=cross_val_score(dtr,x,y,cv=5)
print('dtr')
print('Score:',cross)
print('Mean_score:',cross.mean())
print('STD_score:',cross.std())
```

```
dtr
Score: [0.7447703  0.77881238 0.77652313 0.78537185 0.70802787]
Mean_score: 0.758701108445698
STD_score: 0.02896639699640543
```

The variance is in limit.

C) HYPERPARAMETER

```
from sklearn.model_selection import GridSearchCV
```

```
alphavalue={'alpha':[1,0.1,0.01,0.001,0.001,0.0001,0]}
grid=GridSearchCV(estimator=Lasso(),param_grid=alphavalue)
grid.fit(x,y)
print(grid)
print(grid.best_score_)
print(grid.best_estimator_.alpha)
print(grid.best_params_)
```

```
GridSearchCV(estimator=Lasso(),
             param_grid={'alpha': [1, 0.1, 0.01, 0.001, 0.001, 0.0001, 0]})
0.8835706117906955
1
{'alpha': 1}
```

```
las=Lasso(alpha=1)
las.fit(x_train,y_train)
```

```
Lasso(alpha=1)
```

```
cross=cross_val_score(las,x,y,cv=5)
print('lasso')
print('Score:',cross)
print('Mean_score:',cross.mean())
print('STD_score:',cross.std())
```

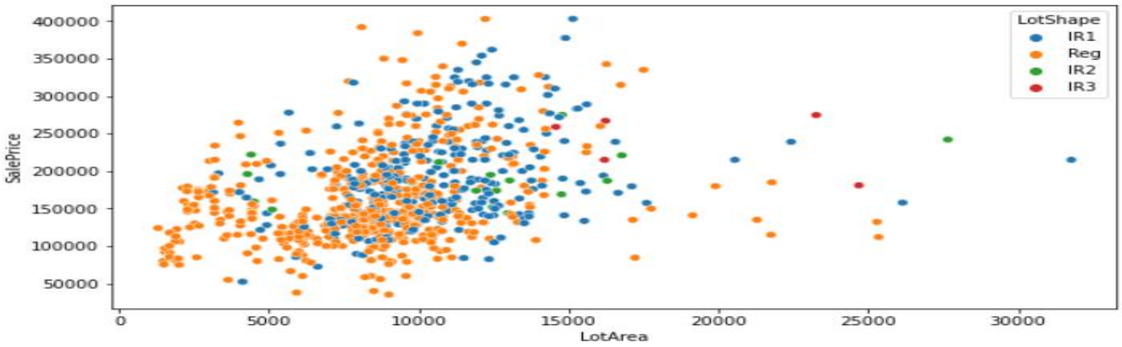
```
lasso
Score: [0.88909972 0.91108469 0.87367468 0.89972734 0.84426663]
Mean_score: 0.8835706117906955
STD_score: 0.023200725782853756
```

The best alpha value is 1, so we use it in Lasso regression and get score of 88.357%

- # Visualizations

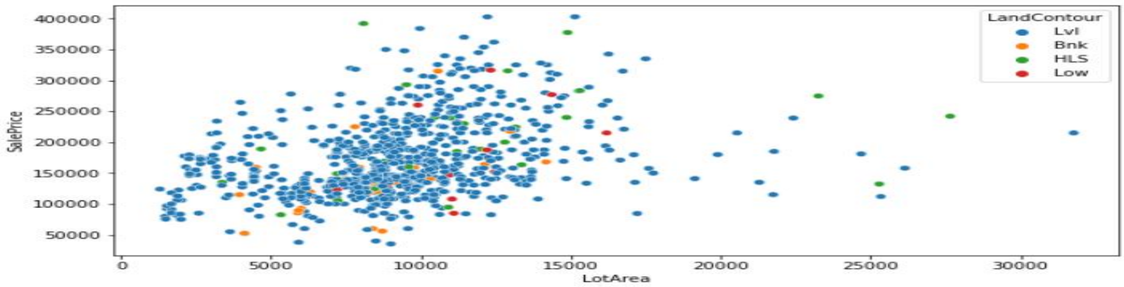
Plotting Target variable ‘SalePrice’ with different variables

```
plt.figure(figsize=(10,5))
sns.scatterplot(data=df,x='LotArea',y='SalePrice',hue='LotShape')
plt.show()
```



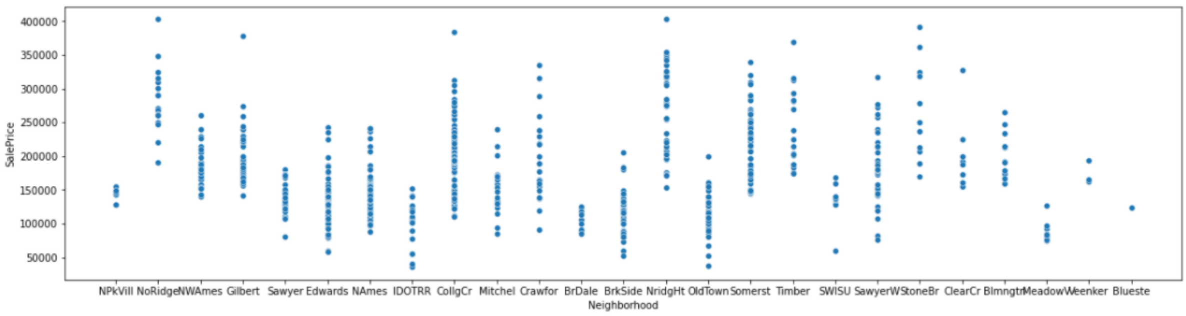
a) Sale price show a positive linear relationship with Lot area.
b) Regular lot shape is seen more in less than 7000sqtmr Lot area.
c) Slightly irregular lot is seen more in above 7000sqmtr Lot area.

```
plt.figure(figsize=(10,5))
sns.scatterplot(data=df,x='LotArea',y='SalePrice',hue='LandContour')
plt.show()
```



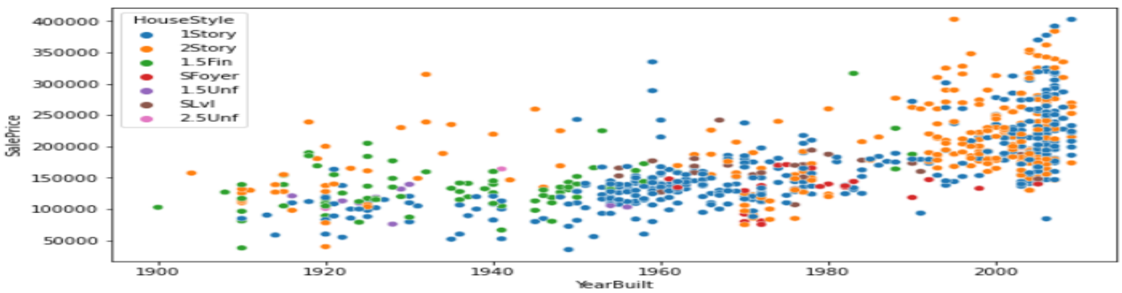
a) Almost all plots are near flat. LandContour have less correlation with Sales Price.

```
plt.figure(figsize=(20,5))
sns.scatterplot(data=df,x='Neighborhood',y='SalePrice')
plt.show()
```



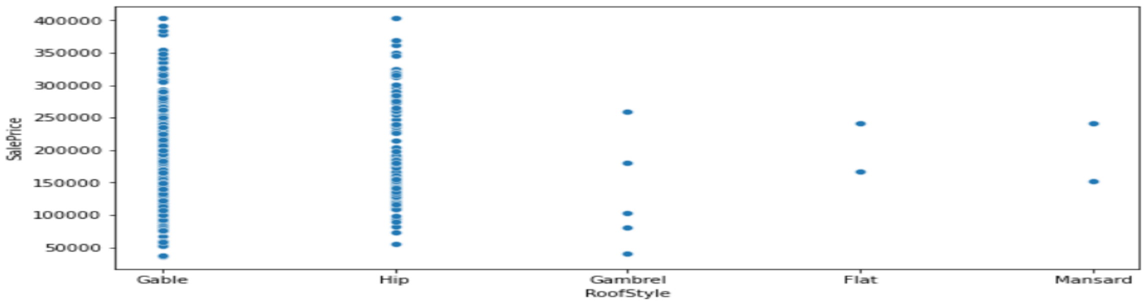
a) Sale price varies with Neighbourhoods.

```
plt.figure(figsize=(10,5))
sns.scatterplot(data=df,x='YearBuilt',y='SalePrice',hue='HouseStyle')
plt.show()
```



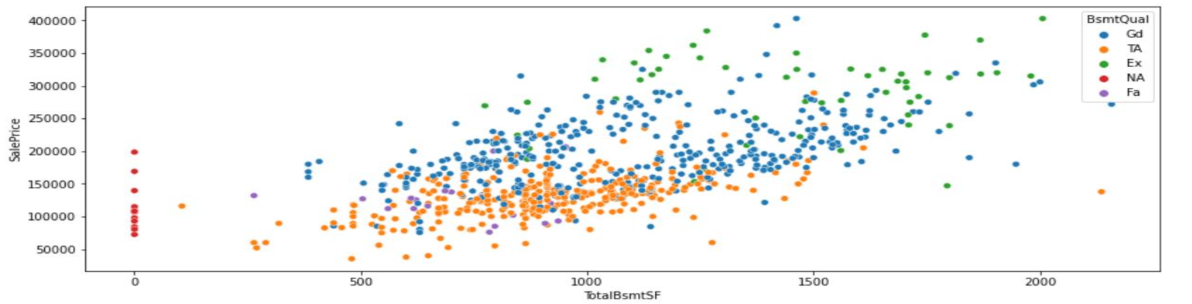
a) Newer constructions are costlier than older construction.
b) Newer constructions are mainly of one storey and two storey

```
plt.figure(figsize=(10,5))
sns.scatterplot(data=df,x='RoofStyle',y='SalePrice')
plt.show()
```



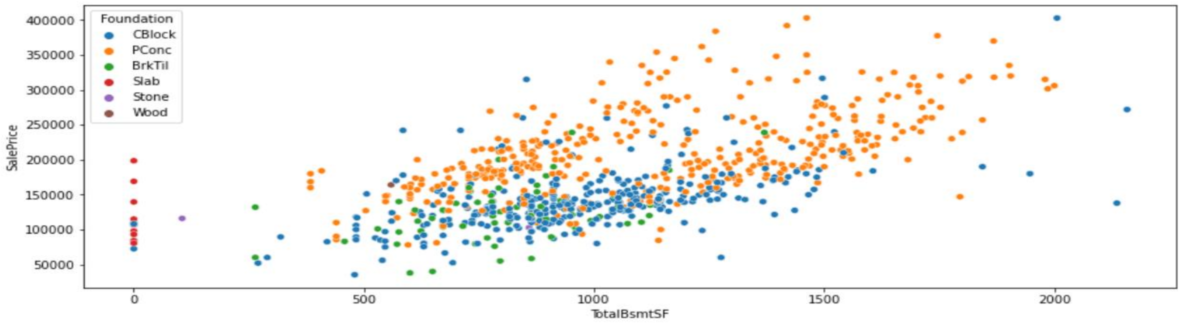
a) Gable & Hip type of roof is found in houses of all price range, but houses which are sold for more than 300000, only have either Gable or Hip type of roof.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='TotalBsmtSF',y='SalePrice',hue='BsmtQual')
plt.show()
```



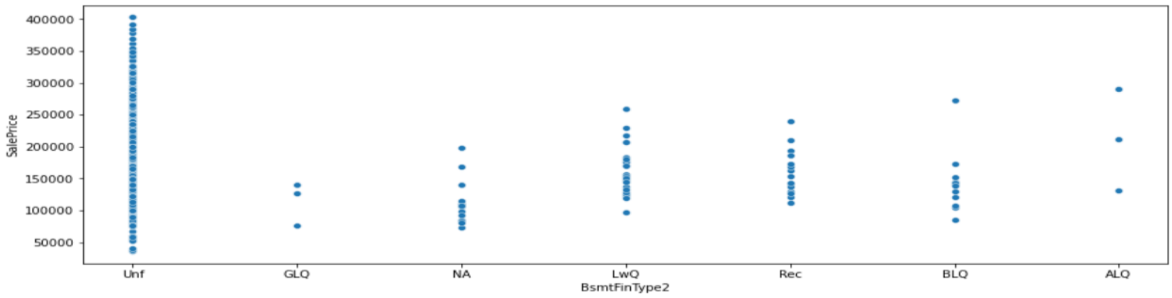
a) Sale price of house is proportional to total basement area.
b) Sale price of house is proportional to elevated height of basement.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='TotalBsmtSF',y='SalePrice',hue='Foundation')
plt.show()
```



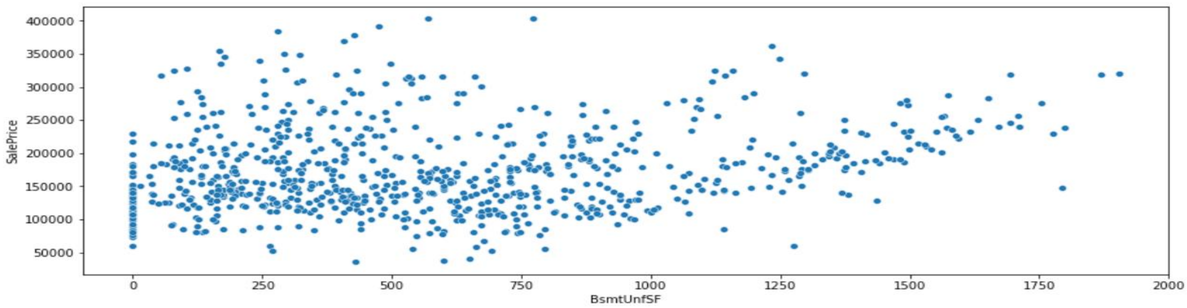
a) Basement Foundation made with poured concrete are found in costlier house, followed by basement foundation made with cinder block.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='BsmtFinType2',y='SalePrice')
plt.show()
```



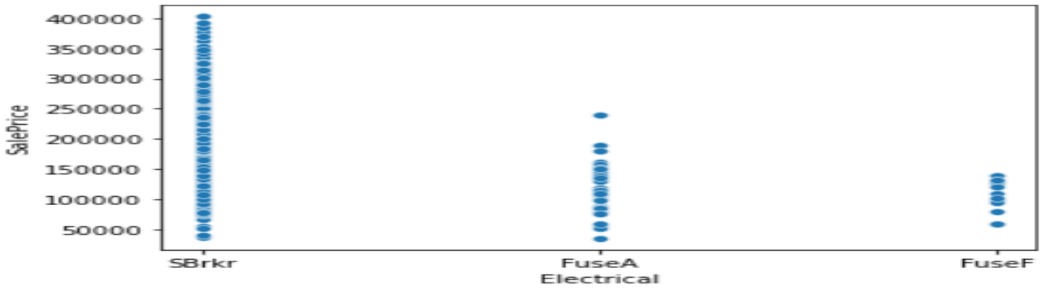
a) Houses having unfinished type 2 basement finish does not contribute to sale price prediction.
b) Since we have dropped Basement type 2 areas earlier, we can also drop type 2 of basement finish.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='BsmtUnfSF',y='SalePrice')
plt.show()
```



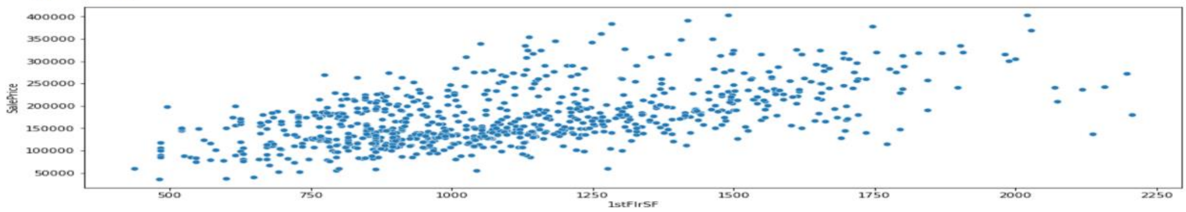
a) Unfinished basement area has a slightly linear relation with Sale price.

```
sns.scatterplot(data=df,x='Electrical',y='SalePrice')
plt.show()
```



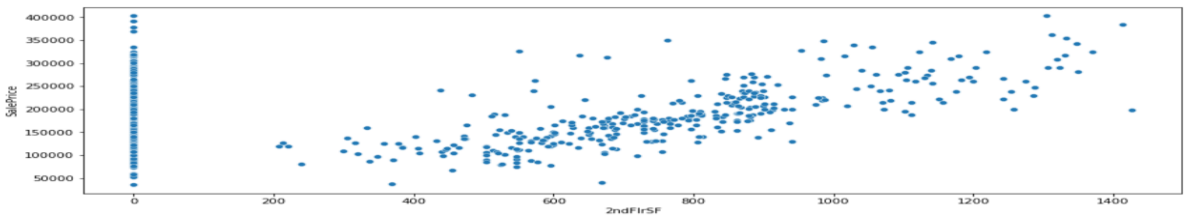
a) Houses with FuseF (60 AMP Fuse Box and mostly Romex wiring) electrical system have selling prices sell than 160000.
b) Houses with FuseA (60 AMP Fuse Box and all Romex wiring) electrical system have selling prices sell than 250000.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='1stFlrSF',y='SalePrice')
plt.show()
```



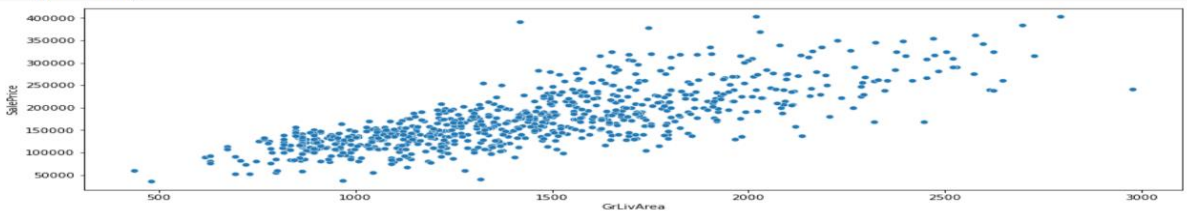
a) Sale price of house is proportional to first floor area.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='2ndFlrSF',y='SalePrice')
plt.show()
```



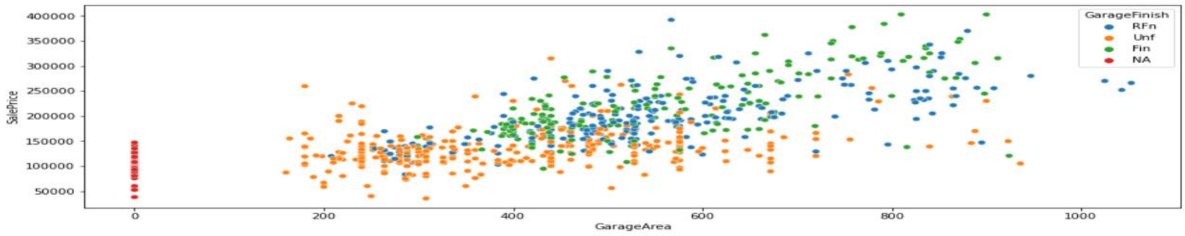
a) Sale price of house is proportional to second floor area.

```
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='GrLivArea',y='SalePrice')
plt.show()
```



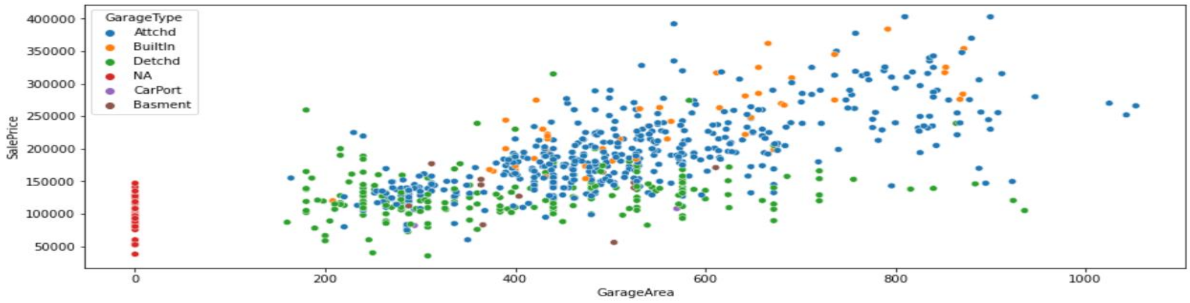
a) Sale price of house is proportional to Above grade (ground) living area

```python
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='GarageArea',y='SalePrice', hue='GarageFinish')
plt.show()
```



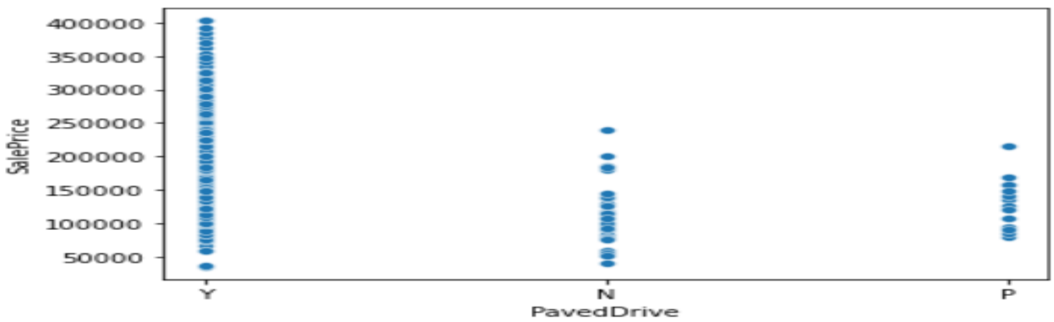
a) Sale price is proportional to Garage area.
b) Sale price of houses depends on Garage finish. (Finished> Rough Finished > Unfinished)

```python
plt.figure(figsize=(15,5))
sns.scatterplot(data=df,x='GarageArea',y='SalePrice', hue='GarageType')
plt.show()
```



a) Sale price of Houses is more with Attached and build-in garage than detached garage to the house.

```python
plt.figure(figsize=(6,4))
sns.scatterplot(data=df,x='PavedDrive',y='SalePrice')
plt.show()
```



a) All Houses sold above 250000, have paved drive way.

- Interpretation of the Results

All the interpretation is mentioned with the graphs ablove.

CONCLUSION

- Key Findings and Conclusions of the Study

We can summarise as follows. The Sale price depends on following parameters:-:-

1) The zone and the neighbourhood of the lot
2) The size and shape of lot.
3) The original construction date and remodelling date of house.
4) Area of masonry veneer, its quality, its present condition and whether there are two type of masonry veneer used.
5) Type of foundation of basement, its height, its total area, its unfinished area, its overall quality and exposure of basement.
6) Quality and condition of Heating system
7) Central air conditioning facility
8) Area of 1st floor, 2nd floor and above ground living area.
9) No. of full Bathroom in basement and above ground. No. of half bathroom above ground
10) Quality of Kitchen
11) Total no. of room above ground
12) Functionality of House
13) No. of Fire places in the house.
14) Garage area and its location, years on which it was build, garage quality and its present condition, and the interior finish of garage.
15) Whether driveway is paved.
16) Wood deck and open porch area.
17) Condition of Sale
18) Overall material and finish quality of House.

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

  It was good experience handling data with over 80 variables. I learn to choose the best parameter which affects the target variables. Filling up nan values by understanding correlated columns.
  Since most of the variables have linear relationship with the target variables, linear regression was the preferred Algorithm. I have consider Lasso regression as there were many variable, so it was important to regularize them. I have extensive studied the correlation of all variables with target variable and chosen 42 variable which have good correlation of above 15% with target variable. Thus, I don’t want to make the coefficient to absolute zero using ridge regression. Thus, I have chosen Lasso regression.

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

  1) Most of the categorical variable was imbalanced.
  2) Many variables were dropped due to missing values. Data should be complete.
  3) Many miscellaneous features like elevators, tennis court, etc were dropped due to its few numbers. The data should be well balanced. Thus, we would have considered more features for predicts, if data was complete and balanced.