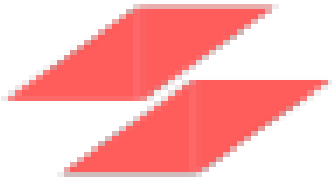


HOUSE PRICE PREDICTION

Submitted by:-

TANUJ SWARNKAR

ACKNOWLEDGMENT

References

1. “Selected Review of the Empirical Literature on House Price Modelling and Forecasting. What does the literature say? “, Zacharias Bragoudakis (Bank of Greece), Marina Emiris (Banque Nationale de Belgique), Mihnea Constantinescu (Bank of Lithuania), September 2016.
2. “House Price Developments in Europe: A Comparison “,Paul Hilbers, Alexander W. Hoffmaister,Angana Banerji, and Haiyan Shi
3. 3. Sean Holly M. Hashem Pesaran Takashi Yamagata, Forschungsinstitut zur Zukunft der Arbeit Institute for the Study of Labor September 2006.

INTRODUCTION

Business Problem Framing

To build a model to find the independent features which have impact on house price. This model will in turn 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. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Conceptual Background of the Domain Problem

The real estate market is one of the most competitive in terms of pricing and same tends to vary significantly based on numerous factors; forecasting property price is an important module in decision making for both the buyers and investors in supporting budget allocation, finding property finding stratagems and determining suitable policies hence it becomes one of the prime fields to apply the concepts of machine learning to optimize and predict the prices with high accuracy.

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia.

The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

Review of Literature

The paper" **Selected Review of the Empirical Literature on House Price Modelling and Forecasting**. **What does the literature say?"** presents a selected review of what the literature discusses in terms of modelling and forecasting house prices. In particular, it distils fundamental and "other" determinants of house prices used in economic models and identifies the most commonly used econometric approaches to estimate and forecast house prices.

This paper "House Price Developments in Europe: A Comparison "presents House prices in Europe have shown diverging trends, and this paper seeks to explain these differences by analyzing three groups of countries: the "fast lane", the average performers, and the slow movers.

Price movements in the first two groups are found to be driven mostly by income and trends in user costs, and housing markets in these countries seem relatively more susceptible to adverse developments in fundamentals.

This paper **"A Spatio-Temporal Model of House Prices in the US** "provides an empirical analysis of changes in real house prices in the USA using State level data. It examines the extent to which real house prices at the State level are driven by fundamentals such as real per capita disposable income, as well as by common shocks, and determines the speed of adjustment of real house prices to macroeconomic and local disturbances. We take explicit account of both cross- sectional dependence and heterogeneity. This allows us to find a counteracting relationship between real house prices and real per capita incomes with coefficients (1, −1), as predicted by the theory.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem

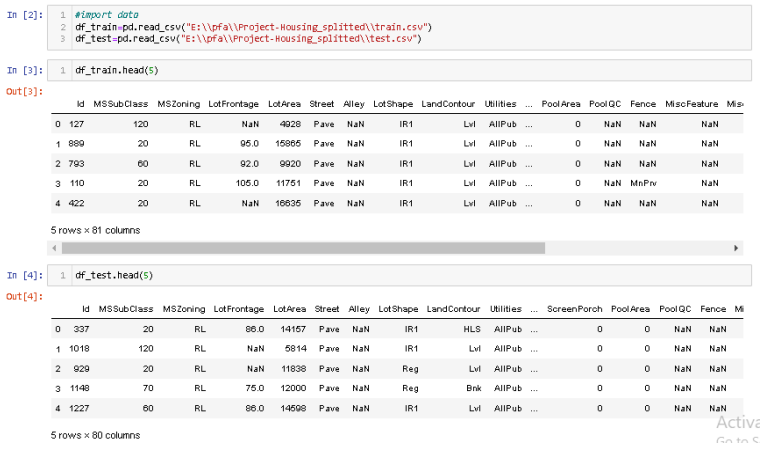
- Handled missing values in numerical data by mean and median.
- The feature distributed normally is replaced with mean others with median.
- Handled missing values in categorical data by mode
- All categorical features are analyzed using countplot
- All numerical features are analyzed using regplot.
- To find the correlation between each feature and target feature I have used correlation matrix
- Feature scaling is done using minmax scalar
- It is a regression problem so I have used 9 regression models

Data Sources and their formats

- This data set has train and test data set both are in csv format.
- The train dataset has 1168 rows and

81 columns The test dataset has 292 rows

and 80 columns



```
In [2]: 1 #import data
        2 df_train=pd.read_csv("E:\\pfa\\Project-Housing_splitted\\train.csv")
        3 df_test=pd.read_csv("E:\\pfa\\Project-Housing_splitted\\test.csv")

In [3]: 1 df_train.head(5)
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	Misc
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	

5 rows × 81 columns

```
In [4]: 1 df_test.head(5)
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	Mi
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	

5 rows × 80 columns

Data Pre-processing

- Importing libraries.
- Import the dataset(training and test dataset)
- Identifying and handling the missing values of categorical column by replacing the null value with mode value
- Handling the missing values of numerical column by replacing the null value with mean and median value
- PoolQc feature have only 0.5% of data, Fence have only 20.2% data and miscFeature have only 3.7% data. We cannot predict using these features so I'm going to drop all three features
- Encoding the categorical data to numerical data
- Feature scaling
- If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values so I have used minmax scalar to covert un scaled data to scaled data.

Data Inputs- Logic- Output Relationships

The target feature is „sales price" is predicted by various features of house like MSSubClass, MSZoning, LotFrontage, LotArea, Street,Alley, LotShape, LandContour, Utilities, LotConfig, LandSlope, Neighborhood, Condition1, Con dition2, BldgType,HouseStyle, OverallQual, OverallCond, YearBuilt, YearRemodAdd,'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType, 'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtEx posure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'Full Bath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType' 'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', ' PoolArea', 'PoolQC',

'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'Sale Type','SaleCondition.

MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible problem-solving approaches

It is a regression problem because the output variable „sale price" is a real or continuous value. To find the best model I have used the following models

- KNeighborsRegressor()
- SVR()
- DecisionTreeRegressor()
- LinearRegression()
- Lasso()
- Ridge()
- RandomForestRegressor()
- GradientBoostingRegressor()

Testing of Identified Approaches (Algorithms)

I have used r2 score to check the accuracy of each model and best fit line is the one that minimizes sum of squared differences between actual and estimated results. Taking average of minimum sum of squared difference is known as Mean Squared Error (MSE). Smaller the values better the regression model.

Run and Evaluate selected models

In [49]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.33,random_state=1)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [50]:
```
models=[KNeighborsRegressor(),SVR(),DecisionTreeRegressor(),LinearRegression(),Lasso(),Ridge(),
        RandomForestRegressor(),GradientBoostingRegressor()]
maelist=[]
mselist=[]
rmselist=[]
r2list=[]
```

In [51]:
```
def create_model(model):
    m=model
    m.fit(xtrain,ytrain)
    p=m.predict(xtest)
    mae=mean_absolute_error(p,ytest)
    mse=mean_squared_error(p,ytest)
    rmse=np.sqrt(mean_squared_error(p,ytest))
    r2=r2_score(ytest,p)

    maelist.append(mae)
    mselist.append(mse)
    rmselist.append(rmse)
    r2list.append(r2)

    print(m)

    print('Mean absolute error',mae)
    print('Mean squared error',mse)
    print('Root Mean squared error',rmse)
    print('R2 Score',r2)


    print('----------------------------------------------------------------------------------------------------')
for i in models:
    create_model(i)

```

```
KNeighborsRegressor()
Mean absolute error 28146.740932642486
Mean squared error 1708658174.0990674
Root Mean squared error 41335.91869184798
R2 Score 0.7119381906222544
----------
SVR()
Mean absolute error 55561.535005685895
Mean squared error 6401199431.000112
Root Mean squared error 80007.49609255443
R2 Score -0.07917494454615026
----------
DecisionTreeRegressor()
Mean absolute error 26714.173575129535
Mean squared error 1504092957.5362694
Root Mean squared error 38782.63732053649
R2 Score 0.7464257360611781
----------
LinearRegression()
Mean absolute error 22251.033363725797
Mean squared error 1109908615.0045874
Root Mean squared error 33315.29100885339
R2 Score 0.8128810731551086
----------
Lasso()
Mean absolute error 22225.62758616074
Mean squared error 1108288356.0244567
Root Mean squared error 33290.965081001435
R2 Score 0.8131542317894991
----------
Ridge()
Mean absolute error 22180.163770269028
Mean squared error 1105086134.9968185
Root Mean squared error 33242.83584468718
R2 Score 0.8136940925979579
----------
RandomForestRegressor()
Mean absolute error 18167.88481865285
Mean squared error 787335936.4341931
Root Mean squared error 28059.50705971495
R2 Score 0.8672634363763588
----------
GradientBoostingRegressor()
Mean absolute error 16787.3669905801
Mean squared error 709357443.7948887
Root Mean squared error 26633.765107376177
R2 Score 0.8804097906458845
----------
```

Considering the R2 score Gradient boosting regressor has high accuracy of 87.87%.In order to avoid over fitting I have used cross validation .To find the best model I have found difference between predicted model and cross validation. Lasso regression model has the least difference

In [52]:
```
scorel=[]
```

In [53]:
```
from sklearn.model_selection import cross_val_score
k=KNeighborsRegressor()
scores=cross_val_score(k,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[53]:
```
array([0.69375879, 0.70078124, 0.66805857, 0.70577678, 0.67881802])
```

In [54]:
```
from sklearn.model_selection import cross_val_score
svr=SVR()
scores=cross_val_score(svr,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[54]:
```
array([-0.03485217, -0.13001527, -0.02716203, -0.11332468, -0.00296562])
```

In [55]:
```
from sklearn.model_selection import cross_val_score
dt=DecisionTreeRegressor()
scores=cross_val_score(dt,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[55]:
```
array([0.74784062, 0.6119579 , 0.74967794, 0.7486124 , 0.62964112])
```

In [56]:
```
from sklearn.model_selection import cross_val_score
lr=LinearRegression()
scores=cross_val_score(lr,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[56]:
```
array([0.80300407, 0.75769039, 0.49841011, 0.84119831, 0.80625658])
```

In [57]:
```
from sklearn.model_selection import cross_val_score
l=Lasso()
scores=cross_val_score(l,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[57]: array([0.80334236, 0.75820729, 0.50044662, 0.84198153, 0.80719067])

In [58]:
```
from sklearn.model_selection import cross_val_score
rid=Ridge()
scores=cross_val_score(rid,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[58]: array([0.81379074, 0.76974957, 0.59412562, 0.8458594 , 0.83289649])

In [59]:
```
from sklearn.model_selection import cross_val_score
rf=RandomForestRegressor()
scores=cross_val_score(rf,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[59]: array([0.88245926, 0.79329757, 0.82623669, 0.88836171, 0.83698464])

In [60]:
```
from sklearn.model_selection import cross_val_score
gb=GradientBoostingRegressor()
scores=cross_val_score(gb,x,y,scoring='r2',cv=5)
scorel.append(scores)
scores
```

Out[60]: array([0.90527498, 0.75951299, 0.89551863, 0.90777743, 0.86468295])

In [61]:
```
models=[KNeighborsRegressor(),SVR(),DecisionTreeRegressor(),LinearRegression(),Lasso(),Ridge(),
        RandomForestRegressor(),GradientBoostingRegressor()]
for i in range(0,8):
    print(models[i],"difference is",scorel[i]-r2list[i])
```

```
KNeighborsRegressor() difference is [-0.0181794  -0.01115695 -0.04387962 -0.00616141 -0.03312017]
SVR() difference is [ 0.04432277 -0.05084033  0.05201291 -0.03414974  0.07620933]
DecisionTreeRegressor() difference is [ 0.00141488 -0.13446784  0.00325221  0.00218667 -0.11678461]
LinearRegression() difference is [-0.00987701 -0.05519068 -0.31447096  0.02831724 -0.0066245 ]
Lasso() difference is [-0.00981187 -0.05494694 -0.31270761  0.02882729 -0.00596356]
Ridge() difference is [ 9.66523437e-05 -4.39445264e-02 -2.19568475e-01  3.21653087e-02
  1.92023963e-02]
RandomForestRegressor() difference is [ 0.01519583 -0.07396586 -0.04102675  0.02109827 -0.03027879]
GradientBoostingRegressor() difference is [ 0.02486519 -0.1208968   0.01510884  0.02736764 -0.01572684]
```

Lasso has least difference

Hyper parameter tuning

Grid search is arguably the most basic hyper parameter tuning method. With this technique, we simply build a model for each possible combination of all of the hyper parameter values provided, evaluating each model, and selecting the architecture which produces the best results.so I have used grid search cv to find best parameters

In [62]:
```
from sklearn.model_selection import GridSearchCV
parameters = {'alpha': (np.logspace(-1, 1,100))
              }
grid = GridSearchCV(Lasso(), param_grid = parameters, cv = 5, scoring = "r2")
```

In [63]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.33,random_state=1)
grid.fit(xtrain,ytrain)

print("Best_parameters",grid.best_params_)
```

```
Best_parameters {'alpha': 10.0}
```

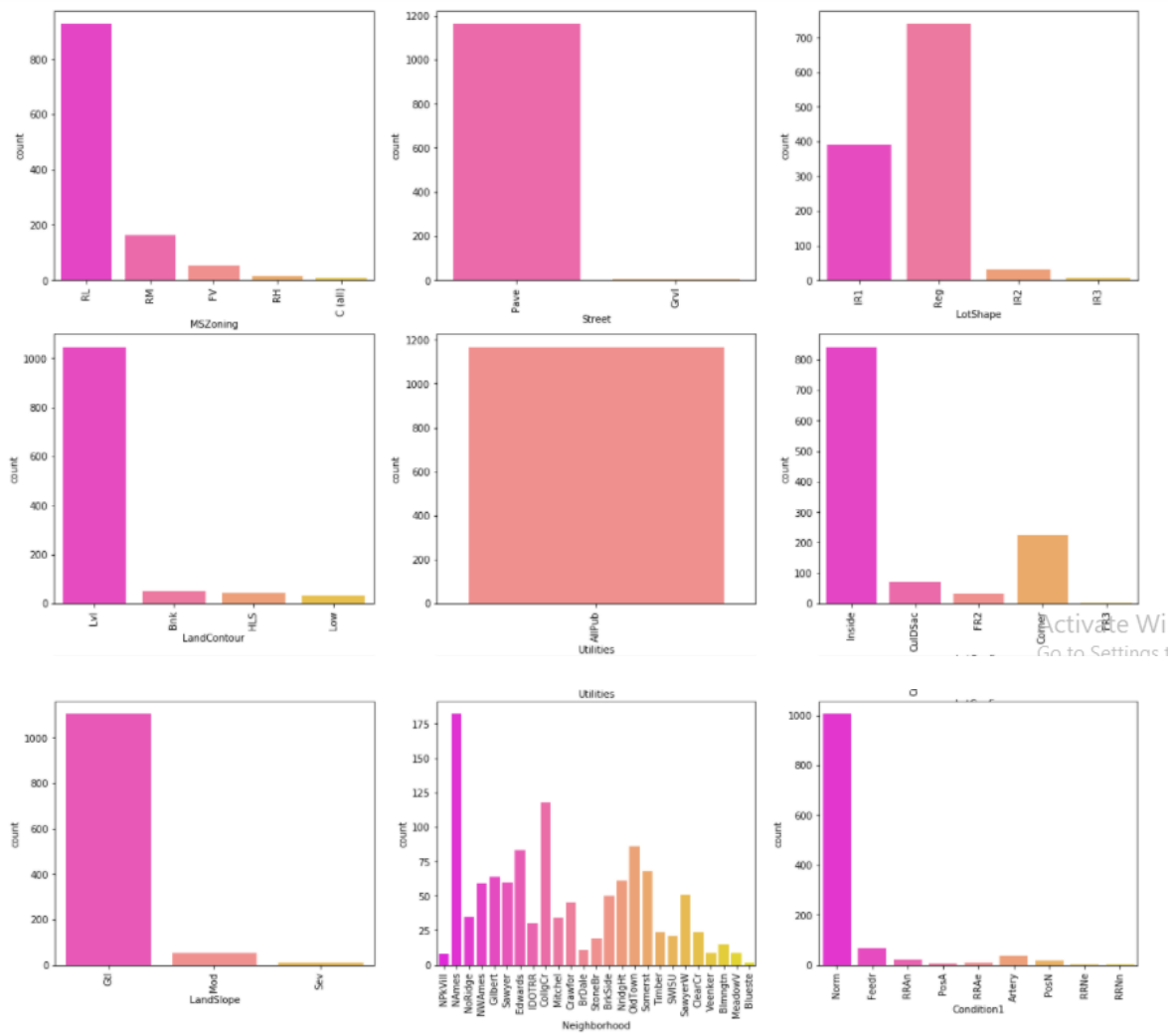
In [64]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.33,random_state=1)
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score,accuracy_score
model1=Lasso(alpha=10.0)
model1.fit(xtrain,ytrain)
p=model1.predict(xtest)
acc=model1.score(xtest,ytest)
mae=mean_absolute_error(p,ytest)
mse=mean_squared_error(p,ytest)
rmse=np.sqrt(mean_squared_error(p,ytest))
r2=r2_score(ytest,p)
print('Accuracy',(round(acc,2))*100)
print('Mean absolute error',mae)
print('Mean squared error',mse)
print('Root Mean squared error',rmse)
print('r2 score',(round(r2,2))*100)
```

```
Accuracy 82.0
Mean absolute error 22079.57395561024
Mean squared error 1096339030.0255156
Root Mean squared error 33111.010706795336
r2 score 82.0
```

Visualizations

```
drive', 'SaleType', 'SaleCondition']
```

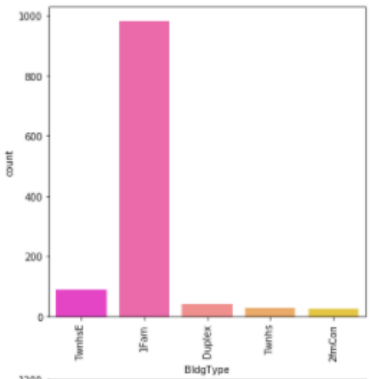
In [28]:
```
data=['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1',
ncol=3
nrows=10
plt.figure(figsize=(20,60))
for i in range(0,9):
    plt.subplot(nrows,ncol,i+1)
    sns.countplot(x=data[i],data=df_mod,palette="spring")
    plt.xticks(rotation=90)
```

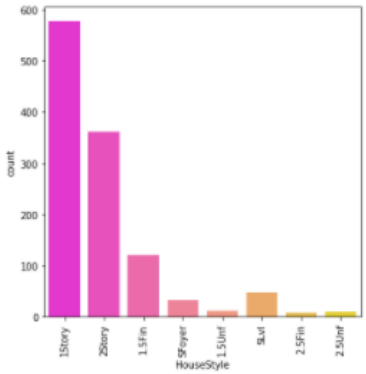


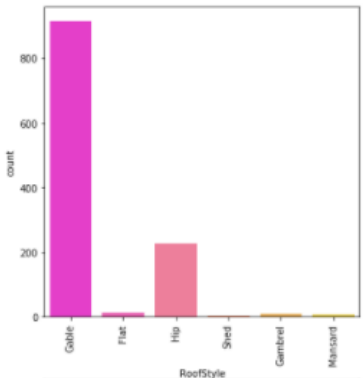
Visualizations

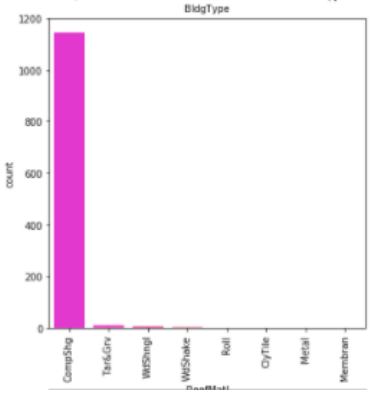
In [31]:

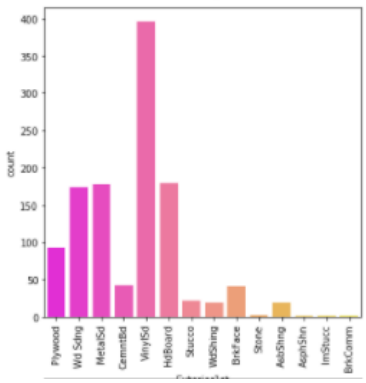
```
data=['BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCon
ncol=3
nrows=10
plt.figure(figsize=(20,70))
for i in range(0,10):
    plt.subplot(nrows,ncol,i+1)
    sns.countplot(x=data[i],data=df_mod,palette="spring")
    plt.xticks(rotation=90)
```

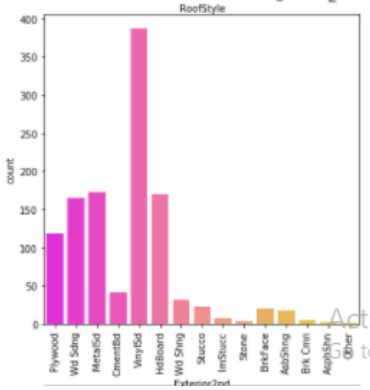


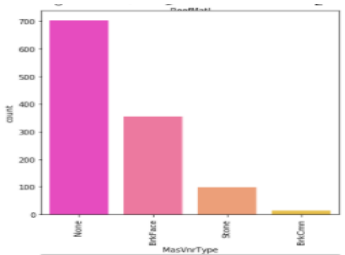


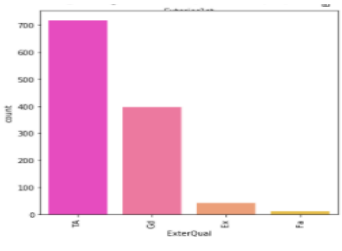


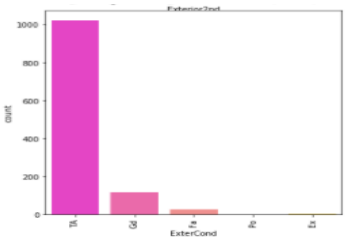


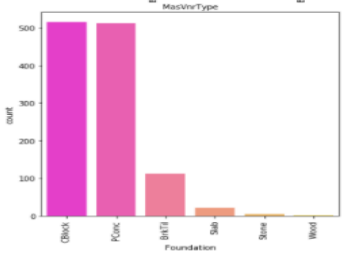






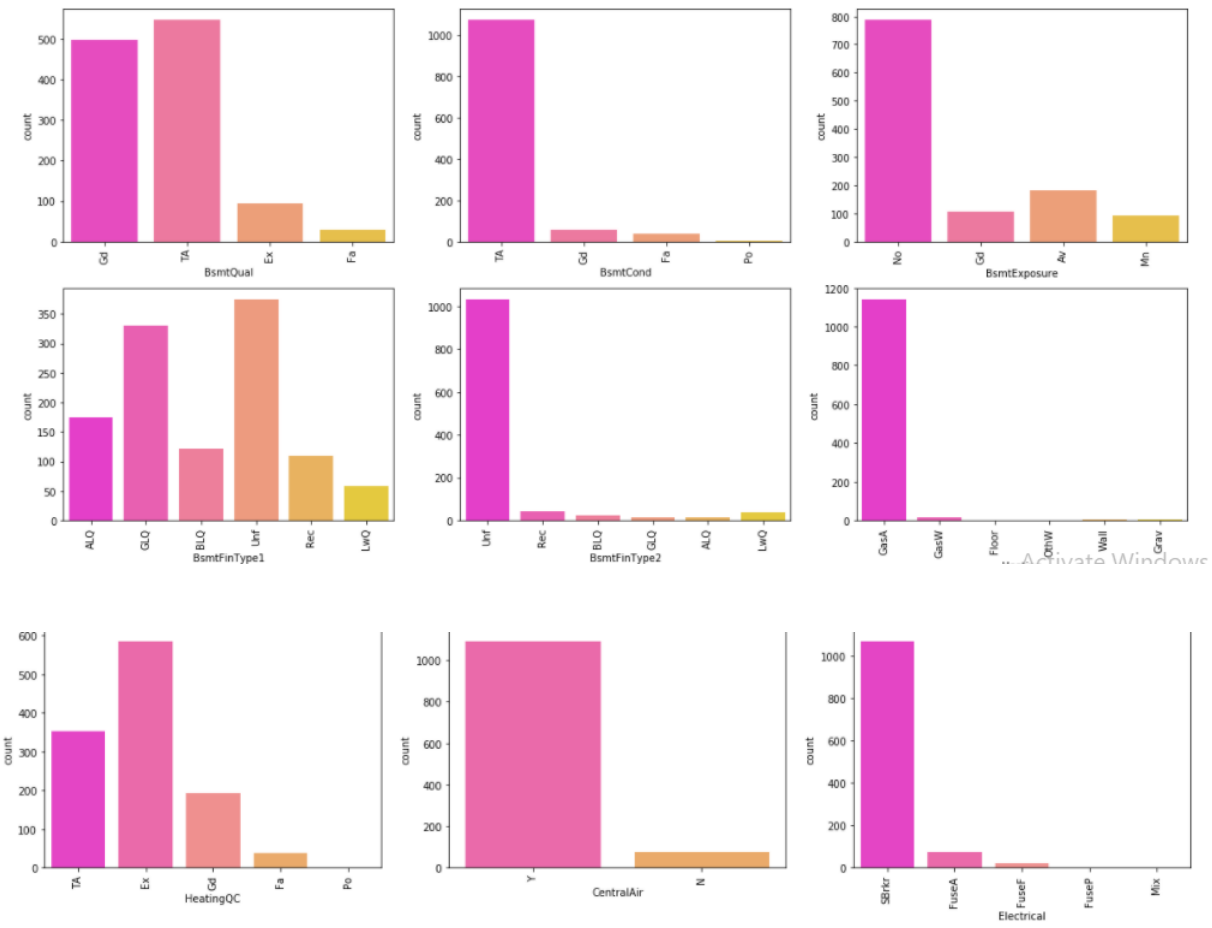






Visualizations

```
In [33]:
data=['BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'El
ncol=3
nrows=10
plt.figure(figsize=(20,50))
for i in range(0,9):
    plt.subplot(nrows,ncol,i+1)
    sns.countplot(x=data[i],data=df_mod,palette="spring")
    plt.xticks(rotation=90)
```

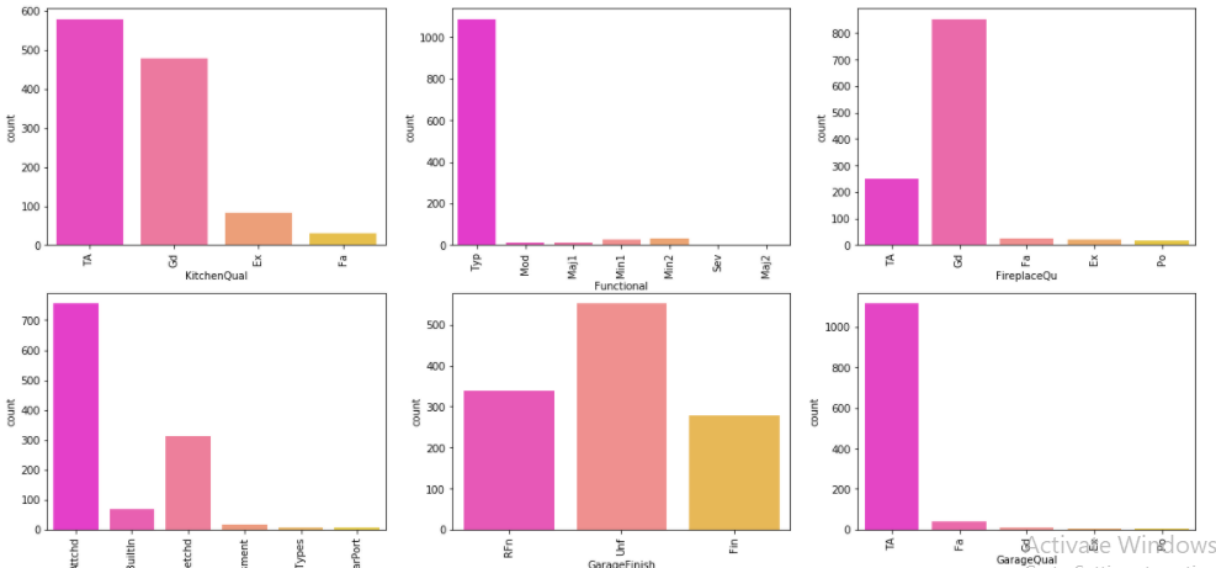


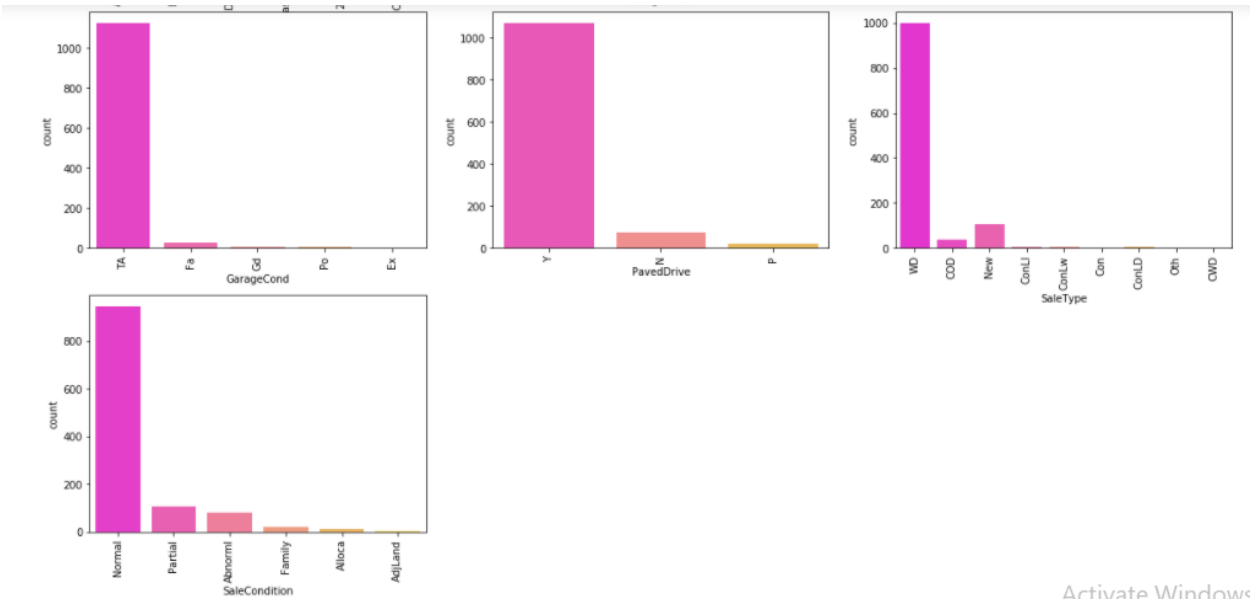
1. Majority of buyers compromise in basement quality and condition even if it is average they ready to buy.
2. majority of buyers expecting Gas forced warm air furnace for heating
3. majority of buyers expecting centralized A/C

4. majority of buyers expecting Standard Circuit Breakers & Romex for Electricity

Visualizations

In [35]:
```
data=['KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive
ncol=3
nrows=10
plt.figure(figsize=(20,50))
for i in range(0,10):
    plt.subplot(nrows,ncol,i+1)
    sns.countplot(x=data[i],data=df_mod,palette="spring")
    plt.xticks(rotation=90)
```





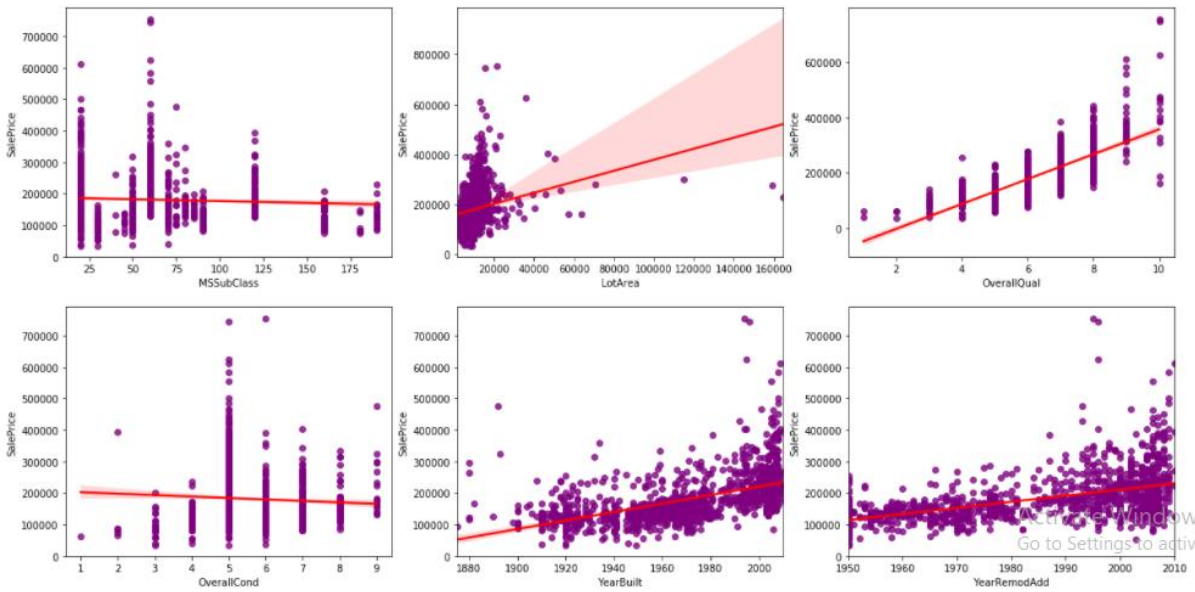
Price value is higher for the house having following amenities

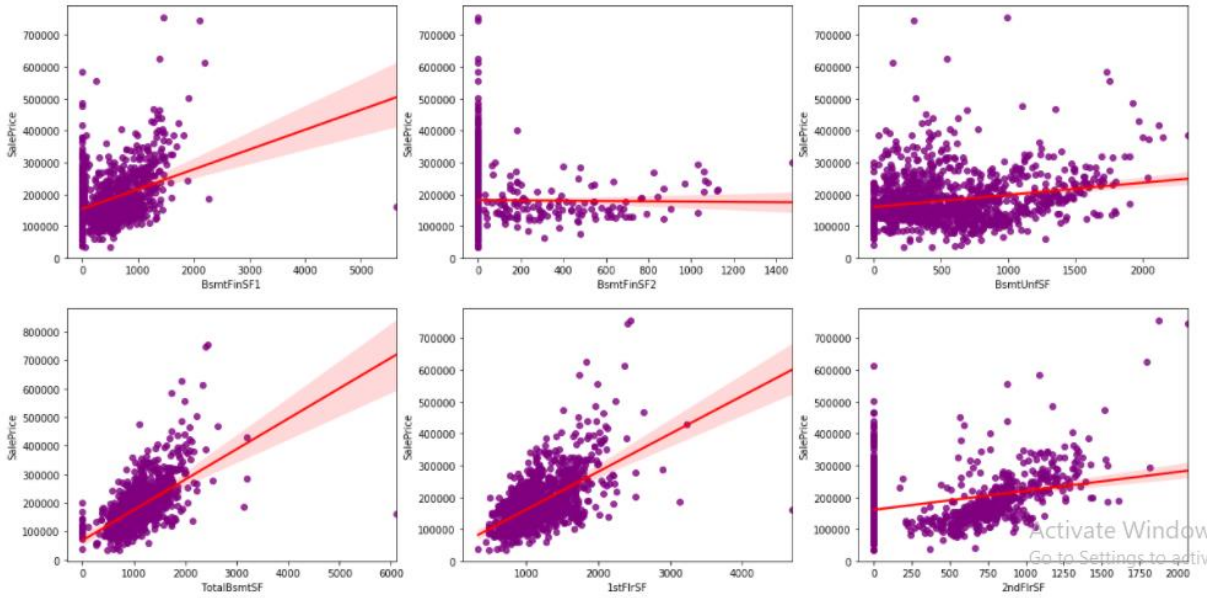
1. Excellent kitchen quality
2. Excellent finished garage
3. Excellent fire place
4. Garage condition should be typical/Average
5. Garage should be attached with house
6. Paved drive way
7. functioning-ready to move
8. Warranty deed-conventional.

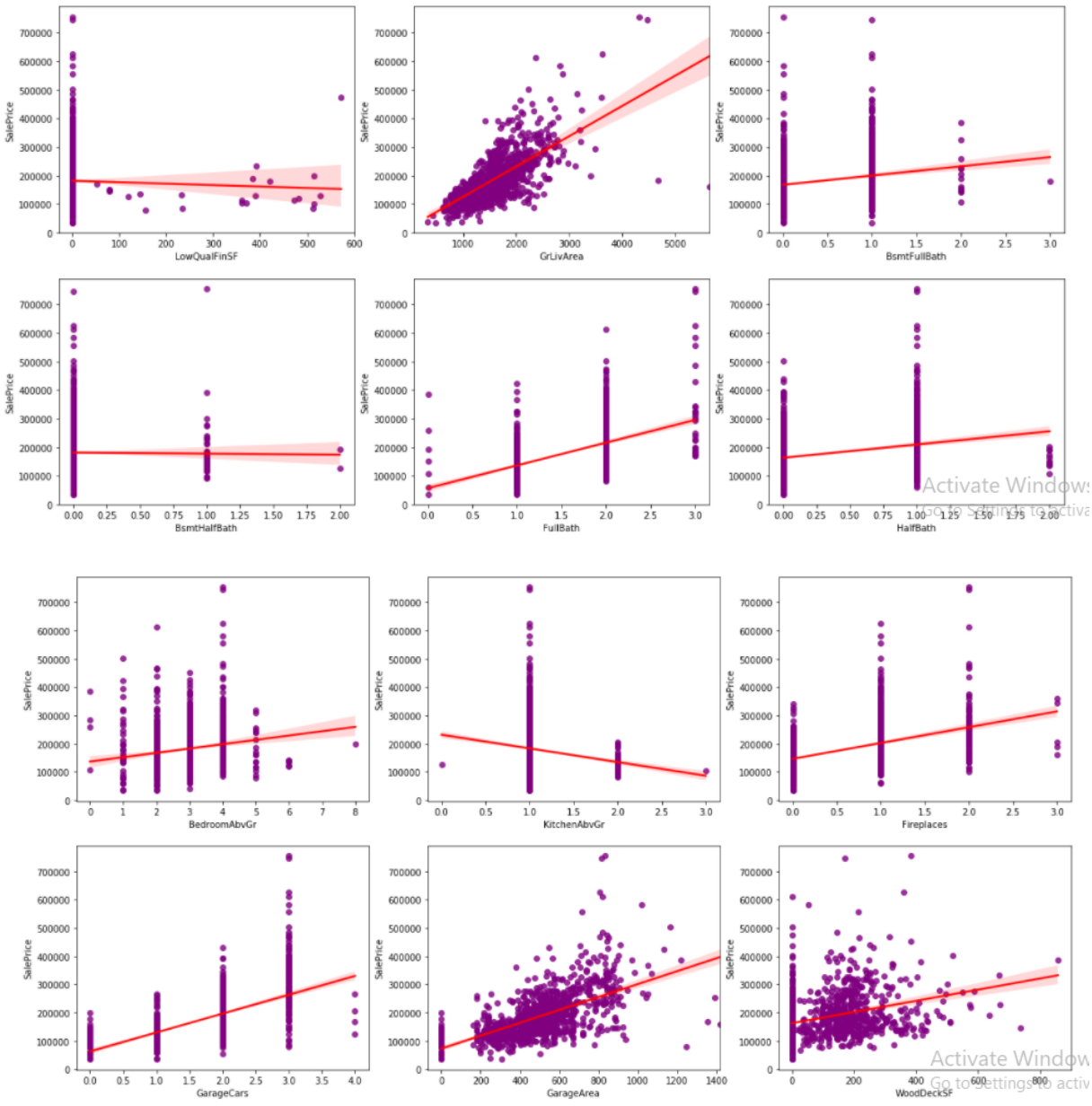
In [36]:

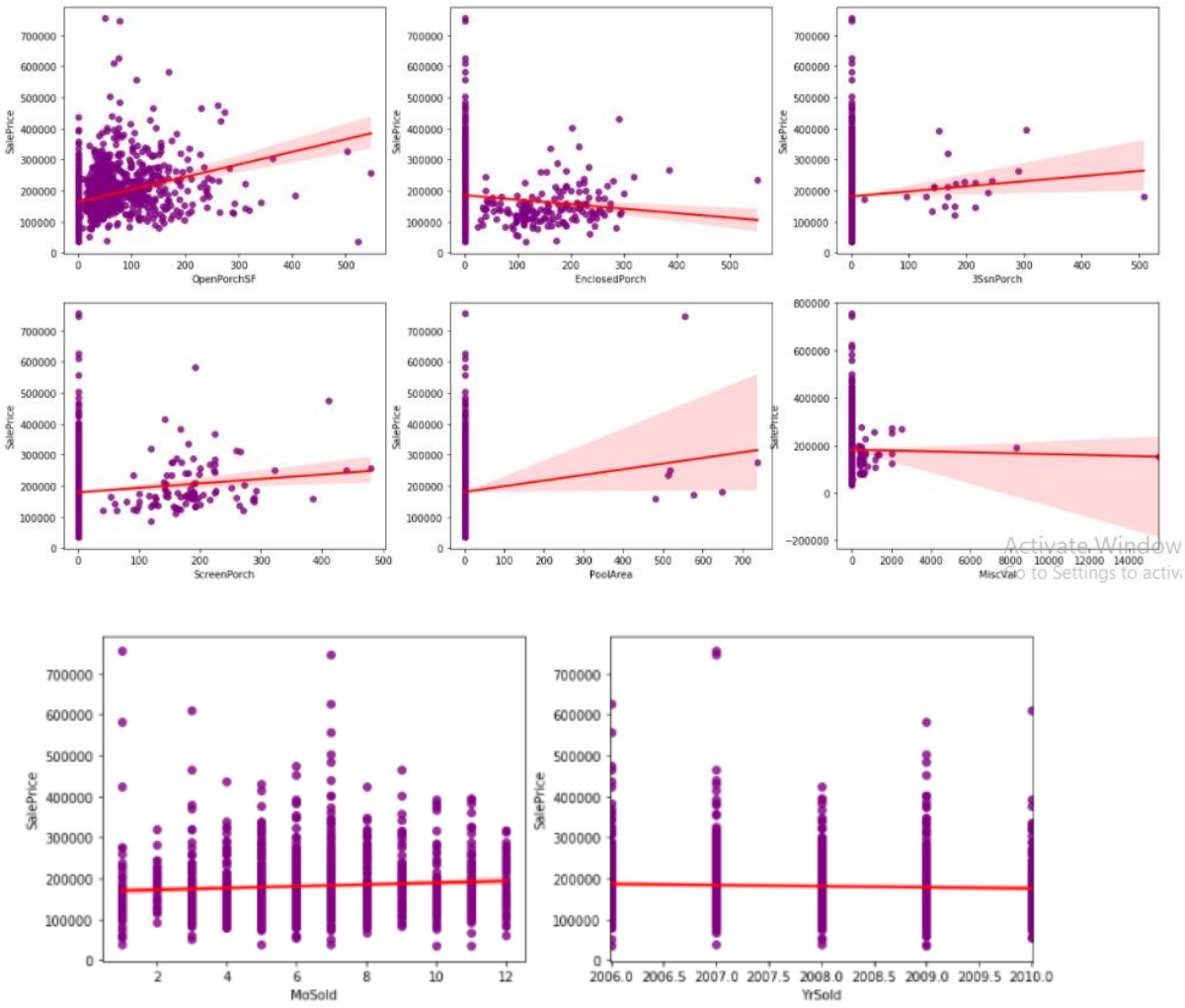
```
data=['MSSubClass','LotArea','OverallQual', 'OverallCond', 'YearBuilt','YearRemodAdd','BsmtFinSF1','BsmtFinSF2', 'BsmtU
ncol=3
nrows=11
plt.figure(figsize=(20,60))
for i in range(0,32):
    plt.subplot(nrows,ncol,i+1)
    sns.regplot(x=df_mod[data[i]],y=df_mod['SalePrice'],scatter_kws = {'color': 'purple'}, line_kws = {'color': 'r'})

```


SalePrice
MSSubClass
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd


SalePrice
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF





1. All numeric features has linear relationship with sale price
2. The featuresMSsubclass, oversllqual, overallcond, Bsmtfullbath, Bsmthalfbath, fullbath, halfbath, bedroomabvgrd, kitchenabgrd, fireplce, garagecars, mosold and yrsold are categorical feature with numerical value
3. GrLivArea, garage area, yrbuilt, woodDecksf, totalbsmtsf, 1stfloorsf, 2ndfloorsf are strongly and positively correlated with sale price.

In [40]:

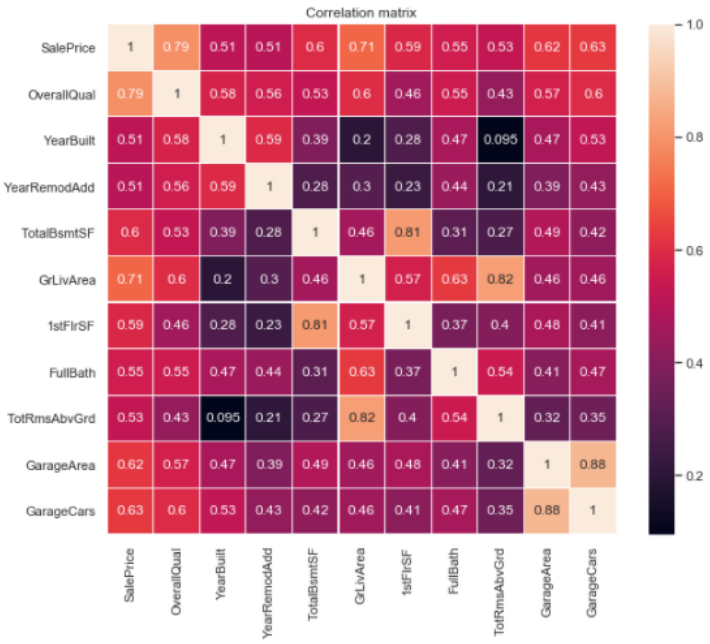
```
numColumns = df_mod.columns[df_mod.dtypes!=object]
for col in numColumns:
    if df_mod['SalePrice'].corr(df_mod[col]) > 0.5:
        print('Correlation between SalePrice and',col, '=' ,df_train['SalePrice'].corr(df_mod[col]))
```

```
Correlation between SalePrice and OverallQual = 0.7891854326077522
Correlation between SalePrice and YearBuilt = 0.5144075581459779
Correlation between SalePrice and YearRemodAdd = 0.5078305923369402
Correlation between SalePrice and TotalBsmtSF = 0.5950418180412798
Correlation between SalePrice and 1stFlrSF = 0.5876422777362369
Correlation between SalePrice and GrLivArea = 0.7073004254041905
Correlation between SalePrice and FullBath = 0.5549875557133787
Correlation between SalePrice and TotRmsAbvGrd = 0.5283629004685783
Correlation between SalePrice and GarageCars = 0.6283286172204958
Correlation between SalePrice and GarageArea = 0.6189999328245078
Correlation between SalePrice and SalePrice = 1.0
```

In [41]:

```
corr=df_mod[["SalePrice","OverallQual","YearBuilt","YearRemodAdd",
                        "TotalBsmtSF","GrLivArea","1stFlrSF","FullBath",
                            "TotRmsAbvGrd","GarageArea","GarageCars",'MSZoning']].corr()

sns.set(font_scale=1.0)
plt.figure(figsize=(10, 8))
sns.heatmap(corr, vmax=1.0, square=True, annot=True, linewidths=0.1,linecolor="white")
plt.title('Correlation matrix');
```



Interpretation of the Results

- Identifying and handling the missing values of categorical column by replacing the null value with mode value.

- Handling the missing values of numerical column by

replacing the null value with mean and median value.

- PoolQc feature have only 0.5% of data, Fence have only 20.2% data and miscFeature have only 3.7% data. We cannot predict using these features so I'm going to drop all three features.

- The feature Utilities have only one value all public utilities so it will not create impact on target value.

- Majority of buyers compromise in basement quality and condition even if it is average they ready to buy.

- majority of buyers expecting Gas forced warm air furnace for heating

- Majority of buyers expecting centralized A/C.

- Majority of buyers expecting Standard Circuit Breakers & Romex for Electricity.

- price value is higher for the house having following amenities excellent kitchen quality excellent finished garage excellent fire place garage condition should be typical/Average garage should be attached with house paved drive way functioning-ready to move warranty deed- conventional.

- All numeric features has linear relationship with sale price.

- Features MSsubclass, oversllqual, overallcond, Bsmtfullbath, Bsmthalfbath, fullbath, halfbath, bedroomabvgrd, kitchenabgrd, fireplce, garagecars, mosold, and yrsold are categorical feature with numerical value

- GrLivArea,garagearea,yrbuilt,woodDecksf,totalbsmtsf,1stfloorsf, 2ndfloor sf are strongly and positively correlated with sale price

- The outliers shown in pool area feature is not outlier. The most frequent value of pool area is 0 but we have to consider house

with pool area

CONCLUSION

Learning Outcomes of the Study in respect of Data Science

- The feature Utilities have only one value all public utilities so it will not create impact on target value.
- majority of buyers compromise in basement quality and condition even if it is average they ready to buy
- majority of buyers expecting Gas forced warm air furnace for heating
- majority of buyers expecting centralized A/C
- Majority of buyers expecting Standard Circuit Breakers & Romex for Electricity.
- price value is higher for the house having following amenities excellent kitchen quality excellent finished garage excellent fire place garage condition should be typical/Average garage should be attached with house paved drive way functioning-ready to move warranty deed- conventional.
- All numeric features has linear relationship with sale price
- Features MSsubclass, oversllqual, overallcond, Bsmtfullbath, Bsmthalfbath, fullbath, halfbath, bedroomabvgrd, kitchenabgrd, fireplce, garagecars, mosold and yrsold are categorical feature with numerical value
- GrLivArea,garagearea,yrbuilt,woodDecksf,totalbsmtsf,1stfloorsf, 2ndfloor sf are strongly and positively correlated with sale price
- The outliers shown in pool area feature is not outlier. The most frequent value of pool area is 0 but we have to consider house with pool area
- Most of the numeric features are ordinal so need to find outliers and skewness
- GrLivArea and TotalBsmtSF has a very linearly relation with 'SalePrice'.
- All the features are positively related with target, which means

that as one variable increases, the other also increases. In the case of 'TotalBsmtSF', we can see that the slope of the linear relationship is particularly high.

- OverallQual and 'YearBuilt' and also seem to be related with 'SalePrice'. The relationship seems to be stronger in the case of 'OverallQual', where the scatter plot shows how sales prices increase with the overall quality.
- 'GarageCars' and 'GarageArea' are also some of the most strongly correlated variables.
- 'TotalBsmtSF' and '1stFlrSF' are strongly and positively correlated.

Best model: Lasso best parameter :{'alpha': 10.0} after hyper tuning the accuracy increased to 82%.

The house price prediction model predicts with accuracy of **82%**

.