FLIP ROBO

Housing :Price Prediction

Submitted by:

Mohamed Adel Hafez

ACKNOWLEDGMENT

References:

- https://www.towardsdatascience.com
- www.analyticsvidhya.com
- www.kaggle.com

INTRODUCTION

- ## Business Problem Framing

You are required to model the price of houses with the available independent variables. This model 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. 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

This project is all about predicting the house Price.

Domain Knowledge: https://www.surpriseaz.gov/448/Housing-Programs

- ## Review of Literature

Information for train dataset

a) Range Index: 1168 entries, 0 to 1167
b) Data columns (total 81 columns)
c) Dtypes: float64(3), int64(35), object(43)
d) memory usage: 739.2+ KB

Information for test dataset

e) Range Index: 292 entries, 0 to 291
f) Data columns (total 80 columns)
g) Dtypes: float64(4), int64(34), object(42)
h) memory usage: 182.64+ KB

- ## Motivation for the Problem Undertaken

The project is provided to me by Flip Robo Technologies as apart of the internship programme. The exposure to real world data and the opportunity to deploy my skills in solving a real time problem has been my primary motivation.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

```
# Descriptive analysis for continous data
num_train.apply(continuous_var_summary)
```

	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	...
N	1.168000e+03	1168.000000	1168.000000	1.168000e+03	1.168000e+03	1168.000000	1168.000000	1168.000000	1168.000000	1.168000e+03	...
NMISS	0.000000e+00	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000e+00	...
SUM	1.224619e+07	7130.000000	6536.000000	2.302047e+06	2.318198e+06	118782.000000	519440.000000	54484.000000	665435.000000	1.239359e+06	...
MEAN	1.048475e+04	6.104452	5.595890	1.970931e+03	1.984759e+03	101.696918	444.726027	46.647260	569.721747	1.061095e+03	...
MEDIAN	9.522500e+03	6.000000	5.000000	1.972000e+03	1.993000e+03	0.000000	385.500000	0.000000	474.000000	1.005500e+03	...
STD	8.957442e+03	1.390153	1.124343	3.014526e+01	2.078519e+01	182.218483	462.664785	163.520016	449.375525	4.422722e+02	...
VAR	8.023577e+07	1.932525	1.264148	9.087364e+02	4.320239e+02	33203.575586	214058.702938	26738.795777	201938.362097	1.956047e+05	...
MIN	1.300000e+03	1.000000	1.000000	1.875000e+03	1.950000e+03	0.000000	0.000000	0.000000	0.000000	0.000000e+00	...
P1	1.883070e+03	3.000000	3.000000	1.900000e+03	1.950000e+03	0.000000	0.000000	0.000000	0.000000	0.000000e+00	...
P5	3.420700e+03	4.000000	4.000000	1.916000e+03	1.950000e+03	0.000000	0.000000	0.000000	0.000000	5.217500e+02	...
P10	5.298300e+03	5.000000	5.000000	1.924000e+03	1.950000e+03	0.000000	0.000000	0.000000	60.700000	6.419000e+02	...
P25	7.621500e+03	5.000000	5.000000	1.954000e+03	1.966000e+03	0.000000	0.000000	0.000000	216.000000	7.990000e+02	...
P50	9.522500e+03	6.000000	5.000000	1.972000e+03	1.993000e+03	0.000000	385.500000	0.000000	474.000000	1.005500e+03	...
P75	1.151550e+04	7.000000	6.000000	2.000000e+03	2.004000e+03	160.000000	714.500000	0.000000	816.000000	1.291500e+03	...
P90	1.431140e+04	8.000000	7.000000	2.006000e+03	2.007000e+03	320.000000	1070.300000	118.200000	1240.000000	1.594300e+03	...
P95	1.747305e+04	8.000000	8.000000	2.007000e+03	2.007000e+03	449.950000	1281.300000	378.950000	1468.000000	1.752650e+03	...
P99	3.686352e+04	10.000000	9.000000	2.009000e+03	2.009000e+03	802.600000	1583.890000	869.990000	1800.000000	2.216330e+03	...
MAX	1.646600e+05	10.000000	9.000000	2.010000e+03	2.010000e+03	1600.000000	5644.000000	1474.000000	2336.000000	6.110000e+03	...

18 rows × 35 columns

```
# Descriptive analysis for categorical data
cat_train.describe(include='O')
```

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	...	Electrical	KitchenQual	Function
count	1168	1168	1168	1168	1168	1168	1168	1168	1168	1168	...	1168	1168	116
unique	5	2	4	4	1	5	3	25	9	8	...	5	4	
top	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	Norm	...	SBrkr	TA	Ty
freq	928	1164	740	1046	1168	842	1105	182	1005	1154	...	1070	578	108

Our objective is to predict house price which can be resolve by use of regression-based algorithm. Final model is select based on evaluation benchmark among different models with different algorithms, Further Hyperparameter tuning performed to build more accurate model out of best model.

- Data Sources and their formats

  There are 2 data sets that are given in CSV format.one is training and one is testing.
  Train file: It contains all the independent variables and the target variable. The dimension of data is 1168 rows and 81 columns.

  Test file: It contains all the independent variables only without the target variable. The dimension of data is 292 rows and 80 columns.

- Data Pre-processing Done

1) Removing all the null values and dropping the column with 45% of the null values- filled it with the central tendency .
2) Encode the object columns- LabelEncoder()
3) Both Feature extraction & selection.
4) Multicollinearity- VIF technique
5) Correlation
6) Visualization

- Data Inputs- Logic- Output Relationships
  1) Selected top 39 Features for modelling.
  2) Correlation
  3) Removed outliers

- Hardware and Software Requirements and Tools Used
- Hardware used: Inteli7 with 2.4GHZ
- RAM- 4GB
- Software used: -
  - Anaconda-navigator
  - Jupyter notebook
  - Matplotlib.pyplot / Seaborn
  - Numpy / pandas
  - Scikit-learn / sklearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

- Null Values
- EDA
- Feature extraction & selection -RFE/Fregression
- VIF
- Correlation
- Visualization
- Model Building
- Hyperparameter Tuning
- Testing dataset

- Testing of Identified Approaches (Algorithms)
  - Linear Regression
  - KNN
  - SVM (Regressor)
  - Decision Tree Regressor
  - RandomForest Regressor
  - ADABoost Regressor
  - Gradient Boost Regressor
  - XGBoost Regressor

- Run and Evaluate selected models

LinearRegression

```
lr=LinearRegression()
lr.fit(x_train_sc,y_train)
pred_train=lr.predict(x_train_sc)
pred_test=lr.predict(x_test_sc)
score_train=r2_score(y_train,pred_train)
score_test=r2_score(y_test,pred_test)
mse = mean_squared_error(y_test, pred_test)
print('R2_Score_train: ',score_train)
print('R2_Score_test: ',score_test)
print('Mean absolute error :', mean_absolute_error(y_test,pred_test))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
R2_Score_train:  0.8845627870917223
R2_Score_test:  0.8744775291604862
Mean absolute error : 0.09757936996349996
RMSE =  0.1369
```

Cross validation of the Model

```
for i in(2,10):
    cv_score=cross_val_score(lr,x,y,cv=i)
    cv_mean= cv_score.mean()
    cv_std= cv_score.std()
    print(f'At Cross fold {i} the cv score mean is {cv_mean} and the cv score std is {cv_std}, testing accuracy score= {score_te
    print('\n')
```

```
At Cross fold 2 the cv score mean is 0.8521720406815334 and the cv score std is 0.011142464250633533, testing accuracy score=
0.8744775291604862


At Cross fold 10 the cv score mean is 0.8571136838625977 and the cv score std is 0.07597908469170463, testing accuracy score=
0.8744775291604862
```



```
cv=10
```

```
gbr_model.best_params_
```

```
{'learning_rate': 0.009,
 'max_depth': 4,
 'min_samples_leaf': 25,
 'min_samples_split': 25,
 'n_estimators': 1200,
 'random_state': 58,
 'subsample': 0.4}
```

```
gbr_model=gbr_model.best_estimator_
```

```
train_pred=gbr_model.predict(x_train_sc)
test_pred=gbr_model.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.9241
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.8925
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.08485719037181733
RMSE =  0.1267
```

```
# Cross_val_score
score = cross_val_score(gbr_model,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',gbr_model,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : GradientBoostingRegressor(learning_rate=0.009, max_depth=4, min_samples_leaf=25,
                          min_samples_split=25, n_estimators=1200,
                          random_state=58, subsample=0.4) :

Mean CV Score : 0.8804369766016034
Difference in R2 & CV Score: 1.2033211219053612
```

Polynominal

```
poly=PolynomialFeatures(degree=3)
xtrain_poly=poly.fit_transform(x_train_sc)
xtest_poly=poly.transform(x_test_sc)
```

```
lr2=LinearRegression()
lr2.fit(xtrain_poly,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
pred_train=lr2.predict(xtrain_poly)
pred_test=lr2.predict(xtest_poly)
```

```
r2_train = r2_score(y_train, pred_train)
print('The train R-square value is: ', r2_train.round(4))
```

```
The train R-square value is:  1.0
```

```
r2_test = r2_score(y_test, pred_test)
print('The test R-square value is: ', r2_test.round(4))
```

```
The test R-square value is:  0.7937
```

```
mse = mean_squared_error(y_test, pred_test)
print('Mean absolute error :', mean_absolute_error(y_test,pred_test))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.12140576453530225
RMSE =  0.1754
```

```
# Using Lasso regularization
tuned_parameters = [{'alpha': [0.001,0.01,0.1,0.003],'random_state':[58]}]

LassoCV = GridSearchCV(Lasso(),
                   tuned_parameters,
                   cv=10,
                   n_jobs=-1,
                   scoring='neg_mean_squared_error',
                   verbose=2)

LassoCV.fit(xtrain_poly,y_train)

print('Best combination:', LassoCV.best_params_);
```

```
Fitting 10 folds for each of 4 candidates, totalling 40 fits
Best combination: {'alpha': 0.01, 'random_state': 58}
```

```
# Lasso model
Lasso=Lasso(alpha=0.01)
lasso_model=Lasso.fit(xtrain_poly,y_train)

train_pred=lasso_model.predict(xtrain_poly)
test_pred=lasso_model.predict(xtest_poly)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.9462
```

```
r_squared = r2_score(y_test, test_pred)
print('The test R-square value is: ', r_squared.round(4))
```

```
The test R-square value is:  0.8553
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.09661152055263018
RMSE =  0.1469
```

```
knn=KNeighborsRegressor()
```

```
knn_params={'n_neighbors':[5,7,9,11],'weights':['distance','uniform'],'metric':['manhatten','euclidean']}
knn_model= GridSearchCV(knn,knn_params,cv=10,n_jobs=-1,verbose=True).fit(x_train_sc,y_train)
```

```
Fitting 10 folds for each of 16 candidates, totalling 160 fits
```

```
knn_model.best_params_
```

```
{'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}
```

```
knn_model = KNeighborsRegressor(n_neighbors=9, weights='distance',metric='euclidean').fit(x_train_sc, y_train)
```

```
train_pred=knn_model.predict(x_train_sc)
test_pred=knn_model.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  1.0
```

```
r_squared = r2_score(y_test, test_pred)
print('The test R-square value is: ', r_squared.round(4))
```

```
The test R-square value is:  0.7877
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.12249927602769145
RMSE =  0.178
```

```
# Cross_val_score
score = cross_val_score(knn,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',knn,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : KNeighborsRegressor() :

Mean CV Score : 0.7876301462591948
Difference in R2 & CV Score: 0.01032916048518473
```


SVM

```
svr=SVR(kernel='linear')
```

```
svr.fit(x_train_sc,y_train)
```

```
SVR
SVR(kernel='linear')
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  1.0
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.7877
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.12249927602769145
RMSE =  0.178
```

```
# Cross_val_score
score = cross_val_score(svr,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',svr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : SVR(kernel='linear') :

Mean CV Score : 0.8629616490197367
Difference in R2 & CV Score: -7.522821115569002
```

Decision Tree Regressor

```
dtr=DecisionTreeRegressor()
```

```
dtr.fit(x_train_sc,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
train_pred=dtr.predict(x_train_sc)
test_pred=dtr.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  1.0
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.643
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.16488604996456133
RMSE =  0.2308
```

```
# Cross_val_score
score = cross_val_score(dtr,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',dtr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : DecisionTreeRegressor() :

Mean CV Score : 0.713004572815519
Difference in R2 & CV Score: -7.001580158314312
```



```
rfr=RandomForestRegressor()
```

```
rfr.fit(x_train_sc,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
train_pred=rfr.predict(x_train_sc)
test_pred=rfr.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.9803
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.8501
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.10180068567191776
RMSE =  0.1496
```

```
# Cross_val_score
score = cross_val_score(rfr,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',rfr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : RandomForestRegressor() :

Mean CV Score : 0.8613797105702938
Difference in R2 & CV Score: -1.125548671744582
```

AdaBoostRegressor

```
adar=AdaBoostRegressor()
```

```
adar.fit(x_train_sc,y_train)
```

```
▾ AdaBoostRegressor
AdaBoostRegressor()
```

```
train_pred= adar.predict(x_train_sc)
test_pred= adar.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.8681
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.8005
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.12155701278070667
RMSE =  0.1725
```

```
# Cross_val_score
score = cross_val_score(adar,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',adar,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : AdaBoostRegressor() :

Mean CV Score : 0.7954522662398535
Difference in R2 & CV Score: 0.505242505904647
```

GradientBoost

```
gbr=GradientBoostingRegressor()
```

```
gbr.fit(x_train_sc,y_train)
```

```
▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

```
train_pred= gbr.predict(x_train_sc)
test_pred= gbr.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.9635
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.8843
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.08939414470938882
RMSE =  0.1314
```

```
# Cross_val_score
score = cross_val_score(gbr,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',gbr,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : GradientBoostingRegressor() :

Mean CV Score : 0.8811804593722684
Difference in R2 & CV Score: 0.3162646156351059
```

```
xgb=xgboost.XGBRegressor()
```

```
xgb.fit(x_train_sc,y_train)
```

```
train_pred= xgb.predict(x_train_sc)
test_pred= xgb.predict(x_test_sc)
```

```
r_squared = r2_score(y_train, train_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.9998
```

```
r_squared = r2_score(y_test, test_pred)
print('The train R-square value is: ', r_squared.round(4))
```

```
The train R-square value is:  0.8426
```

```
mse = mean_squared_error(y_test, test_pred)
print('Mean absolute error :', mean_absolute_error(y_test,test_pred))
print('RMSE = ', np.sqrt(mse).round(4))
```

```
Mean absolute error : 0.1027122775973885
RMSE =  0.1532
```

```
# Cross_val_score
score = cross_val_score(xgb,X_scale, y, cv=10)
print('\033[1m'+'Cross Validation Score :',xgb,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(y_test,test_pred)*100)-(score.mean()*100))
```

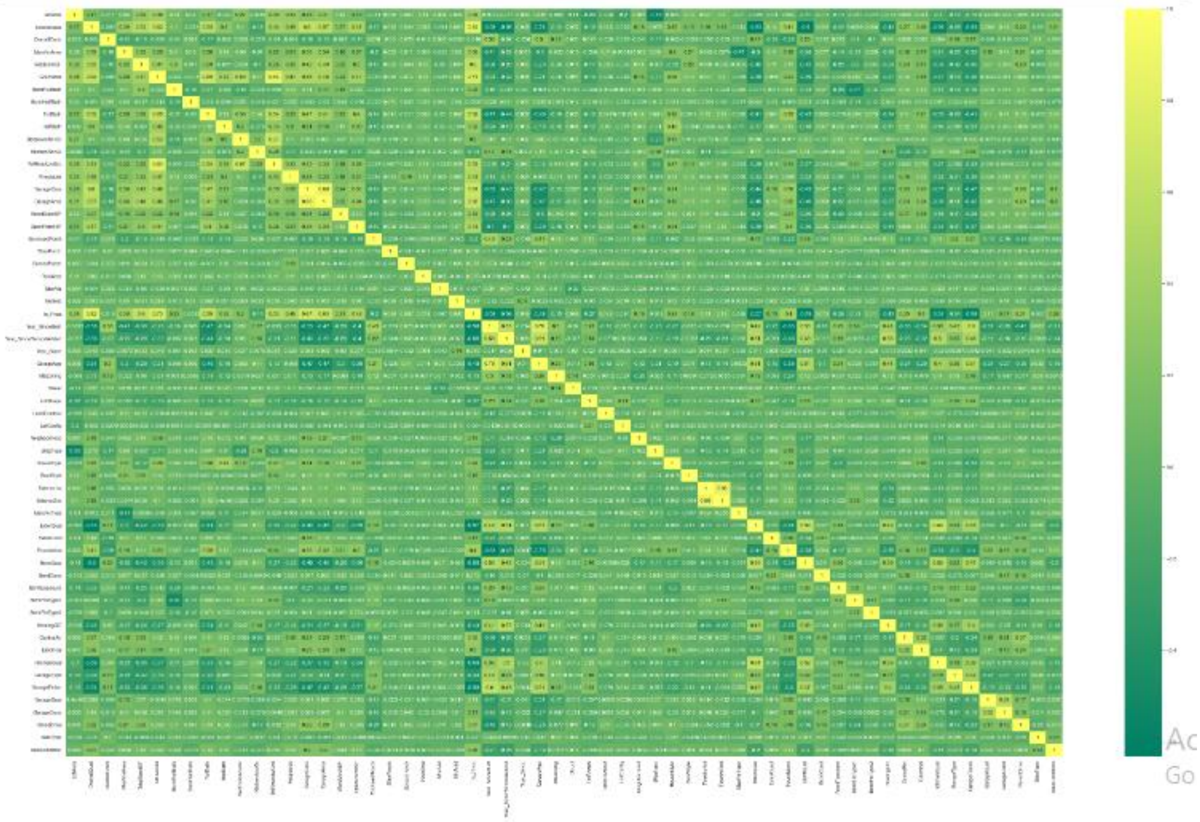
```
Cross Validation Score : XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...) :

Mean CV Score : 0.855554252537833
Difference in R2 & CV Score: [illegible]
```

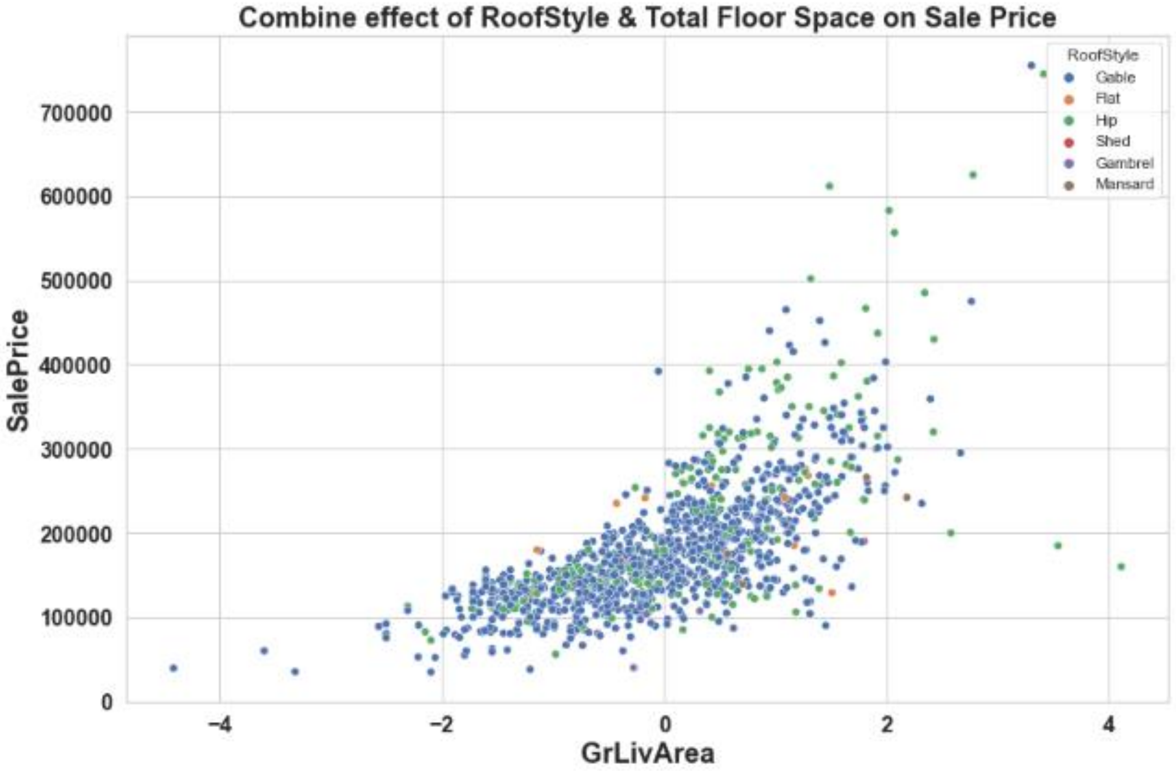

- Key Metrics for success in solving problem under consideration

  Using R2score for accuracy as it is a regression problem with cost function loss metrics as RMSE,MAE

- Visualizations

```python
# heatmap correlation
plt.figure(figsize = (40,25))
sns.heatmap(train_final.corr(), annot=True,cmap='summer')
plt.show()
```

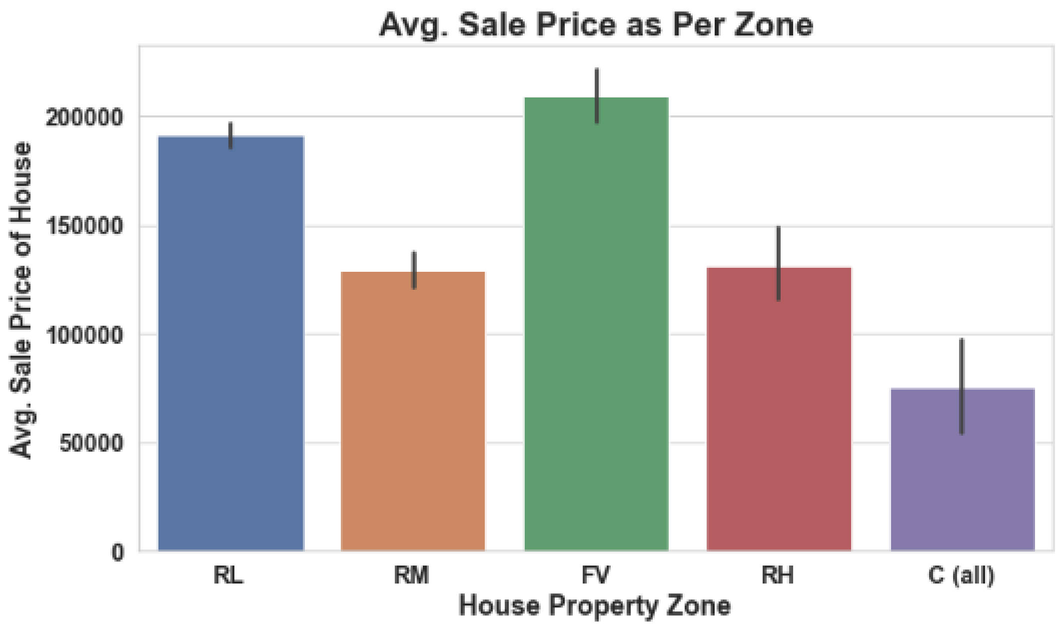




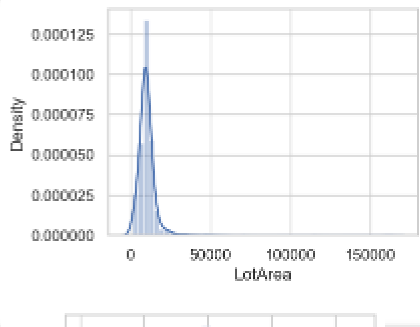
Observation :-

For High floor area construction mainly Hip style Roof is used and invariably high cost properties mostly comes up with Hip Style Roof.*

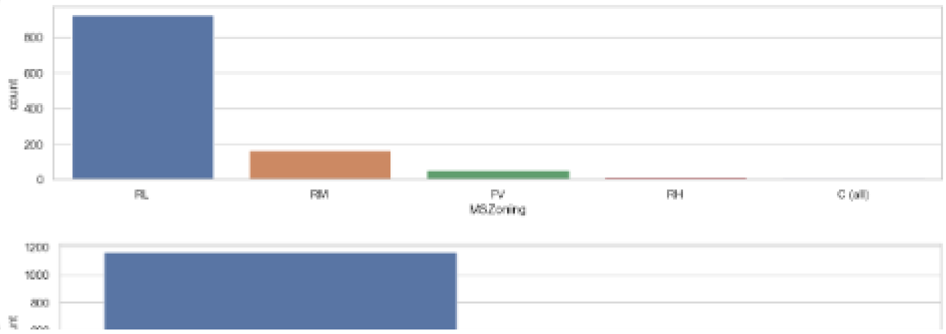
```
plt.rcParams['figure.autolayout'] = True
plt.figure(figsize = (10,6))
sns.barplot(y = num_train_eda['SalePrice'], x= cat_train['MSZoning'])
plt.title('Avg. Sale Price as Per Zone', fontsize=22, fontweight='bold')
plt.xlabel('House Property Zone', fontsize= 18, fontweight='bold')
plt.ylabel('Avg. Sale Price of House', fontsize=18, fontweight= 'bold')
plt.xticks(fontsize=16,fontweight ='bold')
plt.yticks(fontsize=16,fontweight ='bold')
plt.tight_layout()
plt.show()
```



In [37]:
```
for col in num_train_eda.columns:
    plt.figure(figsize=(4,3),dpi=100)
    sns.distplot(num_train_eda[col])
    plt.show()
```



In [38]:
```
for col in cat_train.columns:
    plt.figure(figsize=(15,3),dpi=100)
    sns.countplot(cat_train[col])
    plt.show()
```



In [39]:
```
## Insights from the above graphs

# Average Lot area is around 10000 ft2
# There are old houses built in 1878 and ne whouses built in 2009
# Most of the houses are Residentail Low density & Pave street.
# Landcontoour is flat, Lotshape is regular.
# All Public Utilities are availabe.
# Lotconfig are inside, Land slope are Gentle, Most Neighborhood is North Ames.
# Normal is the only dominant in the condition 2.
# Most of the dwelling are Single-family Detached.
# Most of the houses are 1story(50%) or 2story (30%).
# All houses are Standard (Composite) Shingle roof material.
# Roof style are either Gable or Hip.
# all the exteriorr of the houses are Vinyl Siding.
# Most sale conditions are normal.
# all Houses have Gas heating.
# Most houses have Standard Circuit Breakers,Romex and air conditions
# Most of the sale type are arranty Deed - Conventional (85%).
# Most of the garage houses are in normal condition.
# 50% of the Interior finish of the garage is unfinished.
# More than 70% of the garages are Garage Location.
```

Observation:

There is No Significant relationship found between Sale price & Lot area.

CONCLUSION

From our most important features

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.