```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn import ensemble
from sklearn.linear_model import Lasso,Ridge
#load train data
df_data=pd.read_csv("/content/data_price.csv")
df_data.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlop
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	G
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	G
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	G
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	G
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	G

5 rows × 81 columns

```python
#to know each and every column execute the following
print(df_data.columns)
print(df_data.shape)
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       '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', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
(1460, 81)
```

```python
total = df_data.isnull().sum().sort_values(ascending=False)
percent = (df_data.isnull().sum()/df_data.isnull().count()).sort_valu
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Pe
missing_data.head(20)
```

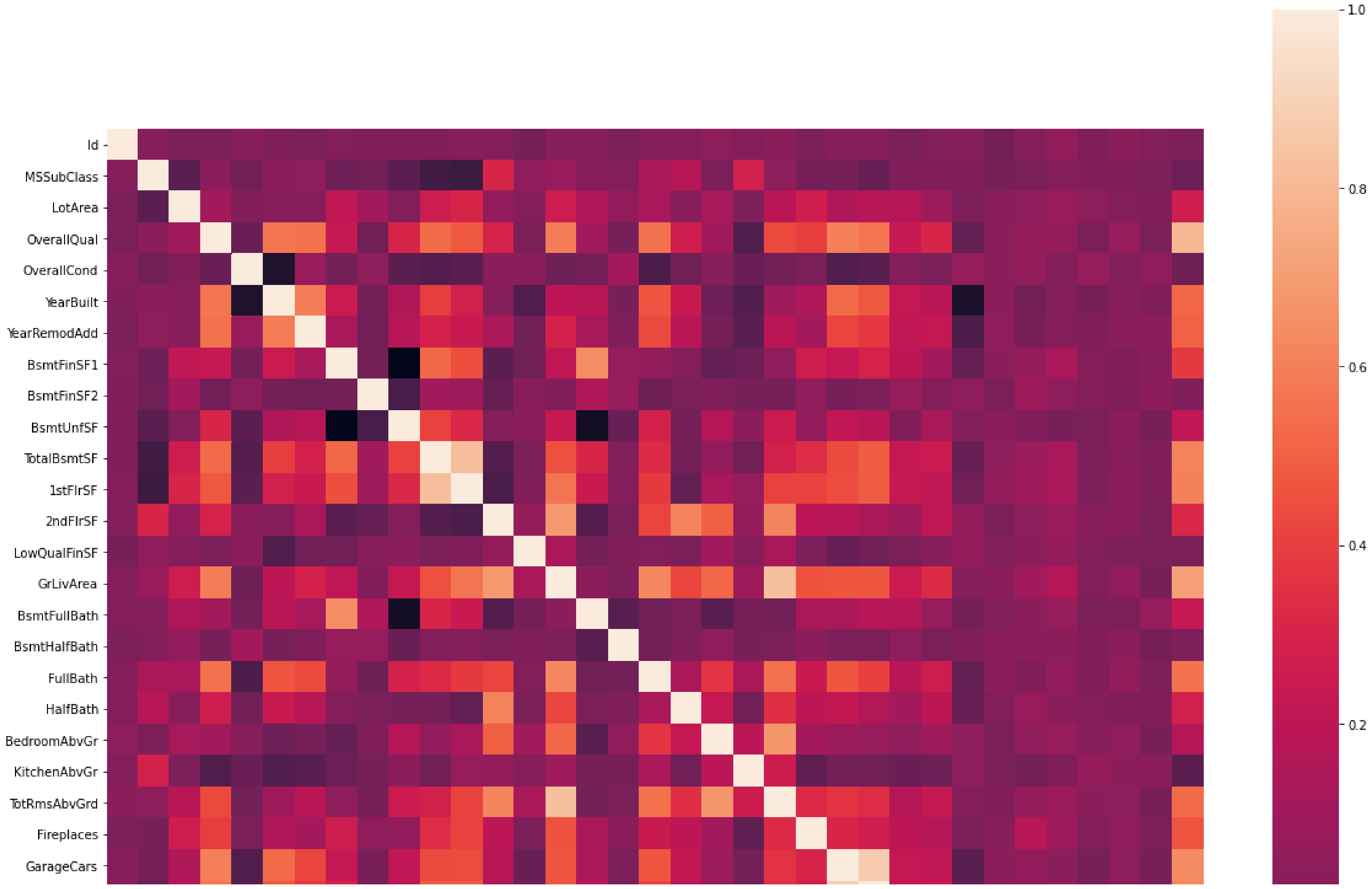
	Total	Percent
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageCond	81	0.055479

```
df_data= df_data.drop(missing_data[missing_data['Total']>1].index.val
df_data= df_data.drop(df_data.loc[df_data['Electrical'].isnull()].ind
```

GarageFinish 81 0.055479

```
corr_mat=df_data.corr()
fi,ax=plt.subplots(figsize=(20,20))
sns.heatmap(corr_mat,square=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7428024630>
```




```
del df_data['Id']
```

```
le=LabelEncoder()
cat_mask= df_data.dtypes=='object'
cat_cols= df_data.columns[cat_mask].tolist()
cat_cols
```

```
['MSZoning',
 'Street',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
```

```
#Lets convert the columns to one hot encoding
df_data[cat_cols]=df_data[cat_cols].apply(lambda x: le.fit_transform(x
df_data_c = df_data.copy()
#get_dummies is used for one hot encoding
df_data_c = pd.get_dummies(df_data_c,columns=cat_cols)
```

```
 'Heating'
```

```
x_train, x_test, y_train, y_test = train_test_split(df_data_c.drop('S
y_train= y_train.values.reshape(-1,1)
y_test= y_test.values.reshape(-1,1)
```

```
 'Functional',
```

Normalize the values in train and test using Standard Scaler function.

```
 'SaleCondition']
```

```
sc_X = StandardScaler()
sc_y = StandardScaler()
x_train = sc_X.fit_transform(x_train)
x_test = sc_X.fit_transform(x_test)
y_train = sc_X.fit_transform(y_train)
y_test = sc_y.fit_transform(y_test)
```

```
lm = LinearRegression()
lm.fit(x_train,y_train)
#predictions on train data
x_pred = lm.predict(x_train)
x_pred = x_pred.reshape(-1,1)
#Prediction of validation data
y_predictions = lm.predict(x_test)
y_predictions= y_predictions.reshape(-1,1)
def scores_(y,x):
    print('MAE:', metrics.mean_absolute_error(y, x))
    print('MSE:', metrics.mean_squared_error(y, x))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y, x)))
    print('R2 Score:' ,metrics.r2_score(y,x))
print('InSample_accuracy')
scores_(y_train, x_pred)
print('---------------------------')
print('OutSample_accuracy')
scores_(y_test,y_predictions)
```

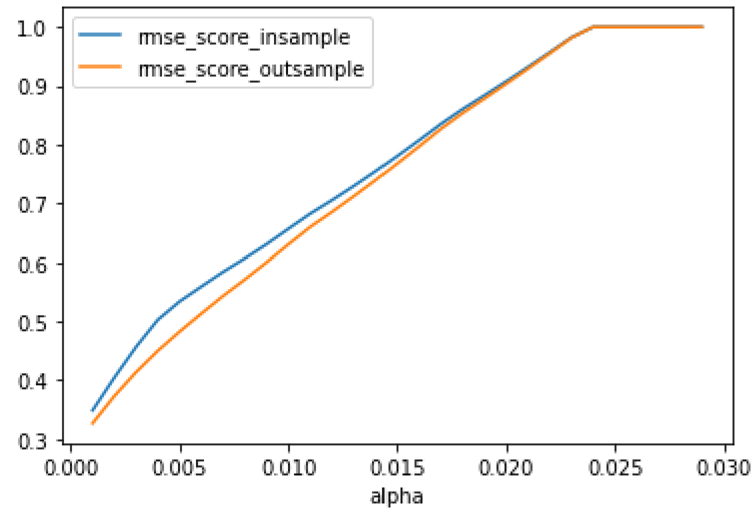
```
InSample_accuracy
MAE: 0.1828459849312525
MSE: 0.08080438090379947
RMSE: 0.2842611139494804
R2 Score: 0.9191956190962005
---------------------------
OutSample_accuracy
MAE: 29345417710.666924
MSE: 3.034299649072542e+21
RMSE: 55084477387.66832
R2 Score: -3.034299649072542e+21
```

The model performed really well on training data with a good 0.92 r2 score and < 1 RMSE score but with test data the performance is no where near good. This is a clear overfitting model. Reason might me because of numerous features. To tackle this we can perform Ridge and Lasso regularization.

```
gularization(model,alpha_range):
se_score_insample=[]
se_score_outsample=[]
_score_insample=[]
_score_outsample=[]
r i in alpha_range:
  regularization = model(alpha=i,normalize=True)
  regularization.fit(x_train,y_train)
  y_pred_train = regularization.predict(x_train)
  y_pred_train = y_pred_train.reshape(-1,1)
```

```
    y_pred_test=regularization.predict(x_test)
    y_pred_test = y_pred_test.reshape(-1,1)
    rmse_score_insample.append(np.sqrt(metrics.mean_squared_error(y_tra
    rmse_score_outsample.append(np.sqrt(metrics.mean_squared_error(y_te
    r2_score_insample.append(metrics.r2_score(y_train, y_pred_train))
    r2_score_outsample.append(metrics.r2_score(y_test, y_pred_test))
  =pd.DataFrame()
  ['alpha']=alpha_range
  ['rmse_score_insample'] = rmse_score_insample
  ['rmse_score_outsample']= rmse_score_outsample
  ['r2_score_insample'] = r2_score_insample
  ['r2_score_outsample'] = r2_score_outsample
  turn df.plot(x = 'alpha', y = ['rmse_score_insample', 'rmse_score_out
  range_lasso = np.arange(0.001,0.03,0.001)
  regularization(Lasso,alpha_range_lasso))
```

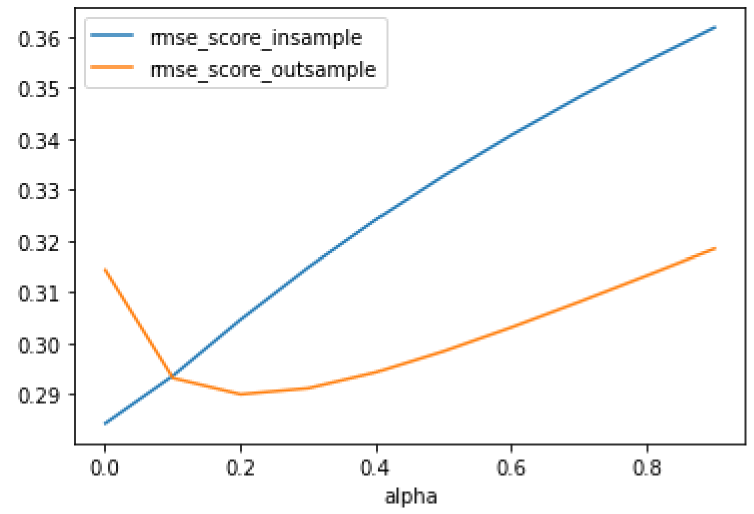
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



We can see that there is no huge difference in in sample and out sample RMSE scores so Lasso has resolved overfitting. One observation here is that after alpha= 0.017 there is no difference in RMSE scores of In sample and Out sample.

```
alpha_range_ridge = np.arange(0.001,1,0.1)
print(regularization(Ridge,alpha_range_ridge))
#writing functions helps reduce redundant lines of code as seen #abov
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



We see in the graph that around alpha=0.1 there is no much difference in the RMSE scores and clearly there is no sign of over fitting as there is very less difference of insample and outsample RMSE scores as compared to huge difference in Linear Regression.