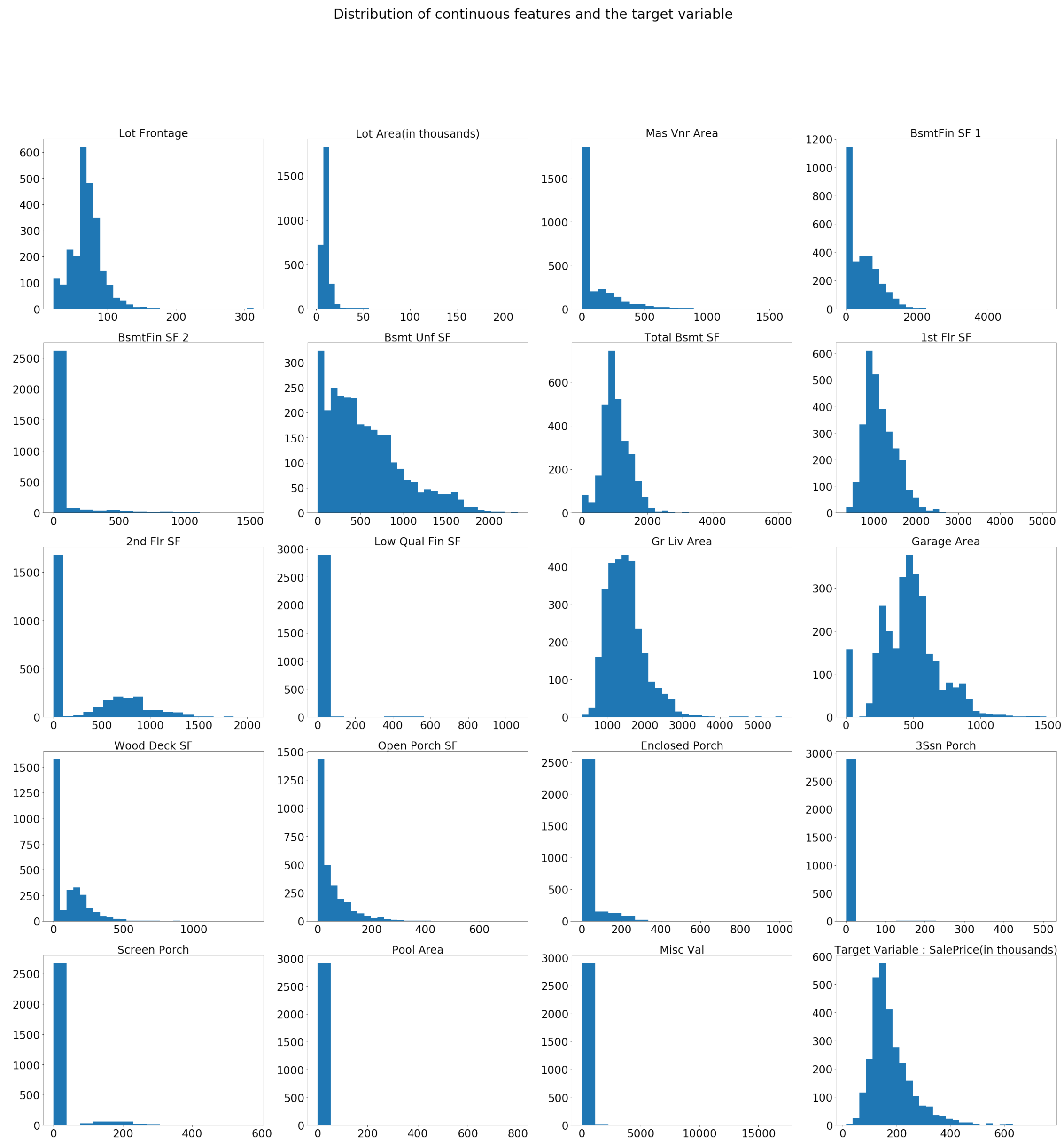
February 20, 2019

# 2 1.1

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        df=pd.read_excel("http://www.amstat.org/publications/jse/v19n3/decock/AmesHousing.xls")

In [2]: col=[4,5,27,35,37,38,39,44,45,46,47,63,67,68,69,70,71,72,76,81]
        cont=pd.DataFrame(df.iloc[:,col])
        df1=df.iloc[:,:81]
        categorical=df1.select_dtypes(include=['object'])
        continuous1=df1.select_dtypes(exclude=['object'])
        for i in range(43):
            categorical.iloc[:,i].fillna('NA',inplace=True)
        for i in range(38):
            continuous1.iloc[:,i].fillna(0,inplace=True)
        df2=categorical.join(continuous1)

In [4]: fig, ax = plt.subplots(5, 4, figsize=(40, 40))
        plt.rc('xtick', labelsize=24)
        plt.rc('ytick', labelsize=24)
        _=ax[0,0].hist(cont.iloc[:,0],bins=30)
        _=ax[0,0].set_xlabel(cont.columns[0],fontsize=24)
        ax[0,0].xaxis.set_label_position('top')

        _=ax[0,1].hist(cont.iloc[:,1]/1000,bins=35)
        _=ax[0,1].set_xlabel(cont.columns[1]+"(in thousands)",fontsize=24)
        ax[0,1].xaxis.set_label_position('top')

        _=ax[0,2].hist(cont.iloc[:,2],bins=25)
        _=ax[0,2].set_xlabel(cont.columns[2],fontsize=24)
        ax[0,2].xaxis.set_label_position('top')

        _=ax[0,3].hist(cont.iloc[:,3],bins=30)
        _=ax[0,3].set_xlabel(cont.columns[3],fontsize=24)
```

```
ax[0,3].xaxis.set_label_position('top')

_=ax[1,0].hist(cont.iloc[:,4],bins=15)
_=ax[1,0].set_xlabel(cont.columns[4],fontsize=24)
ax[1,0].xaxis.set_label_position('top')

_=ax[1,1].hist(cont.iloc[:,5],bins=30)
_=ax[1,1].set_xlabel(cont.columns[5],fontsize=24)
ax[1,1].xaxis.set_label_position('top')

_=ax[1,2].hist(cont.iloc[:,6],bins=30)
_=ax[1,2].set_xlabel(cont.columns[6],fontsize=24)
ax[1,2].xaxis.set_label_position('top')

_=ax[1,3].hist(cont.iloc[:,7],bins=30)
_=ax[1,3].set_xlabel(cont.columns[7],fontsize=24)
ax[1,3].xaxis.set_label_position('top')

_=ax[2,0].hist(cont.iloc[:,8],bins=20)
_=ax[2,0].set_xlabel(cont.columns[8],fontsize=24)
ax[2,0].xaxis.set_label_position('top')

_=ax[2,1].hist(cont.iloc[:,9],bins=15)
_=ax[2,1].set_xlabel(cont.columns[9],fontsize=24)
ax[2,1].xaxis.set_label_position('top')

_=ax[2,2].hist(cont.iloc[:,10],bins=30)
_=ax[2,2].set_xlabel(cont.columns[10],fontsize=24)
ax[2,2].xaxis.set_label_position('top')

_=ax[2,3].hist(cont.iloc[:,11],bins=30)
_=ax[2,3].set_xlabel(cont.columns[11],fontsize=24)
ax[2,3].xaxis.set_label_position('top')

_=ax[3,0].hist(cont.iloc[:,12],bins=30)
_=ax[3,0].set_xlabel(cont.columns[12],fontsize=24)
ax[3,0].xaxis.set_label_position('top')

_=ax[3,1].hist(cont.iloc[:,13],bins=30)
_=ax[3,1].set_xlabel(cont.columns[13],fontsize=24)
ax[3,1].xaxis.set_label_position('top')

_=ax[3,2].hist(cont.iloc[:,14],bins=15)
_=ax[3,2].set_xlabel(cont.columns[14],fontsize=24)
ax[3,2].xaxis.set_label_position('top')

_=ax[3,3].hist(cont.iloc[:,15],bins=20)
_=ax[3,3].set_xlabel(cont.columns[15],fontsize=24)
```

```
ax[3,3].xaxis.set_label_position('top')

_=ax[4,0].hist(cont.iloc[:,16],bins=15)
_=ax[4,0].set_xlabel(cont.columns[16],fontsize=24)
ax[4,0].xaxis.set_label_position('top')

_=ax[4,1].hist(cont.iloc[:,17],bins=15)
_=ax[4,1].set_xlabel(cont.columns[17],fontsize=24)
ax[4,1].xaxis.set_label_position('top')

_=ax[4,2].hist(cont.iloc[:,18],bins=15)
_=ax[4,2].set_xlabel(cont.columns[18],fontsize=24)
ax[4,2].xaxis.set_label_position('top')

_=ax[4,3].hist(cont.iloc[:,19]/1000,bins=30)
_=ax[4,3].set_xlabel("Target Variable : "+ cont.columns[19]+"(in thousands)"
                     ,fontsize=24)
ax[4,3].xaxis.set_label_position('top')

_=fig.suptitle('Distribution of continuous features and the target variable',
               fontsize=30)
```



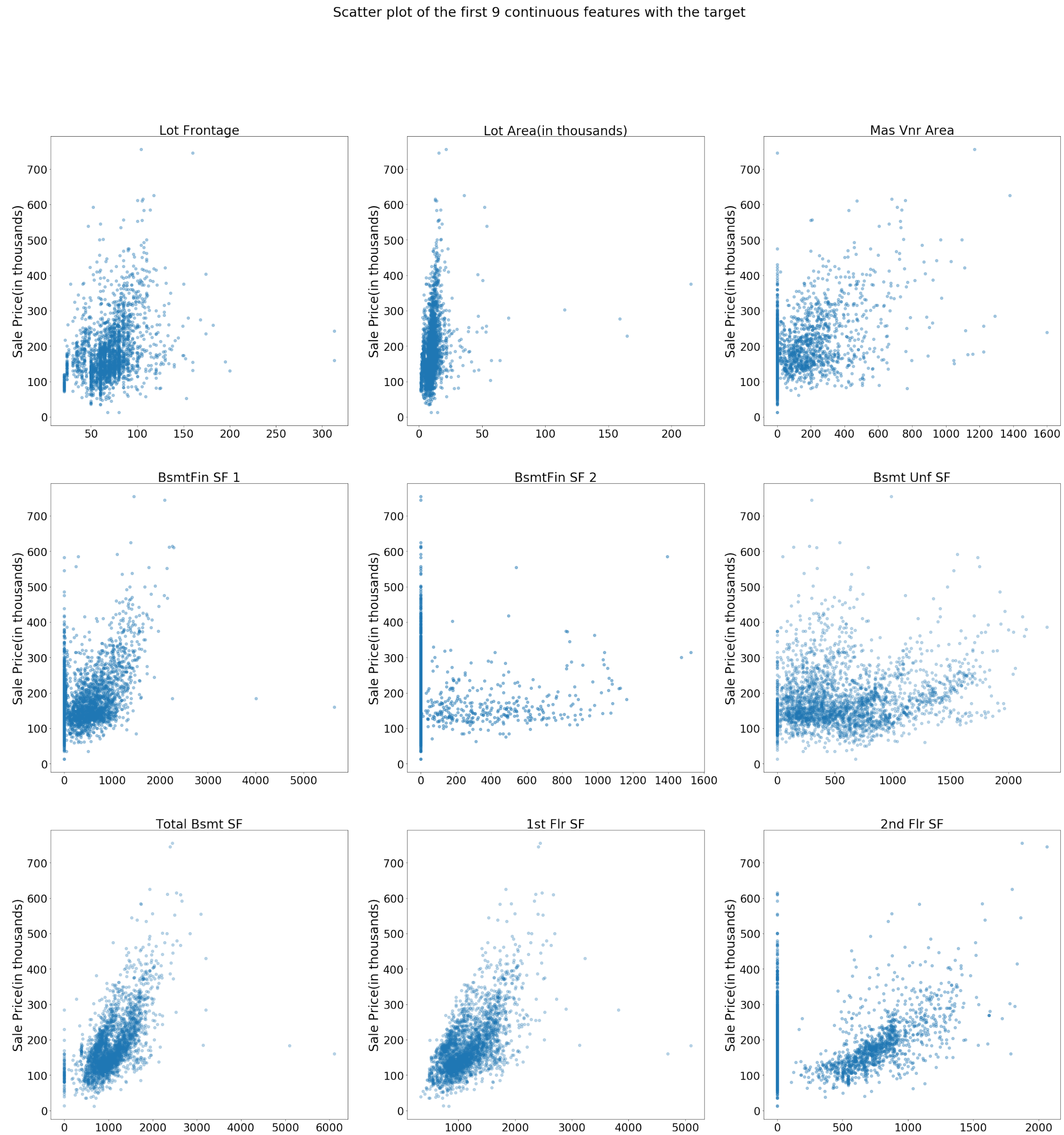

In [ ]:
```
'''
We observe that a lot of these features require imputation because most of the
values for these features are 0.
Some of these features are : BsmtFin SF2, 2nd Flr SF, Low Qual Fin SF,
Wood Deck SF, Enclosed Porch,3Ssn Porch, Screen Porch,Pool Area, Misc Val
Thus to avoid the distribution shown, the missing (here represented as '0')
values need to be filled in using imputation techniques like mean.
Some features like Lot Frontage have a hhghly skewed distribution. This shows
that most of the values of this feature lie in a certain range and the distribution
appears skewed because of some unexpectedly higher values of this feature.
```

```
These values can be outliers.
'''
```

# 3 1.2

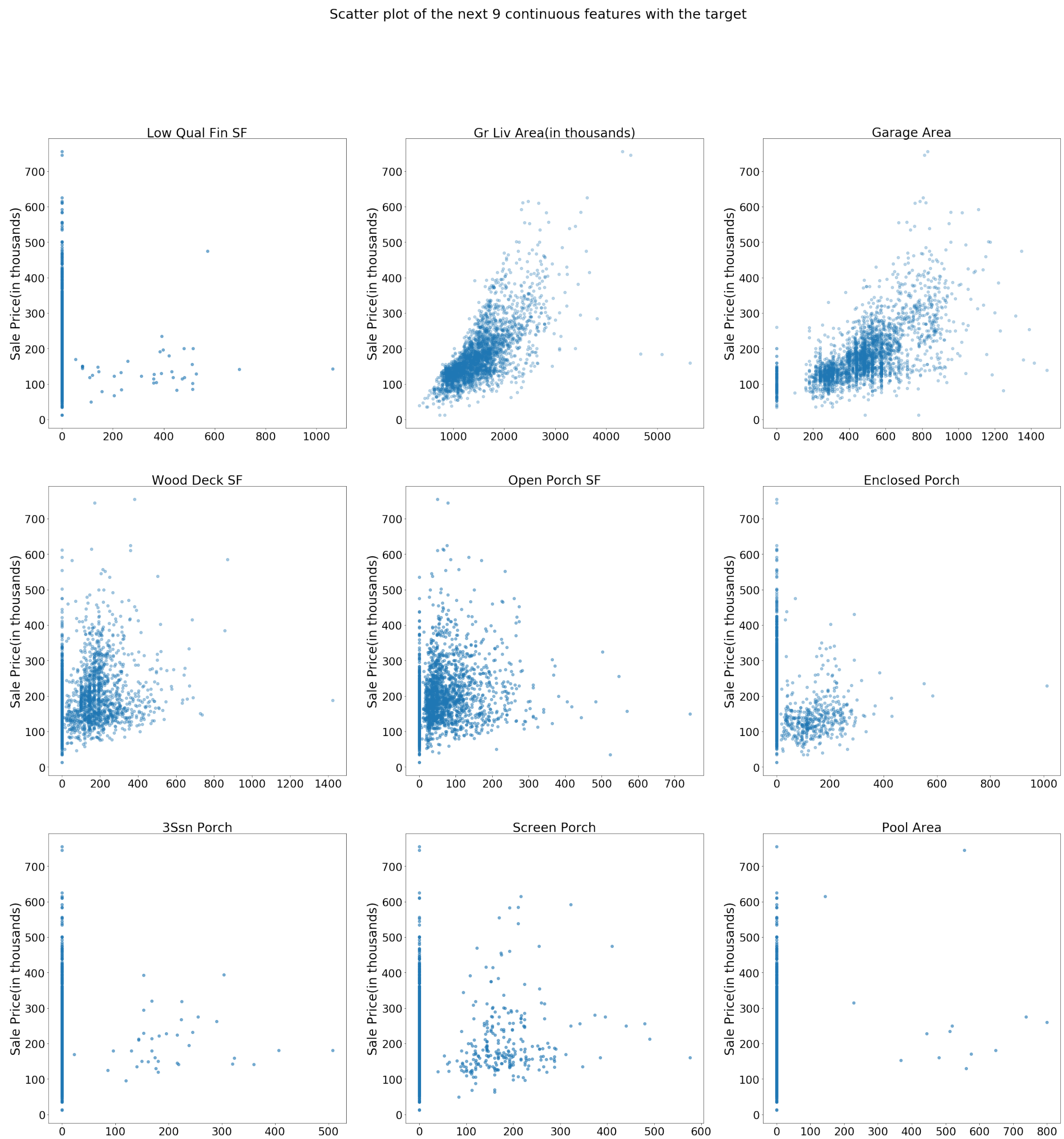
```
In [6]: sp=cont.iloc[:,19]/1000
        fig, ax = plt.subplots(3, 3, figsize=(40, 40))
        _=ax[0,0].scatter(cont.iloc[:, 0], sp,alpha=0.4)
        _=ax[0,0].set_xlabel(cont.columns[0],fontsize=28)
        _=ax[0,0].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[0,0].xaxis.set_label_position('top')

        _=ax[0,1].scatter(cont.iloc[:, 1]/1000, sp,alpha=0.4)
        _=ax[0,1].set_xlabel(cont.columns[1]+"(in thousands)",fontsize=28)
        _=ax[0,1].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[0,1].xaxis.set_label_position('top')

        _=ax[0,2].scatter(cont.iloc[:, 2], sp,alpha=0.4)
        _=ax[0,2].set_xlabel(cont.columns[2],fontsize=28)
        _=ax[0,2].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[0,2].xaxis.set_label_position('top')

        _=ax[1,0].scatter(cont.iloc[:, 3], sp,alpha=0.4)
        _=ax[1,0].set_xlabel(cont.columns[3],fontsize=28)
        _=ax[1,0].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[1,0].xaxis.set_label_position('top')

        _=ax[1,1].scatter(cont.iloc[:, 4], sp,alpha=0.5)
        _=ax[1,1].set_xlabel(cont.columns[4],fontsize=28)
        _=ax[1,1].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[1,1].xaxis.set_label_position('top')

        _=ax[1,2].scatter(cont.iloc[:, 5], sp,alpha=0.3)
        _=ax[1,2].set_xlabel(cont.columns[5],fontsize=28)
        _=ax[1,2].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[1,2].xaxis.set_label_position('top')

        _=ax[2,0].scatter(cont.iloc[:, 6], sp,alpha=0.3)
        _=ax[2,0].set_xlabel(cont.columns[6],fontsize=28)
        _=ax[2,0].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[2,0].xaxis.set_label_position('top')

        _=ax[2,1].scatter(cont.iloc[:, 7], sp,alpha=0.3)
        _=ax[2,1].set_xlabel(cont.columns[7],fontsize=28)
        _=ax[2,1].set_ylabel("Sale Price(in thousands)",fontsize=28)
        ax[2,1].xaxis.set_label_position('top')
```

```
_=ax[2,2].scatter(cont.iloc[:, 8], sp,alpha=0.4)
_=ax[2,2].set_xlabel(cont.columns[8],fontsize=28)
_=ax[2,2].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[2,2].xaxis.set_label_position('top')
_=fig.suptitle("Scatter plot of the first 9 continuous features with
               the target",fontsize=30)
```

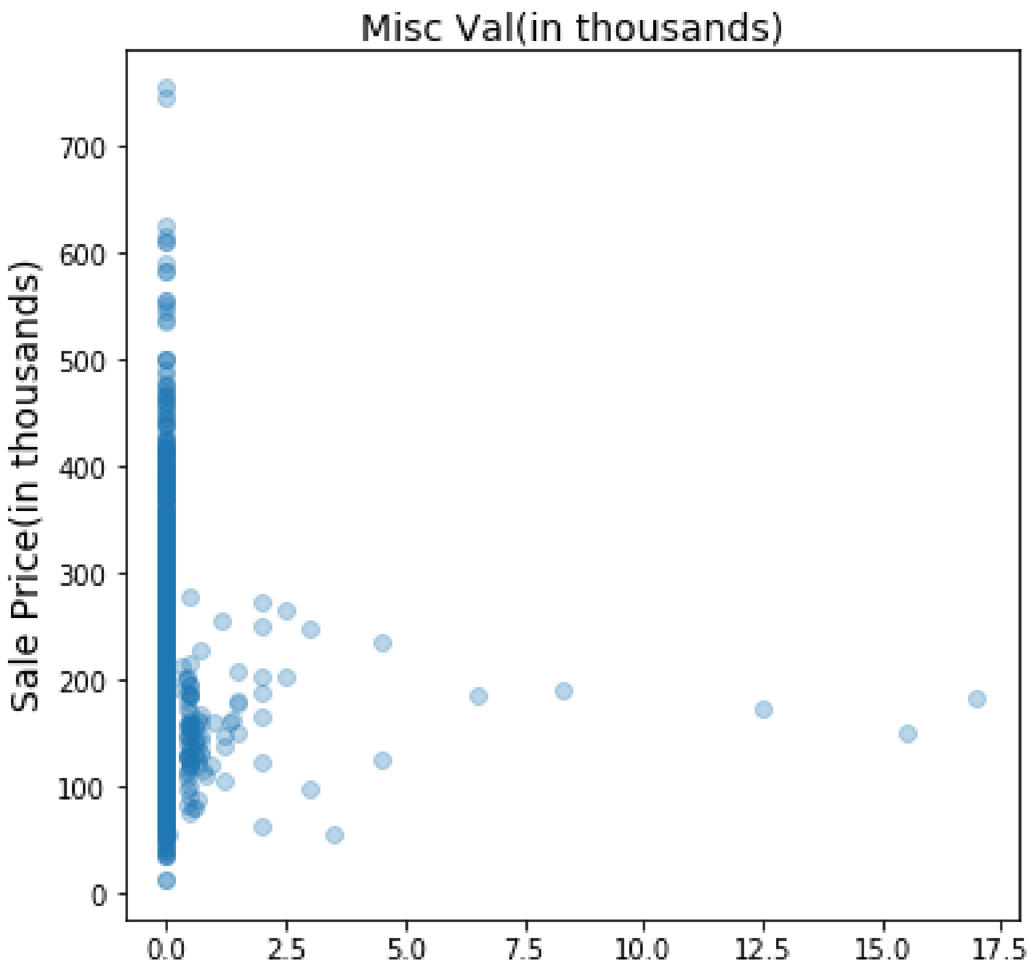



```
In [7]: fig, ax = plt.subplots(3, 3, figsize=(40, 40))
        plt.rc('xtick', labelsize=24)
        plt.rc('ytick', labelsize=24)
        _=ax[0,0].scatter(cont.iloc[:, 9], sp,alpha=0.6)
```

```
_=ax[0,0].set_xlabel(cont.columns[9],fontsize=28)
_=ax[0,0].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[0,0].xaxis.set_label_position('top')

_=ax[0,1].scatter(cont.iloc[:, 10], sp,alpha=0.3)
_=ax[0,1].set_xlabel(cont.columns[10]+"(in thousands)",fontsize=28)
_=ax[0,1].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[0,1].xaxis.set_label_position('top')

_=ax[0,2].scatter(cont.iloc[:, 11], sp,alpha=0.3)
_=ax[0,2].set_xlabel(cont.columns[11],fontsize=28)
_=ax[0,2].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[0,2].xaxis.set_label_position('top')

_=ax[1,0].scatter(cont.iloc[:, 12], sp,alpha=0.4)
_=ax[1,0].set_xlabel(cont.columns[12],fontsize=28)
_=ax[1,0].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[1,0].xaxis.set_label_position('top')

_=ax[1,1].scatter(cont.iloc[:, 13], sp,alpha=0.5)
_=ax[1,1].set_xlabel(cont.columns[13],fontsize=28)
_=ax[1,1].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[1,1].xaxis.set_label_position('top')

_=ax[1,2].scatter(cont.iloc[:, 14], sp,alpha=0.4)
_=ax[1,2].set_xlabel(cont.columns[14],fontsize=28)
_=ax[1,2].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[1,2].xaxis.set_label_position('top')

_=ax[2,0].scatter(cont.iloc[:, 15], sp,alpha=0.6)
_=ax[2,0].set_xlabel(cont.columns[15],fontsize=28)
_=ax[2,0].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[2,0].xaxis.set_label_position('top')

_=ax[2,1].scatter(cont.iloc[:, 16], sp,alpha=0.6)
_=ax[2,1].set_xlabel(cont.columns[16],fontsize=28)
_=ax[2,1].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[2,1].xaxis.set_label_position('top')

_=ax[2,2].scatter(cont.iloc[:, 17], sp,alpha=0.6)
_=ax[2,2].set_xlabel(cont.columns[17],fontsize=28)
_=ax[2,2].set_ylabel("Sale Price(in thousands)",fontsize=28)
ax[2,2].xaxis.set_label_position('top')
_=fig.suptitle("Scatter plot of the next 9 continuous features with
               the target",fontsize=30)
```




```
In [8]: plt.rc('xtick', labelsize=10)
        plt.rc('ytick', labelsize=10)
        fig, ax = plt.subplots(1, 1, figsize=(6,6))
        plt.rc('xtick', labelsize=10)
        plt.rc('ytick', labelsize=10)
        _=ax.scatter(cont.iloc[:, 18]/1000, sp,alpha=0.3)
        _=ax.set_xlabel(cont.columns[18]+"(in thousands)",fontsize=14)
        _=ax.set_ylabel("Sale Price(in thousands)",fontsize=14)
        ax.xaxis.set_label_position('top')
```




# 4 1.3

```
In [9]: dfcat=df.select_dtypes(include=['object'])
        for i in range(43):
            dfcat.iloc[:,i].fillna('NA',inplace=True)
        columns=df.columns[:81].values

In [10]: from sklearn.model_selection import train_test_split,cross_val_score
         from sklearn.preprocessing import OneHotEncoder
         from sklearn.linear_model import LinearRegression
         X_train,X_test,y_train,y_test=train_test_split(df2,df.iloc[:,81])

In [11]: mean_sco=[]
         for i in range(43):
             c=X_train.iloc[:,i].values
             c=c.reshape(-1,1)
```

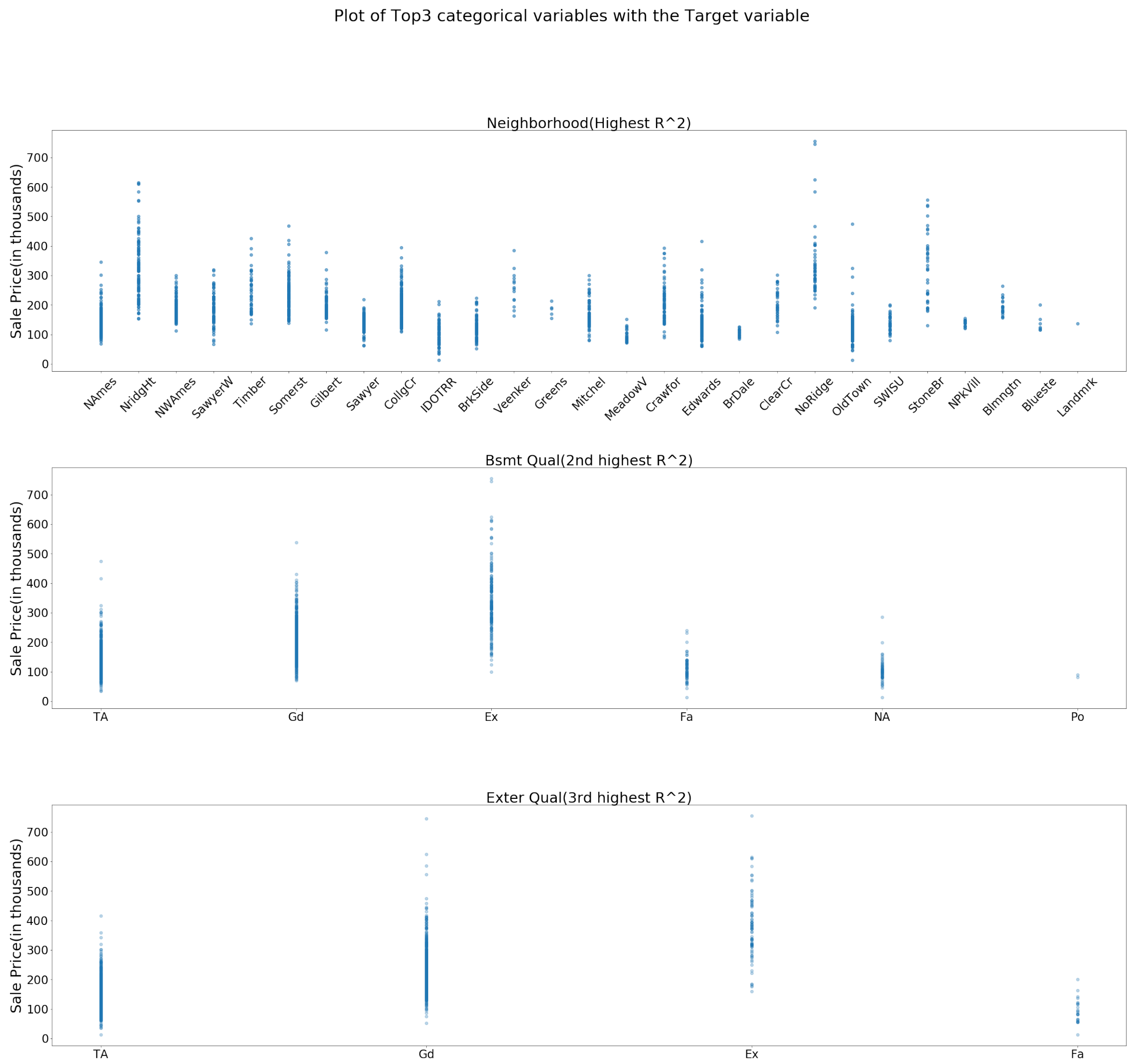
```
        o=OneHotEncoder().fit(c)
        ce1=o.transform(c)
        lr=LinearRegression()
        score = cross_val_score(lr, ce1, y_train, cv=5)
        mean_sc=np.mean(score)
        mean_sco.append(mean_sc)
    np.argsort(mean_sco)
```

Out[11]:
```
array([ 5, 40,  7,  1, 26, 14,  6, 31,  2, 19, 10, 38, 11, 25, 39,  9,  4,
       22, 29, 12, 28, 37, 36, 35,  3, 13,  0, 41, 42, 16, 15, 23, 17, 27,
       24, 33, 20, 32, 34, 30, 18, 21,  8])
```

In [12]:
```
#Top 3 categories with the highest R^2 score are the ones with column number 8,21,18
X_train.columns[[8,21,18]]
```

Out[12]:
```
Index(['Neighborhood', 'Bsmt Qual', 'Exter Qual'], dtype='object')
```

In [13]:
```
plt.rc('xtick', labelsize=24)
plt.rc('ytick', labelsize=24)
fig, ax = plt.subplots(3, 1, figsize=(40, 35))
_=ax[0].scatter(X_train.iloc[:, 8], y_train/1000,alpha=0.6)
_=ax[0].set_xlabel(X_train.columns[8]+"(Highest R^2)",fontsize=30)
_=ax[0].set_ylabel("Sale Price(in thousands)",fontsize=30)
ax[0].xaxis.set_label_position('top')
ax[0].tick_params(axis='x',rotation=45)

_=ax[1].scatter(X_train.iloc[:, 21], y_train/1000,alpha=0.3)
_=ax[1].set_xlabel(X_train.columns[21]+"(2nd highest R^2)",fontsize=30)
_=ax[1].set_ylabel("Sale Price(in thousands)",fontsize=30)
ax[1].xaxis.set_label_position('top')

_=ax[2].scatter(X_train.iloc[:, 18], y_train/1000,alpha=0.3)
_=ax[2].set_xlabel(X_train.columns[18]+"(3rd highest R^2)",fontsize=30)
_=ax[2].set_ylabel("Sale Price(in thousands)",fontsize=30)
ax[2].xaxis.set_label_position('top')
fig.subplots_adjust(hspace=.4)
_=plt.suptitle("Plot of Top3 categorical variables with the
               Target variable",fontsize=34)
```




# 5 1.4

```
In [14]: continuous=df.dtypes!='object'
```

```
In [15]: from sklearn.impute import SimpleImputer
         cont_var=df.loc[:,continuous]
         col=cont_var.columns.values
         for i in range(39):
             cont_var.iloc[:,i].fillna(0,inplace=True)
         imp=SimpleImputer(missing_values=0,strategy="mean")
         arr_imp=imp.fit_transform(cont_var)
         cont_var_imp=pd.DataFrame(arr_imp,columns=col)
```

```
In [20]: from sklearn.compose import make_column_transformer
         from sklearn.pipeline import make_pipeline
```

```
from sklearn.linear_model import Ridge,Lasso,ElasticNet
from sklearn.preprocessing import StandardScaler

preprocess = make_column_transformer(
(OneHotEncoder(handle_unknown='ignore'), categorical.columns.values),
(SimpleImputer(missing_values=0,strategy='mean'), continuous1.columns.values))
model1 = make_pipeline(preprocess, LinearRegression())

model2 = make_pipeline(preprocess, Ridge(alpha=200,max_iter=2000))

model3 = make_pipeline(preprocess, Lasso(alpha=200,max_iter=2000))

model4 = make_pipeline(preprocess, ElasticNet(alpha=200,max_iter=2000))

score1 = cross_val_score(model1, X_train, y_train, cv=5)
mean_lr=np.mean(score1)

score2 = cross_val_score(model2, X_train, y_train, cv=5)
mean_ridge=np.mean(score2)
score3 = cross_val_score(model3, X_train, y_train, cv=5)
mean_lasso=np.mean(score3)
score4 = cross_val_score(model4, X_train, y_train, cv=5)
mean_en=np.mean(score4)
```

In [21]:
```
score1
```

Out[21]:
```
array([0.66899464, 0.50653169, 0.68364334, 0.67498931, 0.6775442 ])
```

In [22]:
```
score2
```

Out[22]:
```
array([ 0.00744402,  0.00544373,  0.00471067, -0.00902518,  0.00643135])
```

In [23]:
```
score3
```

Out[23]:
```
array([0.89339062, 0.76064746, 0.87151614, 0.82075992, 0.90804861])
```

In [24]:
```
score4
```

Out[24]:
```
array([0.74813052, 0.59750146, 0.7566171 , 0.70155237, 0.78792747])
```

In [25]:
```
preprocess = make_column_transformer(
(OneHotEncoder(handle_unknown='ignore'), categorical.columns.values),
(make_pipeline(SimpleImputer(missing_values=0,strategy='mean'),
               StandardScaler()), continuous1.columns.values))

model11 = make_pipeline(preprocess,LinearRegression())
score11 = cross_val_score(model11, X_train, y_train, cv=5)
mean_lr1=np.mean(score11)
```

```
model21 = make_pipeline(preprocess, Ridge(alpha=200,max_iter=2000))
score21 = cross_val_score(model21, X_train, y_train, cv=5)
mean_ridge=np.mean(score21)

model31 = make_pipeline(preprocess, Lasso(alpha=200,max_iter=2000))
score31 = cross_val_score(model31, X_train, y_train, cv=5)
mean_lasso=np.mean(score31)

model41 = make_pipeline(preprocess, ElasticNet(alpha=200,max_iter=2000))
score41 = cross_val_score(model41, X_train, y_train, cv=5)
mean_en=np.mean(score41)
```

In [26]:
```
score11
```

Out[26]: array([0.88836144, 0.7307701 , 0.8647695 , 0.82700163, 0.89168924])

In [27]:
```
score21
```

Out[27]: array([0.88103009, 0.75085059, 0.86362217, 0.79313411, 0.90099089])

In [28]:
```
score31
```

Out[28]: array([0.89630489, 0.76259871, 0.87217128, 0.82421155, 0.90725393])

In [29]:
```
score41
```

Out[29]: array([0.10454429, 0.10648984, 0.11020084, 0.09529783, 0.11880238])

In [ ]:
```
'''
As seen from the scores score1,score2,score3,score4(referring to the scores
obtained without Standard Scaler) and score11,score21,score31,score41
(referring to the scores obtained with Standard Scaler),we see an increase
in the cross val scores after standardising the input.
'''
```

# 6 1.5

In [30]:
```
from sklearn.model_selection import GridSearchCV
model22=make_pipeline(preprocess,Ridge())
param_grid1 = {'ridge__alpha': np.logspace(-3, 3, 13)}
grid = GridSearchCV(model22, param_grid1, cv=5,return_train_score=True)
grid.fit(X_train, y_train)
#scores=cross_val_score(grid,X_train,y_train,cv=10)
#print(grid.best_params_)
#print(grid.best_score_)
grid.cv_results_
d1=grid.cv_results_['mean_train_score']
```

```
In [31]: import warnings
         warnings.filterwarnings("ignore")
         model32=make_pipeline(preprocess,Lasso(normalize=True,max_iter=2000))
         param_grid2 = {'lasso__alpha': np.logspace(-3, 3, 13)}
         grid1 = GridSearchCV(model32, param_grid2, cv=5,return_train_score=True)
         grid1.fit(X_train, y_train)

         #scores=cross_val_score(grid,X_train,y_train,cv=10)
         #print(grid.best_params_)
         #print(grid.best_score_)
         d2=grid1.cv_results_['mean_train_score']
```

```
In [32]: model42=make_pipeline(preprocess,ElasticNet())
         param_grid3 = {'elasticnet__alpha': np.logspace(-4, -1, 10),
                        'elasticnet__l1_ratio': [0.01, .1, .5, .9, .98, 1]}
         grid2 = GridSearchCV(model42, param_grid3, cv=5,return_train_score=True)
         grid2.fit(X_train, y_train)
         d3=grid2.cv_results_['mean_train_score']
```

```
In [33]: print(param_grid1)
         print(param_grid2)
         print(param_grid3)
```

```
{'ridge__alpha': array([1.00000000e-03, 3.16227766e-03, 1.00000000e-02, 3.16227766e-02,
       1.00000000e-01, 3.16227766e-01, 1.00000000e+00, 3.16227766e+00,
       1.00000000e+01, 3.16227766e+01, 1.00000000e+02, 3.16227766e+02,
       1.00000000e+03])}
{'lasso__alpha': array([1.00000000e-03, 3.16227766e-03, 1.00000000e-02, 3.16227766e-02,
       1.00000000e-01, 3.16227766e-01, 1.00000000e+00, 3.16227766e+00,
       1.00000000e+01, 3.16227766e+01, 1.00000000e+02, 3.16227766e+02,
       1.00000000e+03])}
{'elasticnet__alpha': array([0.0001    , 0.00021544, 0.00046416, 0.001     , 0.00215443,
       0.00464159, 0.01      , 0.02154435, 0.04641589, 0.1       ]), 'elasticnet__l1_ratio': [(
```

```
In [34]: d1
```

```
Out[34]: array([0.93075573, 0.93073307, 0.93072716, 0.93061002, 0.93032504,
                0.9294325 , 0.92701803, 0.92184429, 0.91395393, 0.90396874,
                0.89114889, 0.87344618, 0.84429393])
```

```
In [35]: d2
```

```
Out[35]: array([0.94200396, 0.94200396, 0.9420039 , 0.94200327, 0.94199752,
                0.94194815, 0.94167768, 0.94058154, 0.93584453, 0.92270501,
                0.87328846, 0.7578804 , 0.38231203])
```

```
In [86]: d3
```

```
Out[86]: array([0.94025852, 0.94044849, 0.94128573, 0.94194583, 0.94200036,
                0.94200396, 0.9380305 , 0.93838049, 0.9400756 , 0.94180116,
                0.94198995, 0.94200396, 0.93438077, 0.93491574, 0.93770186,
                0.94135726, 0.94195258, 0.94200396, 0.9292868 , 0.92998761,
                0.93388758, 0.94023733, 0.94182372, 0.94200396, 0.92318979,
                0.92397729, 0.92865248, 0.93799195, 0.94142238, 0.94200396,
                0.91672449, 0.91753588, 0.92248767, 0.93432222, 0.94038792,
                0.94200396, 0.91002801, 0.91088328, 0.91600268, 0.92921028,
                0.93826736, 0.94200396, 0.90268192, 0.90364893, 0.90926088,
                0.92310321, 0.93474034, 0.94200395, 0.89410387, 0.89525647,
                0.90180739, 0.9166329 , 0.92975319, 0.94200388, 0.88363227,
                0.88506409, 0.89305544, 0.90992724, 0.92370528, 0.94200359])
```

```
In [90]: grid.cv_results_['mean_test_score']
```

```
Out[90]: array([0.84422755, 0.84424895, 0.84424079, 0.8443346 , 0.84459463,
                0.84526308, 0.84681329, 0.84906673, 0.85040216, 0.84852077,
                0.84279978, 0.83395932, 0.81738957])
```

```
In [91]: grid1.cv_results_['mean_test_score']
```

```
Out[91]: array([0.84414694, 0.84427834, 0.8446325 , 0.84531662, 0.8456647 ,
                0.84655   , 0.84926072, 0.85484958, 0.86218027, 0.86421741,
                0.8351336 , 0.75326036, 0.38291382])
```

```
In [92]: grid2.cv_results_['mean_test_score']
```

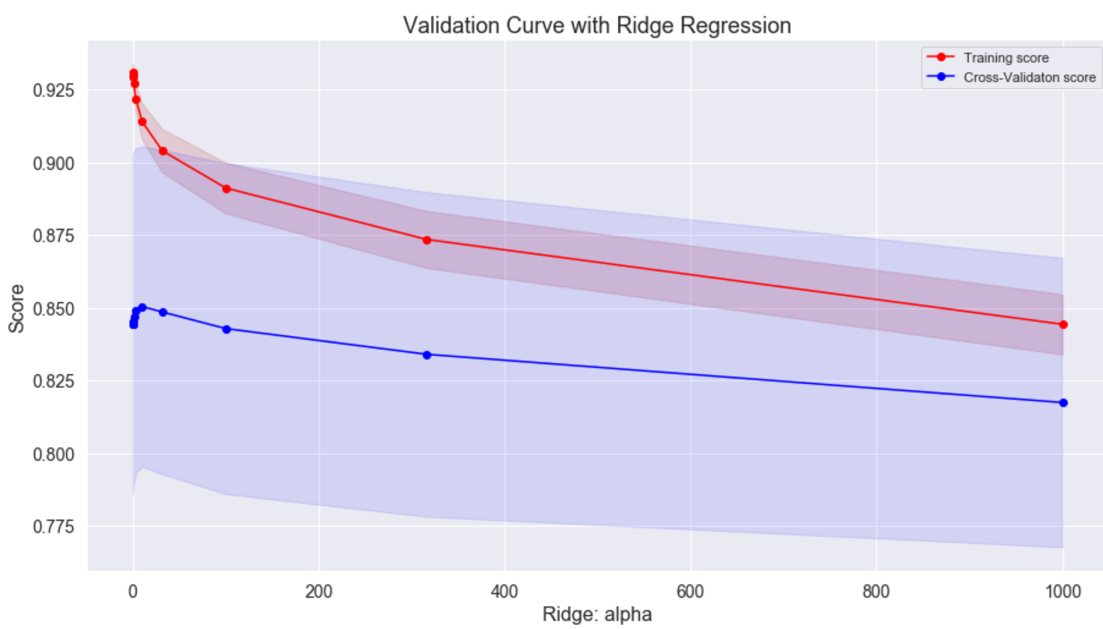
```
Out[92]: array([0.84243674, 0.84234088, 0.84202884, 0.84178813, 0.84309327,
                0.84405889, 0.84377547, 0.8435569 , 0.8425345 , 0.84192576,
                0.84220036, 0.84406259, 0.84584348, 0.84557393, 0.84397956,
                0.8420141 , 0.84178901, 0.84407034, 0.8478463 , 0.8476223 ,
                0.84608101, 0.84244832, 0.84191326, 0.84408617, 0.84934461,
                0.84918901, 0.84803825, 0.84380034, 0.84200288, 0.84408994,
                0.85018839, 0.85013245, 0.8494751 , 0.84587346, 0.84237256,
                0.84409689, 0.84988714, 0.85000903, 0.85022402, 0.84787278,
                0.8436324 , 0.84410014, 0.84789637, 0.8482422 , 0.84975749,
                0.84936462, 0.84566951, 0.84406888, 0.84418357, 0.84472768,
                0.84756791, 0.85019895, 0.84771374, 0.84384217, 0.83904403,
                0.83975531, 0.84368333, 0.84987903, 0.84926325, 0.84383472])
```

```
In [ ]: '''
        The mean_train scores are high and the mean_test_scores are around 0.85
        for all the models.
        The results have improved significantly for theElasticNet model.
        '''
```

```
In [36]: t=grid2.cv_results_['params']
         ea=[0]*60
         el=[0]*60
```

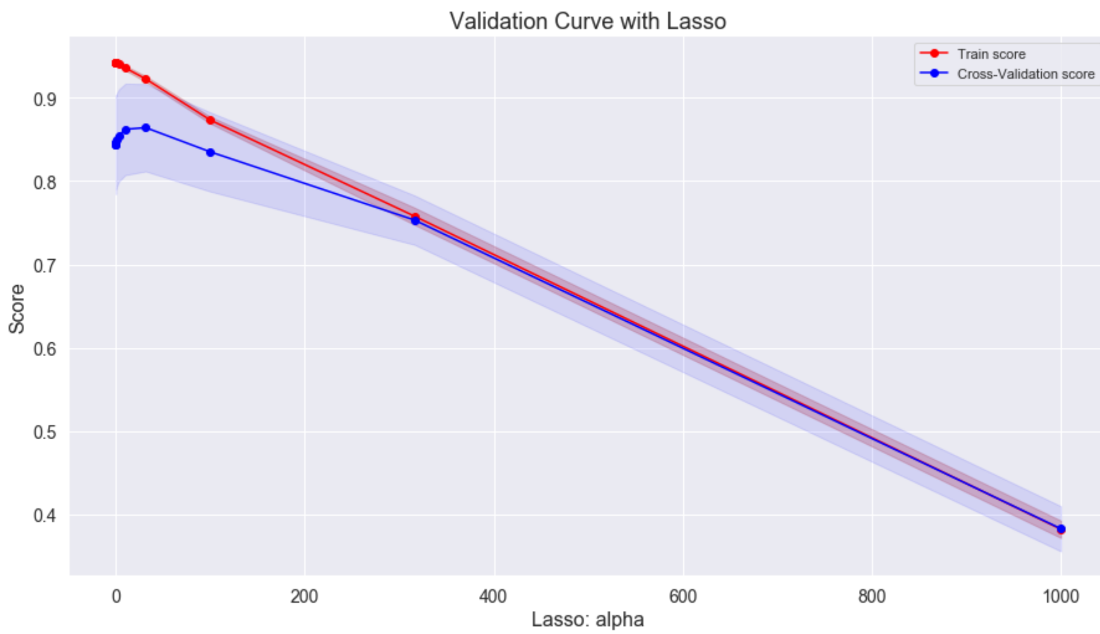
```
    for i in range(60):
        ea[i]=t[i]['elasticnet__alpha']
        el[i]=t[i]['elasticnet__l1_ratio']
```

```
In [109]: plt.figure(figsize=(15,8))
          plt.rc('xtick',labelsize=14)
          plt.rc('ytick',labelsize=14)
          std_score=grid.cv_results_['std_train_score']
          mean_test_score=grid.cv_results_['mean_test_score']
          std_test_score=grid.cv_results_['std_test_score']
          _=plt.plot(param_grid1['ridge__alpha'],d1,marker='o',color='red',
                     label="Training score")
          _=plt.fill_between(param_grid1['ridge__alpha'],d1-std_score,
                             d1+ std_score,alpha=0.2,color="r")
          _=plt.xlabel("Ridge: alpha",fontsize=16)
          _=plt.plot(param_grid1['ridge__alpha'],mean_test_score,marker='o',
                     color='blue',label="Cross-Validaton score")
          _=plt.fill_between(param_grid1['ridge__alpha'],mean_test_score-std_test_score,
                             mean_test_score+std_test_score,alpha=0.1,color="blue")
          _=plt.ylabel("Score",fontsize=16)
          _=plt.title("Validation Curve with Ridge Regression",fontsize=18)
          _=plt.legend()
```

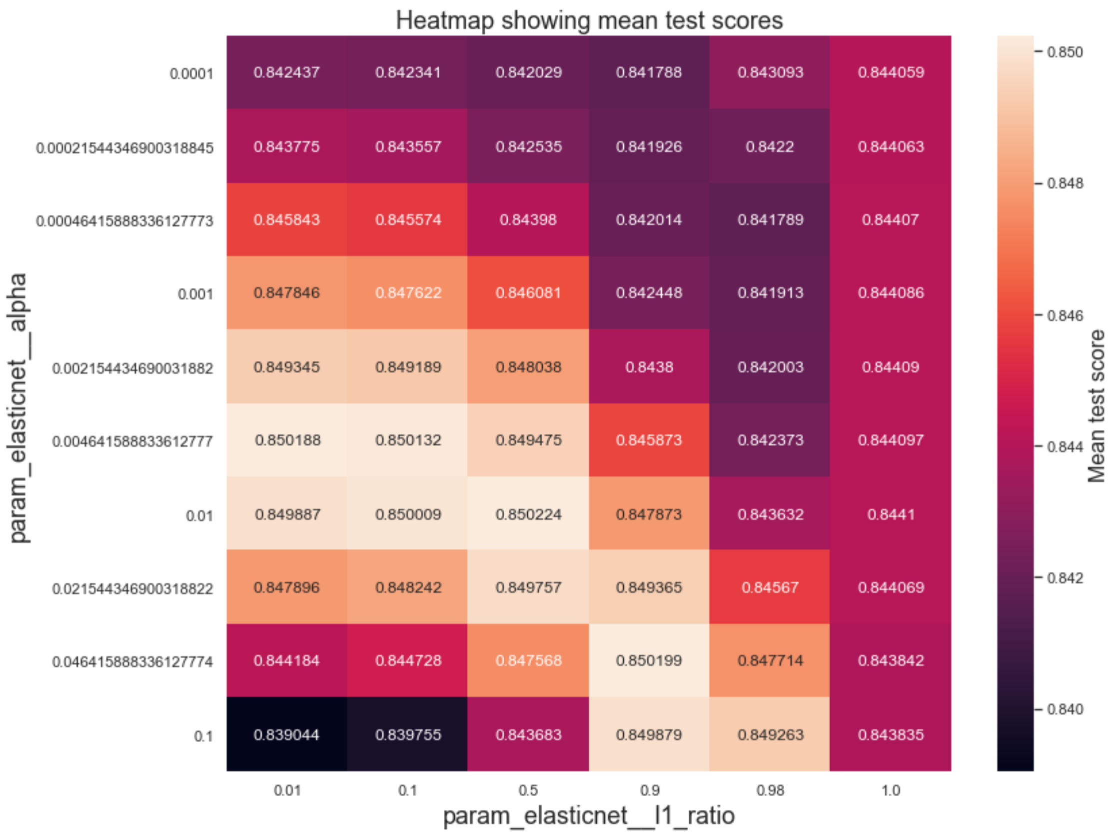



```
In [107]: plt.figure(figsize=(15,8))
          plt.rc('xtick',labelsize=14)
          plt.rc('ytick',labelsize=14)
          std_score=grid1.cv_results_['std_train_score']
```

```
mean_test_score=grid1.cv_results_['mean_test_score']
std_test_score=grid1.cv_results_['std_test_score']
_=plt.plot(param_grid2['lasso__alpha'],d2,marker='o',color='red',
           label="Train score")
_=plt.fill_between(param_grid2['lasso__alpha'],d2-std_score,
                   d2+ std_score,alpha=0.3,color="r")
_=plt.xlabel("Lasso: alpha",fontsize=16)
_=plt.plot(param_grid2['lasso__alpha'],mean_test_score,marker='o',
           color='blue',label="Cross-Validation score")
_=plt.fill_between(param_grid2['lasso__alpha'],mean_test_score-std_test_score,
                   mean_test_score+ std_test_score,alpha=0.1,color="blue")
_=plt.ylabel("Score",fontsize=16)
_=plt.title("Validation Curve with Lasso",fontsize=18)
_=plt.legend()
```




In [73]:
```
import seaborn as sns
res = pd.pivot_table(pd.DataFrame(grid2.cv_results_),
    values='mean_test_score', index='param_elasticnet__alpha',
                     columns='param_elasticnet__l1_ratio')

sns.set(rc={'figure.figsize':(12,10),"axes.labelsize":18})
ax=sns.heatmap(res,annot=True,fmt='g',cbar_kws={'label':
                                               'Mean test score'})
ax.figure.axes[-1].yaxis.label.set_size(16)
_=plt.title('Heatmap showing mean test scores',fontsize=18)
```




# 7 1.6

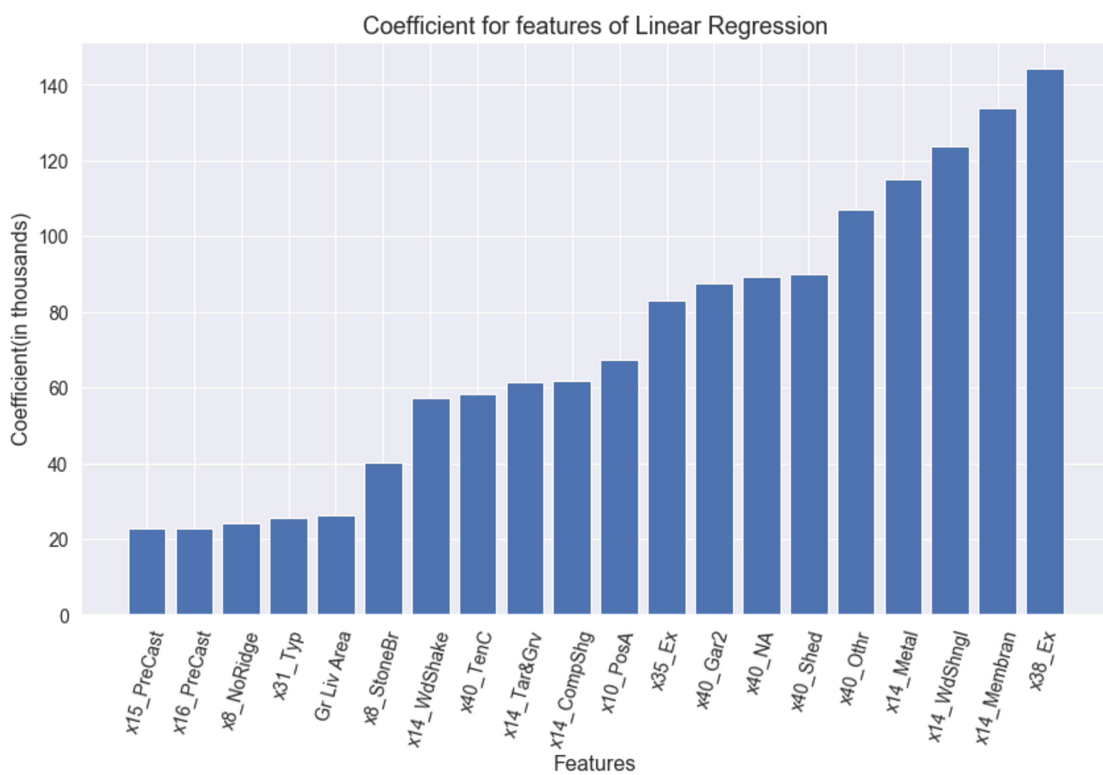
```
In [74]: print(grid.best_params_)
         print(grid1.best_params_)
         print(grid2.best_params_)
```

```
{'ridge__alpha': 10.0}
{'lasso__alpha': 31.622776601683793}
{'elasticnet__alpha': 0.01, 'elasticnet__l1_ratio': 0.5}
```

```
In [75]: cate=X_train.iloc[:,:43]
         ohe1=OneHotEncoder().fit(cate)
         ce11=ohe1.transform(cate)
         ce11.shape
         x=ohe1.get_feature_names()
```

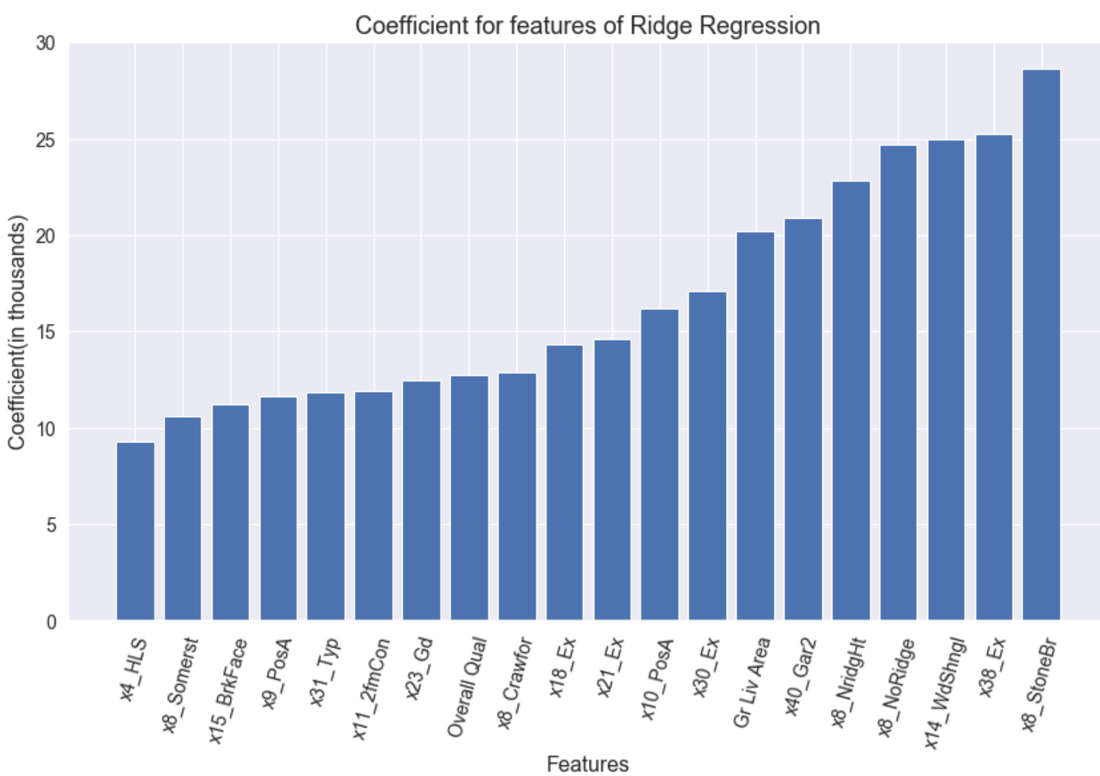
```
In [102]: model11 = make_pipeline(preprocess,LinearRegression())
          lr16=model11.fit(X_train, y_train)
          l=lr16.steps[1][1]
          l=l.coef_
```

```
ind=sorted(range(len(l)), key=lambda i: l[i])[-20:]
l[ind]
cat_names=[None]*20
for i in range(20):
    y=ind[i]
    if y>279:
        cat_names[i]=continuous1.columns[y-280]
    else:
        cat_names[i]=x[y]
plt.rc('xtick',labelsize=14)
plt.rc('ytick',labelsize=14)
fig,ax=plt.subplots(1,1,figsize=(14,8))
_=ax.bar(cat_names,l[ind]/1000)
_=ax.set_xlabel('Features',fontsize=16)
_=ax.set_ylabel('Coefficient(in thousands)',fontsize=16)
_=ax.tick_params(axis='x',rotation=75)
_=plt.title('Coefficient for features of Linear Regression',
            fontsize=18)
```



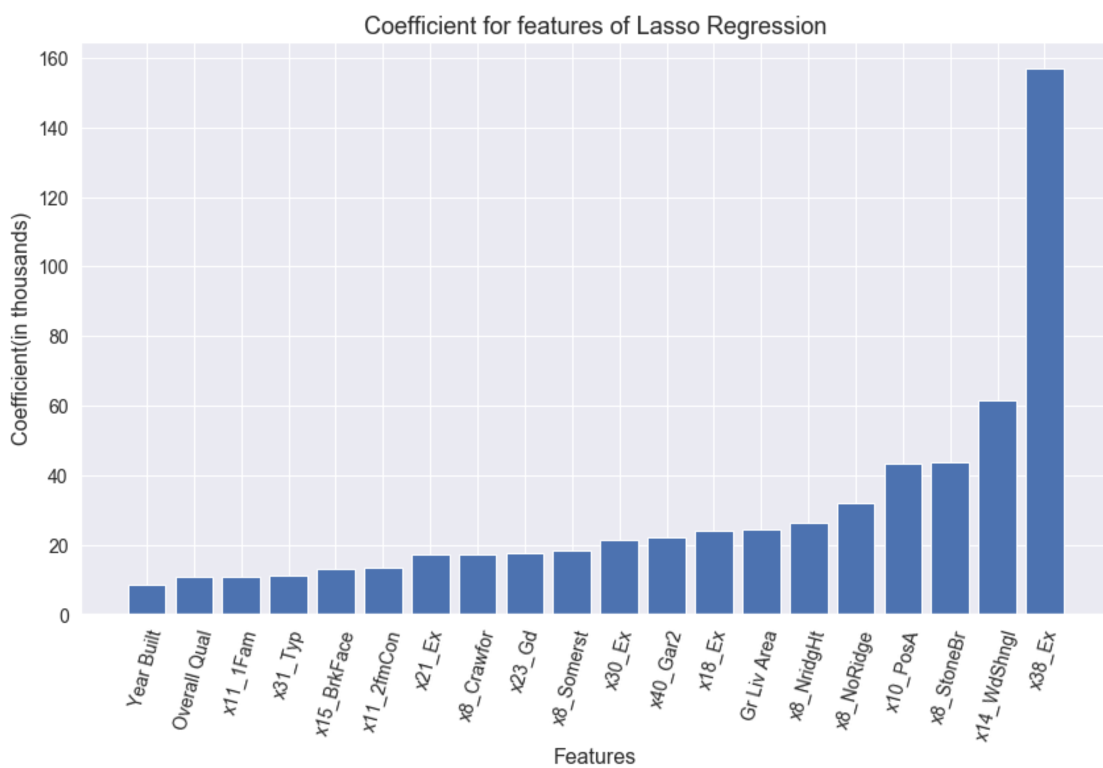

In [103]:
```
new_ridge=Ridge(alpha=grid.best_params_['ridge__alpha'],
                max_iter=2000)
model22=make_pipeline(preprocess,new_ridge)
rr=model22.fit(X_train,y_train)
```

```
r=rr.steps[1][1]
r=r.coef_
ind1=sorted(range(len(r)), key=lambda i: r[i])[-20:]
r[ind1]
cat_names=[None]*20
for i in range(20):
    y=ind1[i]
    if y>279:
        cat_names[i]=continuous1.columns[y-280]
    else:
        cat_names[i]=x[y]
fig,ax=plt.subplots(1,1,figsize=(14,8))
_=ax.bar(cat_names,r[ind1]/1000)
_=ax.set_xlabel('Features',fontsize=16)
_=ax.set_ylabel('Coefficient(in thousands)',fontsize=16)
_=ax.tick_params(axis='x',rotation=75)
_=plt.title('Coefficient for features of Ridge Regression',fontsize=18)
```



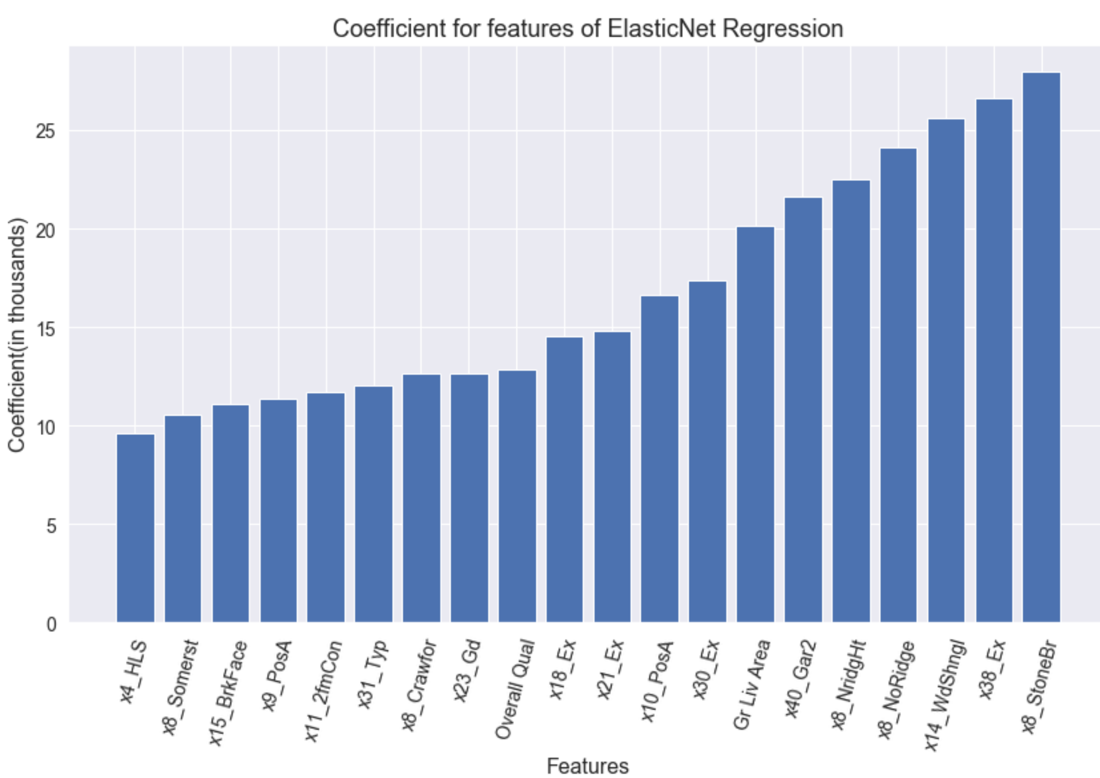

```
In [104]: new_lasso=Lasso(alpha=grid1.best_params_['lasso__alpha'],
                          max_iter=2000)
          model32=make_pipeline(preprocess,new_lasso)
          lar=model32.fit(X_train,y_train)
          la=lar.steps[1][1]
```

```
la=la.coef_
ind2=sorted(range(len(la)), key=lambda i: la[i])[-20:]
la[ind2]
cat_names=[None]*20
for i in range(20):
    y=ind2[i]
    if y>279:
        cat_names[i]=continuous1.columns[y-280]
    else:
        cat_names[i]=x[y]
plt.rc('xtick',labelsize=14)
plt.rc('ytick',labelsize=14)
fig,ax=plt.subplots(1,1,figsize=(14,8))
_=ax.bar(cat_names,la[ind2]/1000)
_=ax.set_xlabel('Features',fontsize=16)
_=ax.set_ylabel('Coefficient(in thousands)',fontsize=16)
_=ax.tick_params(axis='x',rotation=75)
_=plt.title('Coefficient for features of Lasso Regression',fontsize=18)
```




In [105]:
```
new_en=ElasticNet(alpha=grid2.best_params_['elasticnet__alpha'],
        l1_ratio=grid2.best_params_['elasticnet__l1_ratio'],
        max_iter=2000)
model42=make_pipeline(preprocess,new_en)
```

```
enr=model42.fit(X_train,y_train)
en=enr.steps[1][1]
en=en.coef_
ind3=sorted(range(len(en)), key=lambda i: en[i])[-20:]
en[ind3]
cat_names=[None]*20
for i in range(20):
    y=ind3[i]
    if y>279:
        cat_names[i]=continuous1.columns[y-280]
    else:
        cat_names[i]=x[y]
fig,ax=plt.subplots(1,1,figsize=(14,8))
_=ax.bar(cat_names,en[ind3]/1000)
_=ax.set_xlabel('Features',fontsize=16)
_=ax.set_ylabel('Coefficient(in thousands)',fontsize=16)
_=ax.tick_params(axis='x',rotation=75)
_=plt.title('Coefficient for features of ElasticNet Regression',
            fontsize=18)
```




In [550]:
```
'''
Note: The coefficients are so large because the input is scaled
while the target is not.
```

```
Yes,they agree on which features are important. We previously found
that the top 3 categories witht the highest value of R^2 were columns
8,21,18. Atleast one category of each of these columns appears in the
plot of the 20 highest coefficients of the model.Each of these models
shows column 8 to the most important feature because greater number
of categories of this feature appear in the plots above.
'''
```