

Overview

- Data Science Problem: How can one predict future sale prices based upon housing data in Ames, Iowa?
- Using model building techniques and some experimental feature selection, the problem has begun to have been solved

Initial Data Cleaning

- So many ordinal variables
- Numeric values: good analysis

```
Exter dict = {'Po': 0, 'Fa': 1, 'TA': 2, 'Gd': 3, 'Ex': 4}
 2 #First set of replacements
 3 train['Exter Qual'].replace(Exter dict, inplace = True)
 4 train['Exter Cond'].replace(Exter dict, inplace = True)
 5 train['Heating QC'].replace(Exter dict, inplace = True)
    train['Kitchen Qual'].replace(Exter dict, inplace = True)
    Bsmt dict = {'NA': 0, 'Po': 1, 'Fa': 2, 'TA': 3, 'Gd': 4, 'Ex': 5}
    #Second set
10 train['Bsmt Qual'].replace(Bsmt dict, inplace = True)
11 train['Bsmt Cond'].replace(Bsmt dict, inplace = True)
12 train['Fireplace Qu'].replace(Bsmt dict, inplace = True)
13 train['Garage Qual'].replace(Bsmt dict, inplace = True)
14 train['Garage Cond'].replace(Bsmt dict, inplace = True)
15 train['Pool QC'].replace(Bsmt dict, inplace = True)
   Bsmt_ex_dict = {'NA': 0, 'No': 1, 'Mn': 2, 'Av': 3, 'Gd': 4}
18 #First special case set
   train['Bsmt Exposure'].replace(Bsmt ex dict, inplace = True)
    Func_dict = {'Sal': 0, 'Sev': 1, 'Maj2': 2, 'Maj1': 3,
                 'Mod': 4, 'Min2': 5, 'Min1': 6, 'Typ': 7}
    #Second special case
24 train['Functional'].replace(Func_dict, inplace = True)
25
26 Fence dict = {'NA': 0, 'MnWw': 1, 'GdWo': 2, 'MnPrv': 3, 'GdPrv': 4}
28 train['Fence'].replace(Fence dict, inplace = True)
30 Lot_dict = {'IR3': 0, 'IR2': 1, 'IR1': 2, 'Reg': 3}
31 #Fourth
32 train['Lot Shape'].replace(Lot_dict, inplace = True)
```

Feature Engineering

- Garage and Basement
- Interesting interactions
- Few dummies(Neighborhood, House style, etc.)

Initial Feature Selection

- Slice from a larger Heatmap
- Some high interactions
- Some trouble finding the right mix

Functional -	0.13
Screen Porch -	
Bedroom AbvGr -	
Neighborhood_Somerst -	
Exterior 1st CemntBd -	
Lot Frontage -	
Bsmt Unf SF -	
House Style_2Story -	
Bsmt Cond -	
MS Zoning_RL -	
2nd Flr SF -	
Neighborhood_StoneBr -	
Garage Yr Blt -	0.26
Neighborhood_NoRidge -	272
Garage Cond -	
Central Air_Y - Half Bath -	
Bsmt Full Bath - Garage Qual -	
Lot Area -	
Wood Deck SF -	
Open Porch SF -	
Exterior 1st_VinylSd - BsmtFin SF 1 -	
Bsmt Exposure -	0.42
Neighborhood NridgHt -	27.12
Heating QC -	0.45
Fireplaces -	
Mas Vnr Area -	0.5
TotRms AbvGrd -	0.5
Foundation PConc -	0.53
Full Bath -	
Fireplace Qu -	
Year Remod/Add -	
Year Built -	0.57
Bsmt Qual -	0.61
TotRms GrLiv -	0.62
1st Fir SF -	0.62
Total Bsmt SF -	0.63
Bsmt Total -	0.64
Garage Cars -	0.65
Garage Area -	0.65
Kitchen Oual -	0.69
Gr Liv Area -	0.7
Garage Total -	0.71
Exter Qual -	0.71
Overall Qual -	0.8
Exter Overall -	0.82
SalePrice -	1

"The Experiment"

```
1 | ideal r2 = 0
 2 r2 features = []
 3 ideal rmse = 999999999
4 rmse features = []
5 v = train['SalePrice']
 6 feature length = len(features init)#Every possible numeric feature
7 print('start')
8 for n in range(20, 50): # I didn't actually want to run it for 24 hours.
       counter = 1 #To be comprehensive, the range would be 1 to feature length. Tuning.
9
       while counter <= (feature length * n):
10
11
            unique features = []
           features = features init.sample(n) #Random sample of n features
12
13
            X = train[features]
14
           X_train, X_test, y_train, y_test = train_test_split(X,y,random_state = 42)
15
16
            model ridge = RidgeCV(cv = 5)
17
            ss = StandardScaler() #Split and fit
18
            ss.fit(X train)
19
            X train sc = ss.transform(X train)
20
            X test sc = ss.transform(X test)
21
            model ridge.fit(X train sc, y train)
22
23
            r2 temp = model ridge.score(X test sc, y test) #R2 value
           rmse temp = RT MSE(model ridge, X test sc, y test) #Root mean squared error
24
25
           if r2 temp > ideal r2: #Tested two different values for the minimum
26
27
                r2 features = features
28
               ideal r2 = r2 temp
29
            if rmse temp < ideal rmse:
30
                rmse features = features
31
               ideal rmse = rmse temp
32
33
            if not features in unique features: #Makes sure every combination is tried
34
                unique features.append(features)
35
                counter += 1
36
37
       print(n) #Track progress
   print(ideal r2)
39 print(ideal rmse)
```

Results

- The wonders of black box programming.
- Other metrics at play here
- R² ended up being around 90%(with log transformation)

SalePrice -	1
Exter_Overall =	0.82
Overall Qual -	0.8
Exter Qual -	0.71
Gr Liv Area -	
Kitchen Qual -	0.69
Garage Area -	
Bsmt_Total -	0.64
Total Bsmt SF -	0.63
TotRms_GrLiv	0.62
Bsmt Qual -	0.61
Foundation_PConc	0.53
Mas Vnr Area	0.5 0.45
Neighborhood_NridgHt -	
Open Porch SF - Lot Area -	0.33
Bsmt Full Bath -	
Neighborhood StoneBr	0.26
MS Zoning RL -	0.23
House Style_2Story -	
Exterior 1st CemntBd	0.17
Functional -	0.13
Neighborhood Veenker -	0.083
Neighborhood_Crawfor	0.058
3Ssn Porch -	0.049
Exterior 1st_BrkFace -	0.026
Neighborhood_Blmngtn =	0.025
Neighborhood_Gilbert -	0.024
BsmtFin SF 2 -	
Misc Val -	-0.0074
Alley_Pave -	-0.015
MS Zoning_I (all) -	-0.035
House Style_SLvl	
Bsmt Half Bath -	-0.045 -0.064
House Style_SFoyer =	-0.087
MS SubClass -	-0.095
Neighborhood_BrDale - Overall Cond -	-0.097
Foundation Slab	-0.12
Exterior 1st AsbShng -	
Neighborhood Sawyer	-0.13
Neighborhood BrkSide	-0.13
Exterior 1st MetalSd -	-0.15
Alley Grvl	-0.16
Foundation_BrkTil -	-0.23
MS Zoning_RM -	-0.28

Foundation CBlock

Conclusions

- Points for consideration:
 - Strong foundation
 - Large area
 - Right neighborhood
 - High quality
- The best solution may not be readily apparent.