Analysis of Housing Prices in Iowa

Nico A. Espinosa Dice

August, 2017

```
In [113]: #ipython nbconvert FinalProject.ipynb --to slides
          #ipython nbconvert FinalProject.ipynb --to slides --post serve
```

1 Housing Prices

2 Problem

Predict housing prices based on a number of descriptive variables about a house, as well as the final sale price.

By understanding what features of a house lead to a greater increase in final sale price, this can lead to smarter decision making: - Building a house -> where/what qualities to incorporate - Potentially smarter renovations -> improving what parts of the house will lead to the greatest profit - More accurate understanding of what your house will sell for

2.1 Hypothesis

- Size of the house will have the largest impact on the final sale price

3 Data

- Accessed from Kaggle competition
- Approximately 80 columns, or descriptive variables about houses in Iowa
  - A bit over half categorical
    - Categorical variables range from 4 to 15 categories
- Target variable will be the final sale price of the house

```
In [114]: %matplotlib inline
          import matplotlib.pyplot as plt
          import pandas as pd
          import statsmodels.api as sm
          import pylab as pl
          import numpy as np
```

```
from sklearn import metrics
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.model_selection import train_test_split
from sklearn.ensemble import *
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import seaborn as sns
from sklearn import preprocessing
import math
from sklearn.ensemble import RandomForestRegressor
from sklearn import model_selection
```

```
In [115]: prices = pd.read_csv('housing_price_train.csv')

          target = 'SalePrice'

          predictors = []
          for i in prices.columns:
              if i != "SalePrice" and i != "Id":
                  predictors.append(i)
```

```
In [116]: prices.head()
```

```
Out[116]:    Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
          0   1          60       RL         65.0     8450   Pave   NaN      Reg
          1   2          20       RL         80.0     9600   Pave   NaN      Reg
          2   3          60       RL         68.0    11250   Pave   NaN      IR1
          3   4          70       RL         60.0     9550   Pave   NaN      IR1
          4   5          60       RL         84.0    14260   Pave   NaN      IR1

            LandContour Utilities    ...     PoolArea PoolQC Fence MiscFeature MiscVal  \
          0         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          1         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          2         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          3         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          4         Lvl    AllPub    ...            0    NaN   NaN         NaN       0

            MoSold YrSold  SaleType  SaleCondition  SalePrice
          0      2   2008        WD         Normal     208500
          1      5   2007        WD         Normal     181500
          2      9   2008        WD         Normal     223500
          3      2   2006        WD        Abnorml     140000
          4     12   2008        WD         Normal     250000

          [5 rows x 81 columns]
```

MSSubClass: Identifies the type of dwelling involved in the sale.

```
 20  1-STORY 1946 & NEWER ALL STYLES
 30  1-STORY 1945 & OLDER
 40  1-STORY W/FINISHED ATTIC ALL AGES
 45  1-1/2 STORY - UNFINISHED ALL AGES
 50  1-1/2 STORY FINISHED ALL AGES
 60  2-STORY 1946 & NEWER
 70  2-STORY 1945 & OLDER
 75  2-1/2 STORY ALL AGES
 80  SPLIT OR MULTI-LEVEL
 85  SPLIT FOYER
 90  DUPLEX - ALL STYLES AND AGES
120  1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150  1-1/2 STORY PUD - ALL AGES
160  2-STORY PUD - 1946 & NEWER
180  PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190  2 FAMILY CONVERSION - ALL STYLES AND AGES
```

MSZoning: Identifies the general zoning classification of the sale.

```
A    Agriculture
C    Commercial
FV   Floating Village Residential
I    Industrial
RH   Residential High Density
RL   Residential Low Density
RP   Residential Low Density Park
RM   Residential Medium Density
```

LotFrontage: Linear feet of street connected to property
LotArea: Lot size in square feet
Street: Type of road access to property

```
Grvl Gravel
Pave Paved
```

Alley: Type of alley access to property

```
Grvl Gravel
Pave Paved
NA   No alley access
```

```
In [117]: print prices.dtypes
```

```
Id                  int64
MSSubClass          int64
MSZoning           object
LotFrontage       float64
LotArea             int64
```

```
Street            object
Alley             object
LotShape          object
LandContour       object
Utilities         object
LotConfig         object
LandSlope         object
Neighborhood      object
Condition1        object
Condition2        object
BldgType          object
HouseStyle        object
OverallQual        int64
OverallCond        int64
YearBuilt          int64
YearRemodAdd       int64
RoofStyle         object
RoofMatl          object
Exterior1st       object
Exterior2nd       object
MasVnrType        object
MasVnrArea       float64
ExterQual         object
ExterCond         object
Foundation        object
                  ...
BedroomAbvGr       int64
KitchenAbvGr       int64
KitchenQual       object
TotRmsAbvGrd       int64
Functional        object
Fireplaces         int64
FireplaceQu       object
GarageType        object
GarageYrBlt      float64
GarageFinish      object
GarageCars         int64
GarageArea         int64
GarageQual        object
GarageCond        object
PavedDrive        object
WoodDeckSF         int64
OpenPorchSF        int64
EnclosedPorch      int64
3SsnPorch          int64
ScreenPorch        int64
PoolArea           int64
PoolQC            object
```

```
Fence            object
MiscFeature      object
MiscVal           int64
MoSold            int64
YrSold            int64
SaleType         object
SaleCondition    object
SalePrice         int64
Length: 81, dtype: object
```

4 Data Cleaning

- Many null values, particularly in categorical variables
  - Null values actually had meaning in categorical variables:
    - NA meant that feature did not exist in the house
      - Replaced null with None (categorical variable)
  - Null values in numerical variables were replaced with 0
    - This was done when the feature did not exist
      - Ex. Pool size -> 0 if there is no pool

```
In [118]: prices.head()
```

```
Out[118]:    Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
          0   1          60       RL         65.0     8450   Pave   NaN      Reg
          1   2          20       RL         80.0     9600   Pave   NaN      Reg
          2   3          60       RL         68.0    11250   Pave   NaN      IR1
          3   4          70       RL         60.0     9550   Pave   NaN      IR1
          4   5          60       RL         84.0    14260   Pave   NaN      IR1

            LandContour Utilities    ...     PoolArea PoolQC Fence MiscFeature MiscVal  \
          0         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          1         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          2         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          3         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
          4         Lvl    AllPub    ...            0    NaN   NaN         NaN       0

            MoSold YrSold  SaleType  SaleCondition  SalePrice
          0      2   2008        WD         Normal     208500
          1      5   2007        WD         Normal     181500
          2      9   2008        WD         Normal     223500
          3      2   2006        WD        Abnorml     140000
          4     12   2008        WD         Normal     250000

          [5 rows x 81 columns]
```

In [119]:
```
predictors_objects = []
for i in prices.columns:
    if prices[i].dtype == "object":
        predictors_objects.append(i)
```

In [120]:
```
predictors = []
for i in prices.columns:
    if i != "SalePrice" and i != "Id":
        predictors.append(i)

for i in predictors:
    if prices[i].dtype == 'object':
        prices[i].fillna(value = "None", inplace = True)

prices.MasVnrArea.fillna(value = 0, inplace = True)
prices.LotFrontage.fillna(value = 0, inplace = True)
prices.GarageYrBlt.fillna(value = 0, inplace = True)
```

In [121]:
```
prices = pd.get_dummies(prices, columns = predictors_objects, drop_first = True)
prices.head()
```

Out[121]:
```
   Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  \
0   1          60         65.0     8450            7            5       2003
1   2          20         80.0     9600            6            8       1976
2   3          60         68.0    11250            7            5       2001
3   4          70         60.0     9550            7            5       1915
4   5          60         84.0    14260            8            5       2000

   YearRemodAdd  MasVnrArea  BsmtFinSF1           ...             \
0          2003       196.0         706           ...
1          1976         0.0         978           ...
2          2002       162.0         486           ...
3          1970         0.0         216           ...
4          2000       350.0         655           ...

   SaleType_ConLI  SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
0               0               0             0             0            1
1               0               0             0             0            1
2               0               0             0             0            1
3               0               0             0             0            1
4               0               0             0             0            1

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                      0                     0                     0
1                      0                     0                     0
2                      0                     0                     0
3                      0                     0                     0
4                      0                     0                     0
```

```
   SaleCondition_Normal  SaleCondition_Partial
0                     1                      0
1                     1                      0
2                     1                      0
3                     0                      0
4                     1                      0

[5 rows x 262 columns]
```

In [122]:
```python
predictors = []
for i in prices.columns:
    if i != "SalePrice" and i != "Id":
        predictors.append(i)

for i in predictors:
    if prices[i].isnull().sum() > 0:
        print i
```

In [123]:
```python
prices.dropna(inplace = True)
```

5 Visualization

In [124]:
```python
prices_numbers = pd.DataFrame()
for i in predictors:
    if prices[i].dtype == "int" or prices[i].dtype == "float":
        prices_numbers[i] = prices[i]
prices_numbers['SalePrice'] = prices['SalePrice']
```

In [125]:
```python
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(prices_numbers)
prices_normalized = pd.DataFrame(np_scaled)
prices_normalized.columns = prices_numbers.columns
```

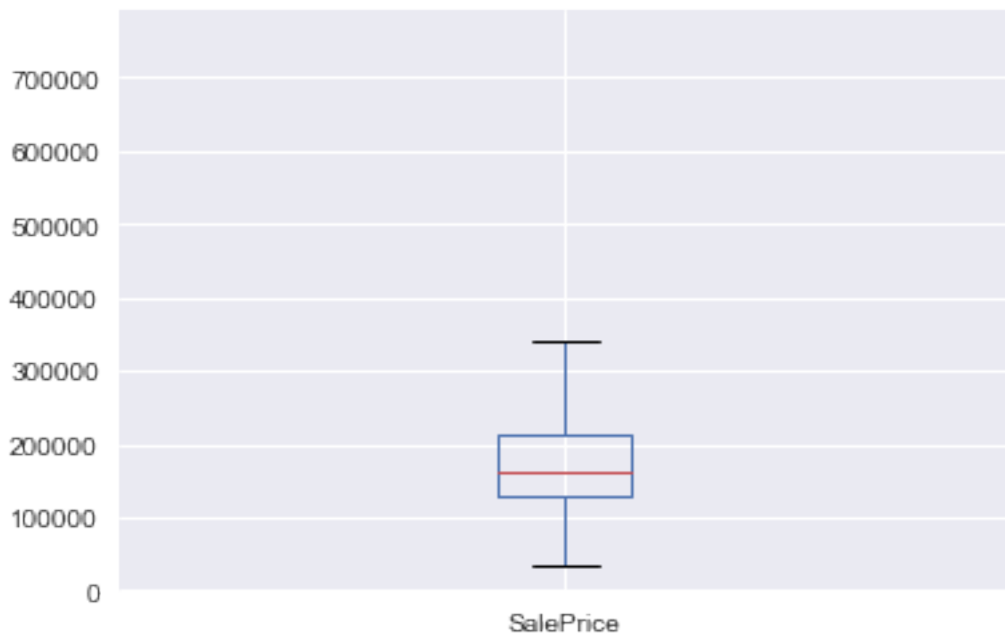
In [126]:
```python
prices_normalized.head()
```

Out[126]:
```
   MSSubClass  LotFrontage   LotArea  OverallQual  OverallCond  YearBuilt  \
0    0.235294     0.207668  0.033420     0.666667        0.500   0.949275
1    0.000000     0.255591  0.038795     0.555556        0.875   0.753623
2    0.235294     0.217252  0.046507     0.666667        0.500   0.934783
3    0.294118     0.191693  0.038561     0.666667        0.500   0.311594
4    0.235294     0.268371  0.060576     0.777778        0.500   0.927536

   YearRemodAdd  MasVnrArea  BsmtFinSF1  BsmtFinSF2     ...      WoodDeckSF  \
0      0.883333     0.12250    0.125089         0.0     ...        0.000000
1      0.433333     0.00000    0.173281         0.0     ...        0.347725
2      0.866667     0.10125    0.086109         0.0     ...        0.000000
3      0.333333     0.00000    0.038271         0.0     ...        0.000000
```

```
4       0.833333      0.21875     0.116052          0.0     ...         0.224037

   OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  MiscVal  \
0     0.111517       0.000000        0.0          0.0       0.0      0.0
1     0.000000       0.000000        0.0          0.0       0.0      0.0
2     0.076782       0.000000        0.0          0.0       0.0      0.0
3     0.063985       0.492754        0.0          0.0       0.0      0.0
4     0.153565       0.000000        0.0          0.0       0.0      0.0

     MoSold  YrSold  SalePrice
0  0.090909    0.50   0.241078
1  0.363636    0.25   0.203583
2  0.727273    0.50   0.261908
3  0.090909    0.00   0.145952
4  1.000000    0.50   0.298709

[5 rows x 37 columns]
```

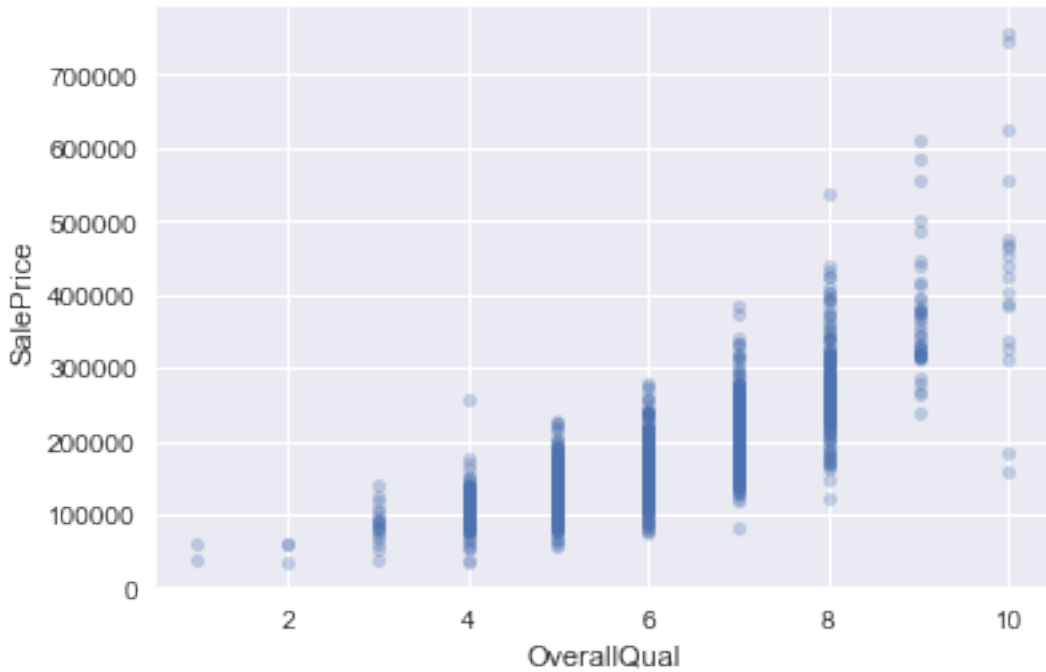
```
In [127]: prices.boxplot(column = "SalePrice")
```

```
Out[127]: <matplotlib.axes._subplots.AxesSubplot at 0x114b6c7d0>
```



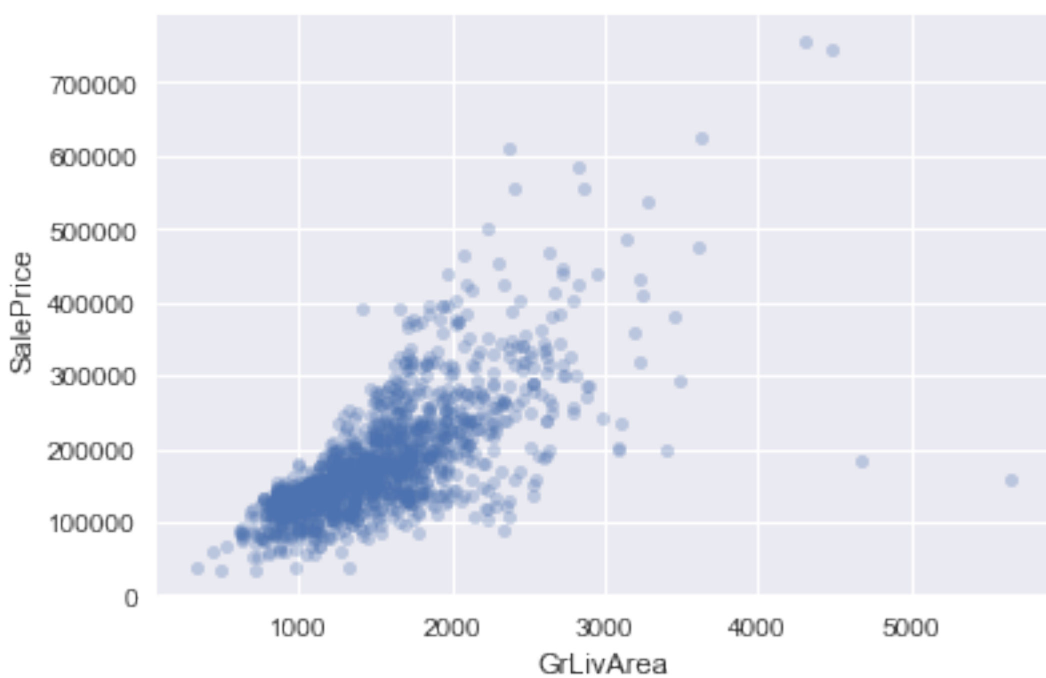

```
In [128]: prices.plot(kind='scatter', x='OverallQual', y='SalePrice', alpha=0.3)
```

```
Out[128]: <matplotlib.axes._subplots.AxesSubplot at 0x114b3c290>
```



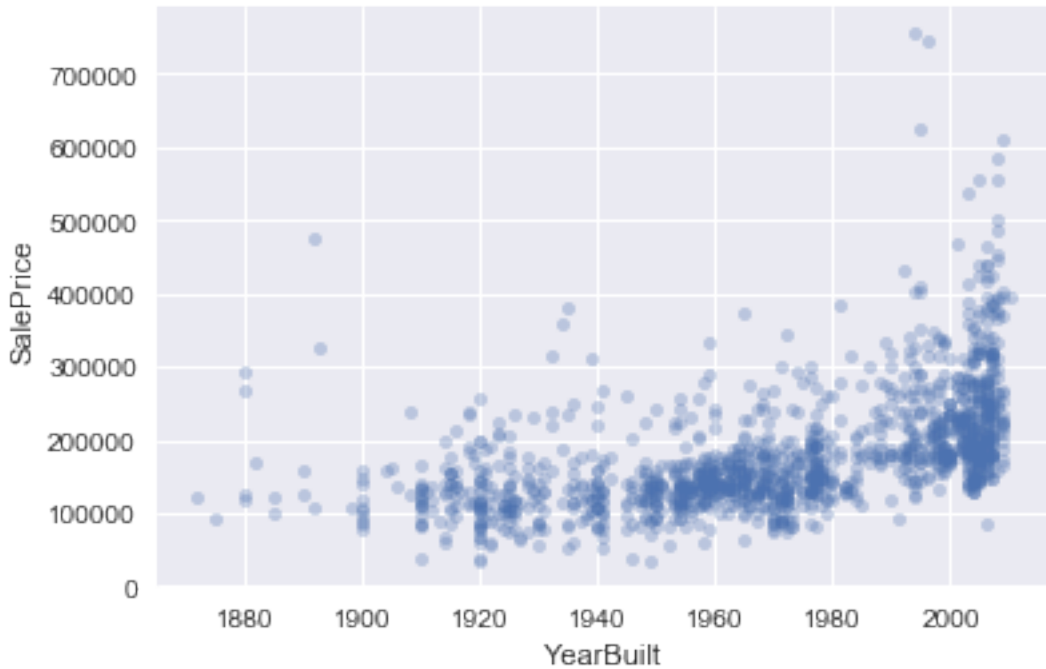
In [129]: prices.plot(kind='scatter', x='GrLivArea', y='SalePrice', alpha=0.3)

Out[129]: <matplotlib.axes._subplots.AxesSubplot at 0x116470810>



```
In [130]: prices.plot(kind='scatter', x='YearBuilt', y='SalePrice', alpha=0.3)
```

```
Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x113c68350>
```




```
In [131]: prices.head()
```

```
Out[131]:    Id  MSSubClass  LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  \
          0   1          60         65.0     8450            7            5       2003
          1   2          20         80.0     9600            6            8       1976
          2   3          60         68.0    11250            7            5       2001
          3   4          70         60.0     9550            7            5       1915
          4   5          60         84.0    14260            8            5       2000

             YearRemodAdd  MasVnrArea  BsmtFinSF1          ...            \
          0          2003       196.0         706          ...
          1          1976         0.0         978          ...
          2          2002       162.0         486          ...
          3          1970         0.0         216          ...
          4          2000       350.0         655          ...

             SaleType_ConLI  SaleType_ConLw  SaleType_New  SaleType_Oth  SaleType_WD  \
          0               0               0             0             0            1
          1               0               0             0             0            1
          2               0               0             0             0            1
          3               0               0             0             0            1
```

```
4                0                0                0                0             1

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                      0                     0                     0
1                      0                     0                     0
2                      0                     0                     0
3                      0                     0                     0
4                      0                     0                     0

   SaleCondition_Normal  SaleCondition_Partial
0                     1                      0
1                     1                      0
2                     1                      0
3                     0                      0
4                     1                      0

[5 rows x 262 columns]
```

6 Decision Tree

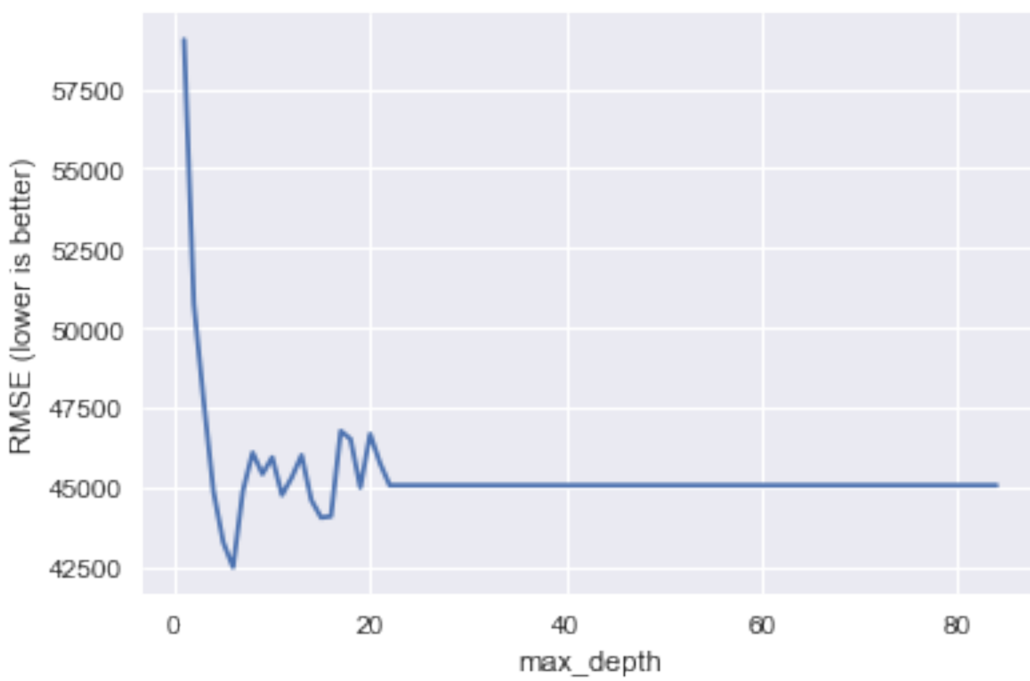
```
In [132]: predictors = []
          for i in prices.columns:
              if i != 'SalePrice':
                  predictors.append(i)

          X_train, X_test, y_train, y_test = model_selection.train_test_split(prices[predictors

In [133]: treereg = DecisionTreeRegressor(random_state=1)

In [134]: scores = cross_val_score(treereg, X_train, y_train, cv = 24, scoring='mean_squared_e
          np.mean(np.sqrt(-scores))
```

Out[134]: 42162.355724391797

In [135]:
```
max_depth_range = range(1, 85)

# list to store the average RMSE for each value of max_depth
RMSE_scores = []

# use LOOCV with each value of max_depth
for depth in max_depth_range:
    treereg = DecisionTreeRegressor(max_depth=depth, random_state=1)
    MSE_scores = cross_val_score(treereg, X_train, y_train, scoring='mean_squared_er
    RMSE_scores.append(np.mean(np.sqrt(-MSE_scores)))
```

In [136]:
```
plt.plot(max_depth_range, RMSE_scores)
plt.xlabel('max_depth')
plt.ylabel('RMSE (lower is better)')
```

Out[136]: <matplotlib.text.Text at 0x116503250>



In [137]:
```
sorted(zip(RMSE_scores, max_depth_range))[0]
```

Out[137]:
```
(42482.074036571663, 6)
```

In [138]:
```
treereg = DecisionTreeRegressor(max_depth=6, random_state=1)
treereg.fit(X_train, y_train)
```

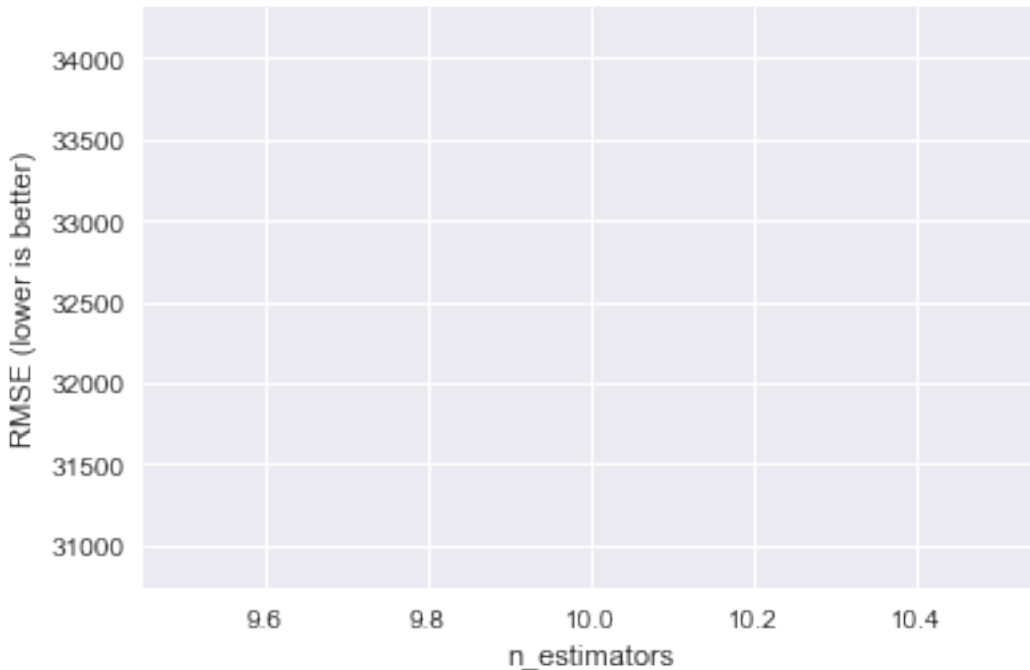
Out[138]:
```
DecisionTreeRegressor(criterion='mse', max_depth=6, max_features=None,
           max_leaf_nodes=None, min_impurity_split=1e-07,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, presort=False, random_state=1,
           splitter='best')
```

In [139]:
```
scores = cross_val_score(treereg, X_train, y_train, cv = 24, scoring='mean_squared_e
np.mean(np.sqrt(-scores))
```

```
Out[139]: 41868.303989185959
```

7 Random Forest

```
In [140]: rfreg = RandomForestRegressor()

In [141]: estimator_range = range(10, 50, 100)

          # list to store the average RMSE for each value of n_estimators
          RMSE_scores = []

          # use 5-fold cross-validation with each value of n_estimators (WARNING: SLOW!)
          for estimator in estimator_range:
              rfreg = RandomForestRegressor(n_estimators=estimator, random_state=1)
              MSE_scores = cross_val_score(rfreg, X_train, y_train, scoring='neg_mean_squared_e
              RMSE_scores.append(np.mean(np.sqrt(-MSE_scores)))

In [142]: plt.plot(estimator_range, RMSE_scores)
          plt.xlabel('n_estimators')
          plt.ylabel('RMSE (lower is better)')

Out[142]: <matplotlib.text.Text at 0x114ef1590>
```



n_estimators optimized at 10

```
In [143]: sorted(zip(RMSE_scores, estimator_range))[0]

Out[143]: (32526.463597690206, 10)

In [144]: # list of values to try for max_features
          #feature_range = range(1, len(predictors)+1)

          # list to store the average RMSE for each value of max_features
          #RMSE_scores = []

          # use 10-fold cross-validation with each value of max_features (WARNING: SLOW!)
          #for feature in feature_range:
              #rfreg = RandomForestRegressor(n_estimators=150, max_features=feature, random_st
              #MSE_scores = cross_val_score(rfreg, X_train, y_train, scoring='mean_squared_err
              #RMSE_scores.append(np.mean(np.sqrt(-MSE_scores)))

In [145]: #plt.plot(feature_range, RMSE_scores)
          #plt.xlabel('max_features')
          #plt.ylabel('RMSE (lower is better)')
```

Using a for loop, different values of max_features were used and by using the RMSE, the max_features was optimized at 62.

```
In [146]: #sorted(zip(RMSE_scores, feature_range))[0]
```

In [147]:
```
rfreg = RandomForestRegressor(n_estimators=10, max_features=62, random_state=1)
rfreg.fit(X_train, y_train)
```

Out[147]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features=62, max_leaf_nodes=None, min_impurity_split=1e-07,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=1, verbose=0, warm_start=False)
```

In [148]:
```
scores = cross_val_score(rfreg, X_train, y_train, cv = 24, scoring='neg_mean_squared_
np.mean(np.sqrt(-scores))
```

Out[148]:
```
31950.385134446598
```

In [149]:
```
feature_importance = pd.DataFrame({'feature':predictors, 'importance':rfreg.feature_i
```

In [150]:
```
feature_importance.head(4)
```

Out[150]:

	feature	importance
4	OverallQual	0.157437
16	GrLivArea	0.145564
27	GarageArea	0.083353
153	ExterQual_TA	0.078937

In [151]:
```
X_train.shape
```

Out[151]:
```
(1168, 261)
```

In [152]:
```
#X_important = rfreg.transform(X_train, threshold='mean')
```

In [153]:
```
#rfreg_important = RandomForestRegressor(n_estimators=10, random_state=1)
#scores = cross_val_score(rfreg_important, X_important, y_train, cv = 24, scoring='n
#np.mean(np.sqrt(-scores))
#rfreg_important.fit(X_train, y_train)
```

In [154]:
```
#cv_range = range(2, 30, 1)

# list to store the average RMSE for each value of n_estimators
#RMSE_scores = []

# use 5-fold cross-validation with each value of n_estimators (WARNING: SLOW!)
#for cv_score in cv_range:
    #rfreg = RandomForestRegressor(n_estimators=10, random_state=1)
    #MSE_scores = cross_val_score(rfreg, X_important, y_train, cv = 24, scoring='neg_
    #RMSE_scores.append(np.mean(np.sqrt(-MSE_scores)))

#sorted(zip(RMSE_scores, cv_range))[0]
```

In [155]:
```
scores = cross_val_score(treereg, X_train, y_train, cv = 2, scoring='neg_mean_squared
score = np.mean(np.sqrt(-scores))
print "TreeReg Score:", score
```

```
TreeReg Score: 46267.1259865
```

```
In [156]: scores = cross_val_score(rfreg, X_train, y_train, cv = 24, scoring='neg_mean_squared_
          score = np.mean(np.sqrt(-scores))
          print "RFReg Score:", score
```

```
RFReg Score: 31950.3851344
```

```
In [157]: #scores = cross_val_score(rfreg_important, X_important, y_train, cv = 24, scoring='n
          #score = np.mean(np.sqrt(-scores))
          #print "RFReg (Important Features) Score:", score

In [158]: rfreg_predictions = rfreg.predict(X_test)

In [159]: test_predictions = pd.DataFrame(y_test)
          test_predictions["RFReg Predictions"] = rfreg_predictions

In [160]: treereg_predictions = treereg.predict(X_test)
          for i in range(len(treereg_predictions)):
              treereg_predictions[i] = round(treereg_predictions[i], 1)
          test_predictions["TreeReg Predictions"] = treereg_predictions

          #rfreg_importantfeatures_predictions = rfreg_important.predict(X_test)
          #test_predictions["RFReg (Important Features) Predictions"] = rfreg_importantfeature
```

8 Model Performance and Conclusion

```
In [161]: test_predictions.head(10)
```

Out[161]:

	SalePrice	RFReg Predictions	TreeReg Predictions
258	231500	208870.0	195070.0
267	179500	186240.0	140095.1
288	122000	124215.0	129696.4
649	84500	82000.0	69133.9
1233	142000	150100.0	141960.6
167	325624	301175.0	381427.4
926	285000	287271.3	292524.4
831	151000	156105.6	164482.5
1237	195000	215711.0	236673.8
426	275000	249407.2	295718.0

```
In [162]: rfreg_predictions = []
          for i in range(len(test_predictions["SalePrice"])):
              diff = abs(test_predictions.iloc[i, 0] - test_predictions.iloc[i, 1])
              rfreg_predictions.append(diff)

          test_predictions["RFReg Predictions Difference"] = rfreg_predictions
```

In [163]:
```
treereg_predictions = []
for i in range(len(test_predictions["SalePrice"])):
    diff = abs(test_predictions.iloc[i, 0] - test_predictions.iloc[i, 2])
    treereg_predictions.append(diff)

test_predictions["TreeReg Predictions Difference"] = treereg_predictions
```

In [164]:
```
rfreg_predictions_percent = []
for i in range(len(test_predictions["SalePrice"])):
    diff = abs((test_predictions.iloc[i, 0] - test_predictions.iloc[i, 1]) / test_pre
    rfreg_predictions_percent.append(diff)

test_predictions["RFReg Percent Difference"] = rfreg_predictions_percent
```

In [165]:
```
treereg_predictions_percent = []
for i in range(len(test_predictions["SalePrice"])):
    diff = abs((test_predictions.iloc[i, 0] - test_predictions.iloc[i, 2]) / test_pre
    treereg_predictions_percent.append(diff)

test_predictions["TreeReg Percent Difference"] = treereg_predictions_percent
```

In [166]:
```
test_predictions.head()
```

Out[166]:
```
      SalePrice  RFReg Predictions  TreeReg Predictions  \
258      231500           208870.0             195070.0
267      179500           186240.0             140095.1
288      122000           124215.0             129696.4
649       84500            82000.0              69133.9
1233     142000           150100.0             141960.6

      RFReg Predictions Difference  TreeReg Predictions Difference  \
258                        22630.0                         36430.0
267                         6740.0                         39404.9
288                         2215.0                          7696.4
649                         2500.0                         15366.1
1233                        8100.0                            39.4

      RFReg Percent Difference  TreeReg Percent Difference
258                   9.775378                   15.736501
267                   3.754875                   21.952591
288                   1.815574                    6.308525
649                   2.958580                   18.184734
1233                  5.704225                    0.027746
```

In [167]:
```
test_predictions = test_predictions.sort_values("RFReg Percent Difference", ascending
test_predictions.head()
```

Out[167]:
```
      SalePrice  RFReg Predictions  TreeReg Predictions  \
1340     123000           123067.5             102163.0
```

```
1010     135000          135097.5            129696.4
750       96500           96430.0            102163.0
498      130000          129870.0            129696.4
1434     160000          160160.0            141960.6

      RFReg Predictions Difference  TreeReg Predictions Difference  \
1340                          67.5                         20837.0
1010                          97.5                          5303.6
750                           70.0                          5663.0
498                          130.0                           303.6
1434                         160.0                         18039.4

      RFReg Percent Difference  TreeReg Percent Difference
1340                  0.054878                   16.940650
1010                  0.072222                    3.928593
750                   0.072539                    5.868394
498                   0.100000                    0.233538
1434                  0.100000                   11.274625
```

In [168]:
```
print "RFReg Mean Percent Difference:", test_predictions['RFReg Percent Difference']
print "TreeReg Mean Percent Difference:", test_predictions['TreeReg Percent Differen
```

```
RFReg Mean Percent Difference: 11.398929783
TreeReg Mean Percent Difference: 14.8535089154
```

9 Moving Forward

- Continue to tune model
  - Analyze effects of adjusting other features in the model
- Analyze other effects:
  - Do different regions (cities, states, etc) pay higher for certain features?
  - How does the duration that a house was on the market for affect the final sale price?
    - Effect of the number of owners
  - Additional datapoints moving beyond descriptive features of a house
    - How much other houses around it have been sold for