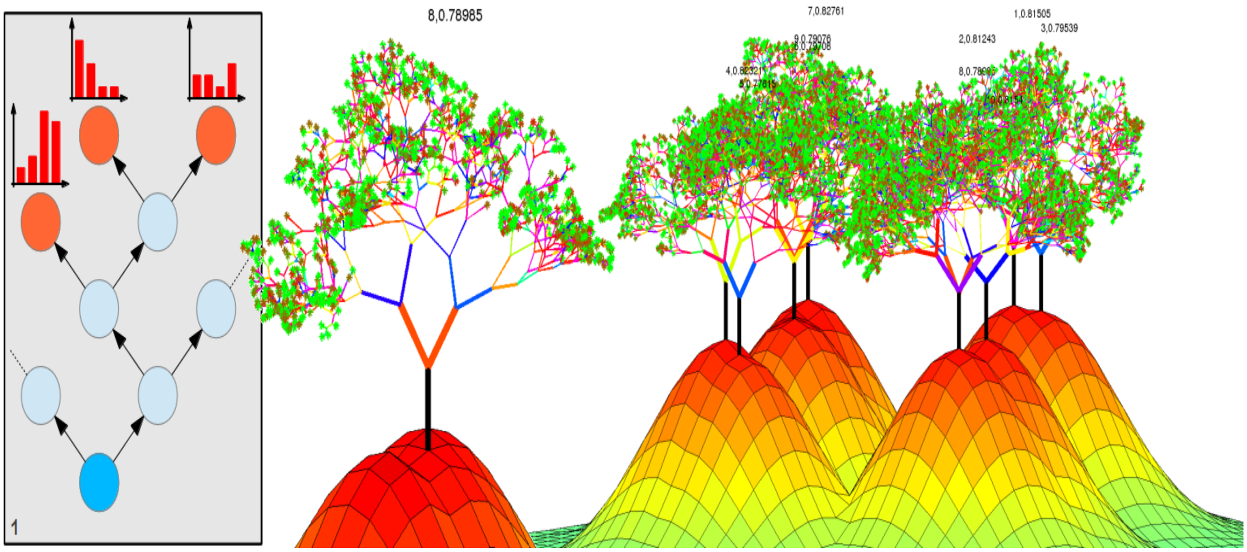
Housing Regression:
A Random Forest Approach



Abstract:

This report analyzes the methods and the results of an attempt to regress the sale price of a house on about 80 other features of said house. The data was drawn from the Ames Housing dataset, compiled by Dean De Cock and posted on Kaggle. The key phases of analysis were preprocessing the data, and then fine-tuning a Random Forest regressor to fit the data. The accuracy of prediction was measured by the root mean square error (RMSE).

Minas Badikyan | Math 542A | Professor Adriano Zambom

Introduction:

The aim of trying to reasonably predict the selling price of a house with a given set of attributes is a clear point of interest in the real estate business. Moreover, it is not an unreasonable assumption that certain house attributes may generalize to the wider housing market in their ability to predict selling prices. Hence, reliable analysis of the Ames dataset might bear fruit in wider contexts.

The dimension of the training data is approximately 1460 x 81, including the target variable 'SalesPrice', where about half of the variables are categorical. This data was plagued with missing values and redundant variables (i.e. the number of cars the garage fits and the square footage of the garage are highly correlated). Hence, the project was split into the data preprocessing phase, the model selection/application phase, and the conclusion.

Methodology:

A data preprocessing pipeline was built using Scikit-Learn. This pipeline first separated the numerical variables from the categoricals. Each numerical variable was then imputed by its respective median value, and then scaled to be standard normal. Concurrently, each categorical variable was imputed by its most frequently occurring value, to then be transformed into multiple dummy variables. The numerical and categorical variables were then rejoined into the same dataset. This pipeline facilitates preprocessing datasets with new observations (with the same variables) and is good practice in a business context.

The added dummy variables swelled the total number of variables from 80 up to 250+. A dimensionality reduction method known as F-Regression was used to curb the number of predictors. F-Regression regresses the target variable, sales price, on each predictor independently. A p-value is calculated for each variable based on an F-test, and the k most significant variables are retained. In this case, k=50 was found (by trial and error) to reduce the RMSE of each preliminary model.

The preprocessed, reduced dataset was then separately fed into a linear regressor, a decision tree regressor, and a random forest regressor, all of which were initialized in their default states to simply acquire a preliminary sense of accuracy. After a series of trials and errors, it was discovered that the untuned random forest regressor yielded the lowest average RMSE at around $31,638 with a standard deviation of $6,759 [under Preliminary Models in Appendix].

It was, in fact, unsurprising that the random forest regressor outperformed the decision tree regressor since a random forest is a set of decision trees. If the ‘bootstrap method’ is toggled on in the random forest function’s parameter line, then each of the forest’s decision trees will fit a different subset of the data. If ‘bootstrap’ is toggled off (as was the case here), then each decision tree fits the entire dataset. In effect, a random forest is a ‘cross-validation’ of decision trees and so it’s virtually guaranteed to outperform any single decision tree. After choosing the random forest as our model, all that was left was to fine-tune its parameters.

Scikit-Learn features the GridCV function, which iterates across a specified parameter space for a specified model, and outputs a list of error metrics for each iteration. This streamlines the process of finding the optimum parameters to use in the model so that it minimizes the RMSE. Our application of GridCV found that the best random forest model does not implement the bootstrap method and is composed of 100 decision trees (the parameter space of 'number of decision trees' was defined across orders of magnitude: 10, 100, 1000, etc). The RMSE of this fine-tuned random forest is $29,141. [see Appendix]

Data Analysis:

GridSearch's 'Feature Importances' method assigns each predictor an importance metric between zero and one. According to our fine-tuned random forest, the top four predictors of sale price, from most to least relevant, are: the number of cars the garage fits, the square footage of living area above the first floor, and whether the exterior quality of the house is labeled as 'average' [see appendix for importance scores].

Larger garages fit more cars, and it's not unreasonable to assume that increased garage square footage correlates with the house's total square footage, which is a prime factor for determining selling price. The square footage above the first floor indicates a multi-story house, which would clearly sell for a higher price. Houses with an 'average' rating regarding exterior quality sell at higher prices; the interpretation here might be that if a house has 'poor' exterior quality, then it is simply unappealing to potential buyers and thus not likely to be bought; if the exterior is rated as 'high' or 'good', then any renovation might incur great cost and not raise the value of the home, and so any potential buyer is off-put by the fact that he/she can't quite customize the exterior of the house without a significant added investment; hence, a house with ‘average’ exterior quality is appealing enough to buy, and more than likely worth renovating since the property value is likely to increase. In many ways, these results don’t disagree with one’s intuition as to which factors might impact the selling price of a house.

Conclusion:

The average RMSE of our fine-tuned random forest is accurate to about 16% when compared to the mean value of selling price in the dataset. This is not a bad start, but there is room for improvement. Better accuracy might be achieved through more thorough feature engineering, i.e. combining and/or removing variables so that the data better reflects the reality of what affects selling price; this process can be more intuitive with better domain (real estate) knowledge.

Additional models may be trained that might bear a lower RMSE than our random forest, examples being lasso regression, support vector machines, and gradient boosting. A similar process of trial and error in addition to fine-tuning across large parameter spaces could refine any one of these models to outperform what has been tried in this project.

Appendix

In [1]:
```
# TODO:
#     Figure out which columns to drop (there certainly are superfluous ones)
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
training_df = pd.read_csv("C://Users//mbadi//M542A_Project//train.csv")
```

In [4]:
```
df_target = training_df[["SalePrice"]]
df_predictors = training_df.drop("SalePrice",axis=1)
```

Data Cleaning

Observe that there are three columns ('MSSubClass','OverallQual','OverallCond'), that are floats but also categorical, so that dtypes won't capture them in the right category. Hence we'll capture them manually. The following code segragates the categorical variables (cat_var) from the numerical variables (num_var).

The "cols_to_drop" array lists the columns of the dataset that we elect to ignore.

In [43]:
```
cols_to_drop = ['GarageArea','TotalBsmtSF','1stFlrSF','SalePrice']

numeric_categoricals = ['MSSubClass','OverallQual','OverallCond']

cat_var = [col for col in df_predictors.columns
           if df_predictors[col].dtype == 'O' and
           col not in cols_to_drop]

for col in numeric_categoricals:
    cat_var.append(col)

num_var = [col for col in df_predictors.columns
           if df_predictors[col].dtype in ['float64','int64'] and
           col not in numeric_categoricals and
           col not in cols_to_drop]
```

Below we will implement a class (DataFrameSelector) that will return the values of any set of selected columns as an array. This enables us to create separate pipelines for categorical and numerical data.

In [6]:
```
from sklearn.base import BaseEstimator, TransformerMixin
```

Implement two classes, one to drop selected columns from the dataframe, another to convert the data corresponding to selected columns into a numpy array. These will be used as part of the data preprocessing pipeline.

In [7]:
```
class DropDFColumns(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
        return
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X.drop(self.attribute_names, axis=1, inplace=False)
```

In [8]:
```
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
        return
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].to_numpy()
```

Now we'll import an imputer, an encoder, a standard scaler, as well as the pipeline framework to build our data preprocessing pipeline.

In [9]:
```
from sklearn.feature_selection import SelectKBest, f_regression, chi2
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.decomposition import PCA
```

In [10]:
```
numerical_pipeline = Pipeline([
    ('col_dropper', DropDFColumns(cols_to_drop)),
    ('selector', DataFrameSelector(num_var)),
    ('imputer', SimpleImputer(missing_values=np.nan, strategy='median')),
    ('std_scaler', StandardScaler()),
])

categorical_pipeline = Pipeline([
    ('selector', DataFrameSelector(cat_var)),
    ('imputer', SimpleImputer(missing_values=np.nan, strategy='most_frequent'
)),
    ('encoder', OneHotEncoder()),
])

full_pipeline = FeatureUnion(transformer_list=[
    ('num_pipeline', numerical_pipeline),
    ('cat_pipeline', categorical_pipeline),
])
```

In [11]:
```
X_num = pd.DataFrame(numerical_pipeline.fit_transform(df_predictors), columns=
num_var)
```

In [12]:
```
def dummies(data, columns):
    imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
    dummy_df = pd.DataFrame(imp.fit_transform(data[columns]), columns=columns)
    return(pd.get_dummies(dummy_df, drop_first=True))
```

In [13]:
```
X_dummies = dummies(df_predictors, cat_var)
```

In [14]:
```
X_train = X_num.join(X_dummies)
```

In [15]:
```
y_train = SimpleImputer(missing_values=np.nan, strategy='median').fit_transfor
m(df_target)
```

In [79]:
```
np.mean(y_train)
```

Out[79]:
```
180921.19589041095
```

In [80]:
```
29000/np.mean(y_train)*100
```

Out[80]:
```
16.029078216775723
```

In [66]:
```
fs = SelectKBest(score_func=f_regression, k=50)
fs.fit(X_train, y_train)

cols = fs.get_support(indices=False)
X_new = X_train.iloc[:,cols].copy()

# X_new = X_train[['GarageCars','GrLivArea','1stFlrSF','TotalBsmtSF',
#                  'YearBuilt','Fireplaces','2ndFlrSF','Neighborhood_NridgH
t']].copy()
```

```
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: D
ataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]:
```

```

Preliminary Models

We commence to attack the processed X_train data with an array of common ML models, the accuracies of which will be measured by the Root Mean Squared Error (RMSE) between the fitted values and the actual values of the Sale Price. The RMSE will be the average of a set of RMSEs obtained by the various iterations of the cross-validation (CV) function. The hyperparameters of each model will be optimized by GridSearchCV.

In [17]:
```
from sklearn.metrics import mean_squared_error, mean_squared_log_error
from sklearn.model_selection import cross_val_score, GridSearchCV

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR, LinearSVR
```

In [18]:
```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard Deviation:", scores.std())
```

In [19]:
```
reg = LinearRegression()
reg_score = cross_val_score(reg, X_new, y_train,
                            scoring ='neg_mean_squared_error', cv=10)
display_scores(np.sqrt(-reg_score))
```

```
Scores: [25816.28779811 27656.20481532 26172.70499779 42444.62440647
 36316.30625587 29442.80424962 24790.32104869 28515.04322191
 56659.84334077 25230.8657891 ]
Mean: 32304.50059236432
Standard Deviation: 9707.988405495626
```

In [20]:
```
tree = DecisionTreeRegressor()
tree_score = cross_val_score(tree, X_new, y_train,
                             scoring='neg_mean_squared_error', cv=10)
display_scores(np.sqrt(-tree_score))
```

```
Scores: [40961.92163352 45065.68856141 42397.03032449 62265.38981814
 63063.90190795 36206.5949055  33411.12086095 36766.02966148
 39186.64255159 34468.00209466]
Mean: 43379.23223196814
Standard Deviation: 10224.273393443667
```

In [77]:
```
forest = RandomForestRegressor()
forest_score = cross_val_score(forest, X_new, y_train,
                               scoring='neg_mean_squared_error', cv=10)
display_scores(np.sqrt(-forest_score))
```

```
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
```

```
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\mbadi\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:528: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

Scores: [23579.64131383 29210.86620043 31448.03895097 47482.12207882
 36145.76492665 27370.80514794 26397.81704028 28368.35629758
 38362.06857658 28022.17428033]
Mean: 31638.765481341972
Standard Deviation: 6759.191407069979
```

28740.58906104257 {'bootstrap': False, 'max_features': 8, 'n_estimators': 70}

In [27]:

```python
# forest_reg = RandomForestRegressor(n_estimators=70, max_features=8, bootstra
p=False)
forest_reg = RandomForestRegressor()

# forest_scores = cross_val_score(forest_reg, X_train, y_train,
#                                 scoring='neg_mean_squared_log_error', cv=10)

# forest_scores = cross_val_score(forest_reg, X_new, y_train,
#                                 scoring='neg_mean_squared_error', cv=10)

# display_scores(np.sqrt(-forest_scores))

# forest_reg.fit(X_new, y_train)
```

In [23]:
```
# feature_importances = forest_reg.feature_importances_
# sorted(zip(feature_importances, X_train.columns), reverse=True)
```

In [45]:
```
param_grid = [
    {'n_estimators':[30,40,100,50,70], 'max_features':[2,4,6,8,10]},
    {'bootstrap':[False], 'n_estimators':[40,10,70,100], 'max_features':[2,4,6
,8,10]},
]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error')

grid_search.fit(X_new, y_train)
```

In [78]:
```
grid_search.best_estimator_
```

Out[78]:
```
RandomForestRegressor(bootstrap=False, criterion='mse', max_depth=None,
           max_features=10, max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=100, n_jobs=None, oob_score=False,
           random_state=None, verbose=0, warm_start=False)
```

In [29]:
```
feature_imp = grid_search.best_estimator_.feature_importances_
sorted(zip(feature_imp, X_new.columns), reverse=True)
```

Out[29]:
```
[(0.1511872504169663, 'GarageCars'),
 (0.146180640214848, 'GrLivArea'),
 (0.08333303717614429, 'ExterQual_TA'),
 (0.08160363265304117, 'YearBuilt'),
 (0.058113278461225465, 'BsmtFinSF1'),
 (0.04072671624246063, 'LotArea'),
 (0.03701776312560587, 'FullBath'),
 (0.034559437521671545, 'TotRmsAbvGrd'),
 (0.03300420890822887, '2ndFlrSF'),
 (0.03046437054797408, 'KitchenQual_TA'),
 (0.025659064635206126, 'MasVnrArea'),
 (0.024454605422299814, 'GarageYrBlt'),
 (0.022234006483450742, 'YearRemodAdd'),
 (0.021974862849721003, 'Fireplaces'),
 (0.0211130508865941, 'LotFrontage'),
 (0.017842705643579128, 'Foundation_PConc'),
 (0.016596895240374422, 'GarageFinish_Unf'),
 (0.013134627374556564, 'OpenPorchSF'),
 (0.013102168102544932, 'BsmtQual_TA'),
 (0.01242898012050959, 'OverallQual_9'),
 (0.010804484363104869, 'KitchenQual_Gd'),
 (0.010711210509192944, 'OverallQual_10'),
 (0.01018651484831107, 'WoodDeckSF'),
 (0.00993224700656162, 'ExterQual_Gd'),
 (0.009691956892035185, 'OverallQual_8'),
 (0.006337561902175434, 'BsmtFinType1_GLQ'),
 (0.00496637750137116, 'BsmtExposure_Gd'),
 (0.0038948444950549503, 'Neighborhood_NridgHt'),
 (0.003800434216093384, 'BsmtExposure_No'),
 (0.0036137942822809218, 'CentralAir_Y'),
 (0.003603990400942198, 'HalfBath'),
 (0.0035324278351433636, 'Neighborhood_NoRidge'),
 (0.003447374515964851, 'MSZoning_RL'),
 (0.003101152449812834, 'GarageType_Detchd'),
 (0.0025529993357742546, 'MSZoning_RM'),
 (0.0023786333358916264, 'LotShape_Reg'),
 (0.0023349616288896153, 'MSSubClass_60'),
 (0.002328565281570928, 'HouseStyle_2Story'),
 (0.0021420315608720137, 'OverallQual_5'),
 (0.0020741368764507885, 'OverallCond_5'),
 (0.0019176274176424795, 'SaleCondition_Partial'),
 (0.0016101420953850063, 'OverallQual_4'),
 (0.0015032578574289677, 'SaleType_WD'),
 (0.0014942878564236808, 'SaleType_New'),
 (0.001421114215346942, 'MasVnrType_Stone'),
 (0.0013777682722759046, 'Exterior2nd_VinylSd'),
 (0.0012244633688769401, 'Exterior1st_VinylSd'),
 (0.0012008475247681665, 'Foundation_CBlock'),
 (0.00109444158320118, 'HeatingQC_TA'),
 (0.0009890485441541121, 'MasVnrType_None')]
```

In [30]:
```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print (np.sqrt(-mean_score), params)
```

```
33940.16536029949 {'max_features': 2, 'n_estimators': 30}
33719.63202112688 {'max_features': 2, 'n_estimators': 40}
33181.31442529863 {'max_features': 2, 'n_estimators': 100}
33734.24093131673 {'max_features': 2, 'n_estimators': 50}
33411.81187148052 {'max_features': 2, 'n_estimators': 70}
31446.174919320238 {'max_features': 4, 'n_estimators': 30}
31607.24504275972 {'max_features': 4, 'n_estimators': 40}
31617.01293795404 {'max_features': 4, 'n_estimators': 100}
33304.482016339694 {'max_features': 4, 'n_estimators': 50}
31459.477098959684 {'max_features': 4, 'n_estimators': 70}
31024.751091122787 {'max_features': 6, 'n_estimators': 30}
30883.35667230647 {'max_features': 6, 'n_estimators': 40}
31610.811823500084 {'max_features': 6, 'n_estimators': 100}
30680.118784988677 {'max_features': 6, 'n_estimators': 50}
30592.855764041713 {'max_features': 6, 'n_estimators': 70}
30963.088454256682 {'max_features': 8, 'n_estimators': 30}
31533.045922222922 {'max_features': 8, 'n_estimators': 40}
30223.985726086892 {'max_features': 8, 'n_estimators': 100}
31123.04036686616 {'max_features': 8, 'n_estimators': 50}
30428.095878085267 {'max_features': 8, 'n_estimators': 70}
31854.06476512492 {'max_features': 10, 'n_estimators': 30}
29929.93762714081 {'max_features': 10, 'n_estimators': 40}
30854.099434603635 {'max_features': 10, 'n_estimators': 100}
30968.125902049986 {'max_features': 10, 'n_estimators': 50}
30670.776460529258 {'max_features': 10, 'n_estimators': 70}
32047.012237340572 {'bootstrap': False, 'max_features': 2, 'n_estimators': 4
0}
34019.019520874856 {'bootstrap': False, 'max_features': 2, 'n_estimators': 1
0}
31460.715127203213 {'bootstrap': False, 'max_features': 2, 'n_estimators': 7
0}
31662.914939786155 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10
0}
30693.31685766021 {'bootstrap': False, 'max_features': 4, 'n_estimators': 40}
34402.10950200633 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
29819.075910531225 {'bootstrap': False, 'max_features': 4, 'n_estimators': 7
0}
30312.774841288654 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10
0}
29772.759283324693 {'bootstrap': False, 'max_features': 6, 'n_estimators': 4
0}
32100.88328547335 {'bootstrap': False, 'max_features': 6, 'n_estimators': 10}
29399.699005430575 {'bootstrap': False, 'max_features': 6, 'n_estimators': 7
0}
29783.63254114489 {'bootstrap': False, 'max_features': 6, 'n_estimators': 10
0}
29680.148017278534 {'bootstrap': False, 'max_features': 8, 'n_estimators': 4
0}
32037.30207704292 {'bootstrap': False, 'max_features': 8, 'n_estimators': 10}
29589.625601897857 {'bootstrap': False, 'max_features': 8, 'n_estimators': 7
0}
29428.424806193438 {'bootstrap': False, 'max_features': 8, 'n_estimators': 10
0}
29406.750248043987 {'bootstrap': False, 'max_features': 10, 'n_estimators': 4
0}
31313.479377048734 {'bootstrap': False, 'max_features': 10, 'n_estimators': 1
0}
```

```
29720.774309617926 {'bootstrap': False, 'max_features': 10, 'n_estimators': 7
0}
29141.917948756472 {'bootstrap': False, 'max_features': 10, 'n_estimators': 1
00}
```

In [61]:
```
def intersection(lst1, lst2):
    return list(set(lst1) & set(lst2))
```

In [62]:
```
cols_test = intersection(X_test.columns, cols)
```

In [32]:
```
tuned_forest = grid_search.best_estimator_
```

In [41]:
```
X_test_raw = pd.read_csv("C://Users//mbadi//M542A_Project//test.csv")
```

In [44]:
```
X_raw_num = pd.DataFrame(numerical_pipeline.fit_transform(X_test_raw), columns
=num_var)
```

In [47]:
```
X_raw_dummies = dummies(X_test_raw, cat_var)
```

In [48]:
```
X_test = X_raw_num.join(X_raw_dummies)
```

In [56]:
```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Columns: 254 entries, Id to OverallCond_9
dtypes: float64(31), uint8(223)
memory usage: 671.2 KB
```

In [58]:
```
X_test_prepared = X_test.loc[:,cols].copy()
```

In [ ]:

Feature Descriptions

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
```

LotShape: General shape of property

```
Reg  Regular
IR1  Slightly irregular
IR2  Moderately Irregular
IR3  Irregular
```

LandContour: Flatness of the property

```
Lvl  Near Flat/Level
Bnk  Banked - Quick and significant rise from street grade to building
HLS  Hillside - Significant slope from side to side
Low  Depression
```

Utilities: Type of utilities available

```
AllPub   All public Utilities (E,G,W,& S)
NoSewr   Electricity, Gas, and Water (Septic Tank)
NoSeWa   Electricity and Gas Only
ELO  Electricity only
```

LotConfig: Lot configuration

```
Inside   Inside lot
Corner   Corner lot
CulDSac  Cul-de-sac
FR2  Frontage on 2 sides of property
FR3  Frontage on 3 sides of property
```

LandSlope: Slope of property

```
Gtl  Gentle slope
Mod  Moderate Slope
Sev  Severe Slope
```

Neighborhood: Physical locations within Ames city limits

```
Blmngtn  Bloomington Heights
Blueste  Bluestem
BrDale   Briardale
BrkSide  Brookside
ClearCr  Clear Creek
CollgCr  College Creek
Crawfor  Crawford
Edwards  Edwards
Gilbert  Gilbert
IDOTRR   Iowa DOT and Rail Road
MeadowV  Meadow Village
Mitchel  Mitchell
Names    North Ames
NoRidge  Northridge
NPkVill  Northpark Villa
NridgHt  Northridge Heights
NWAmes   Northwest Ames
OldTown  Old Town
SWISU    South & West of Iowa State University
Sawyer   Sawyer
SawyerW  Sawyer West
Somerst  Somerset
StoneBr  Stone Brook
Timber   Timberland
Veenker  Veenker
```

Condition1: Proximity to various conditions

```
Artery   Adjacent to arterial street
Feedr    Adjacent to feeder street
Norm Normal
RRNn Within 200' of North-South Railroad
RRAn Adjacent to North-South Railroad
PosN Near positive off-site feature--park, greenbelt, etc.
PosA Adjacent to postive off-site feature
RRNe Within 200' of East-West Railroad
RRAe Adjacent to East-West Railroad
```

Condition2: Proximity to various conditions (if more than one is present)

```
Artery   Adjacent to arterial street
Feedr    Adjacent to feeder street
Norm Normal
RRNn Within 200' of North-South Railroad
RRAn Adjacent to North-South Railroad
PosN Near positive off-site feature--park, greenbelt, etc.
PosA Adjacent to postive off-site feature
RRNe Within 200' of East-West Railroad
RRAe Adjacent to East-West Railroad
```

BldgType: Type of dwelling

```
1Fam Single-family Detached
2FmCon   Two-family Conversion; originally built as one-family dwelling
Duplx    Duplex
TwnhsE   Townhouse End Unit
TwnhsI   Townhouse Inside Unit
```

HouseStyle: Style of dwelling

```
1Story   One story
1.5Fin   One and one-half story: 2nd level finished
1.5Unf   One and one-half story: 2nd level unfinished
2Story   Two story
2.5Fin   Two and one-half story: 2nd level finished
2.5Unf   Two and one-half story: 2nd level unfinished
SFoyer   Split Foyer
SLvl Split Level
```

OverallQual: Rates the overall material and finish of the house

```
10   Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

OverallCond: Rates the overall condition of the house

```
10   Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

```
Flat Flat
Gable    Gable
Gambrel  Gabrel (Barn)
Hip  Hip
Mansard  Mansard
Shed Shed
```

RoofMatl: Roof material

```
ClyTile  Clay or Tile
CompShg  Standard (Composite) Shingle
Membran  Membrane
Metal    Metal
Roll Roll
Tar&Grv  Gravel & Tar
WdShake  Wood Shakes
WdShngl  Wood Shingles
```

Exterior1st: Exterior covering on house

```
AsbShng  Asbestos Shingles
AsphShn  Asphalt Shingles
BrkComm  Brick Common
BrkFace  Brick Face
CBlock   Cinder Block
CemntBd  Cement Board
HdBoard  Hard Board
ImStucc  Imitation Stucco
MetalSd  Metal Siding
Other    Other
Plywood  Plywood
PreCast  PreCast
Stone    Stone
Stucco   Stucco
VinylSd  Vinyl Siding
Wd Sdng  Wood Siding
WdShing  Wood Shingles
```

Exterior2nd: Exterior covering on house (if more than one material)

```
AsbShng  Asbestos Shingles
AsphShn  Asphalt Shingles
BrkComm  Brick Common
BrkFace  Brick Face
CBlock   Cinder Block
CemntBd  Cement Board
HdBoard  Hard Board
ImStucc  Imitation Stucco
MetalSd  Metal Siding
Other    Other
Plywood  Plywood
PreCast  PreCast
Stone    Stone
Stucco   Stucco
VinylSd  Vinyl Siding
Wd Sdng  Wood Siding
WdShing  Wood Shingles
```

MasVnrType: Masonry veneer type

```
BrkCmn   Brick Common
BrkFace  Brick Face
CBlock   Cinder Block
None None
Stone    Stone
```

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
Po   Poor
```

ExterCond: Evaluates the present condition of the material on the exterior

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
Po   Poor
```

Foundation: Type of foundation

```
BrkTil   Brick & Tile
CBlock   Cinder Block
PConc    Poured Contrete
Slab Slab
Stone    Stone
Wood Wood
```

BsmtQual: Evaluates the height of the basement

```
Ex   Excellent (100+ inches)
Gd   Good (90-99 inches)
TA   Typical (80-89 inches)
Fa   Fair (70-79 inches)
Po   Poor (<70 inches
NA   No Basement
```

BsmtCond: Evaluates the general condition of the basement

```
Ex   Excellent
Gd   Good
TA   Typical - slight dampness allowed
Fa   Fair - dampness or some cracking or settling
Po   Poor - Severe cracking, settling, or wetness
NA   No Basement
```

BsmtExposure: Refers to walkout or garden level walls

```
Gd   Good Exposure
Av   Average Exposure (split levels or foyers typically score average or above)
Mn   Mimimum Exposure
No   No Exposure
NA   No Basement
```

BsmtFinType1: Rating of basement finished area

```
GLQ  Good Living Quarters
ALQ  Average Living Quarters
BLQ  Below Average Living Quarters
Rec  Average Rec Room
LwQ  Low Quality
Unf  Unfinshed
NA   No Basement
```

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

```
GLQ  Good Living Quarters
ALQ  Average Living Quarters
BLQ  Below Average Living Quarters
Rec  Average Rec Room
LwQ  Low Quality
Unf  Unfinshed
NA   No Basement
```

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

```
Floor    Floor Furnace
GasA Gas forced warm air furnace
GasW Gas hot water or steam heat
Grav Gravity furnace
OthW Hot water or steam heat other than gas
Wall Wall furnace
```

HeatingQC: Heating quality and condition

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
Po   Poor
```

CentralAir: Central air conditioning

```
N    No
Y    Yes
```

Electrical: Electrical system

```
SBrkr    Standard Circuit Breakers & Romex
FuseA    Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF    60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP    60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix  Mixed
```

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
```

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

```
Typ  Typical Functionality
Min1 Minor Deductions 1
Min2 Minor Deductions 2
Mod  Moderate Deductions
Maj1 Major Deductions 1
Maj2 Major Deductions 2
Sev  Severely Damaged
Sal  Salvage only
```

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

```
  Ex   Excellent - Exceptional Masonry Fireplace
  Gd   Good - Masonry Fireplace in main level
  TA   Average - Prefabricated Fireplace in main living area or Masonry Fireplace
in basement
  Fa   Fair - Prefabricated Fireplace in basement
  Po   Poor - Ben Franklin Stove
  NA   No Fireplace
```

GarageType: Garage location

```
2Types   More than one type of garage
Attchd   Attached to home
Basment  Basement Garage
BuiltIn  Built-In (Garage part of house - typically has room above garage)
CarPort  Car Port
Detchd   Detached from home
NA   No Garage
```

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

```
Fin  Finished
RFn  Rough Finished
Unf  Unfinished
NA   No Garage
```

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

GarageCond: Garage condition

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

PavedDrive: Paved driveway

```
Y    Paved
P    Partial Pavement
N    Dirt/Gravel
```

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
NA   No Pool
```

Fence: Fence quality

```
GdPrv    Good Privacy
MnPrv    Minimum Privacy
GdWo Good Wood
MnWw Minimum Wood/Wire
NA   No Fence
```

MiscFeature: Miscellaneous feature not covered in other categories

```
Elev Elevator
Gar2 2nd Garage (if not described in garage section)
Othr Other
Shed Shed (over 100 SF)
TenC Tennis Court
NA   None
```

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

```
WD   Warranty Deed - Conventional
CWD  Warranty Deed - Cash
VWD  Warranty Deed - VA Loan
New  Home just constructed and sold
COD  Court Officer Deed/Estate
Con  Contract 15% Down payment regular terms
ConLw    Contract Low Down payment and low interest
ConLI    Contract Low Interest
ConLD    Contract Low Down
Oth  Other
```

SaleCondition: Condition of sale

```
   Normal   Normal Sale
   Abnorml  Abnormal Sale -  trade, foreclosure, short sale
   AdjLand  Adjoining Land Purchase
   Alloca   Allocation - two linked properties with separate deeds, typically condo
with a garage unit
   Family   Sale between family members
   Partial  Home was not completed when last assessed (associated with New Homes)
```

Citations

Thanks to the Scikit Learn team.
Link to kaggle dataset: https://www.kaggle.com/c/house-prices-advanced-regression-techniques (https://www.kaggle.com/c/house-prices-advanced-regression-techniques)