House Prices: Advanced Regression Techniques

With 79 explanatory variables describing almost every aspect of residential homes in Ames, Iowa, this term paper shows the prediction of the final price of each home.

Data fields

Here's a brief version of what can be found in the data description file.

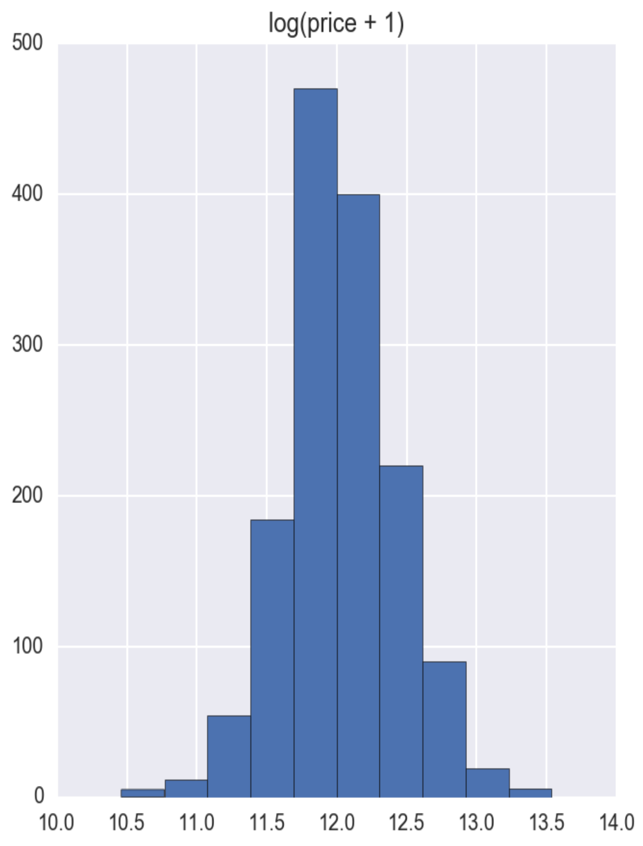
- **SalePrice** - the property's sale price in dollars. This is the target variable that you're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access
- **Alley**: Type of alley access
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling
- **HouseStyle**: Style of dwelling
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation

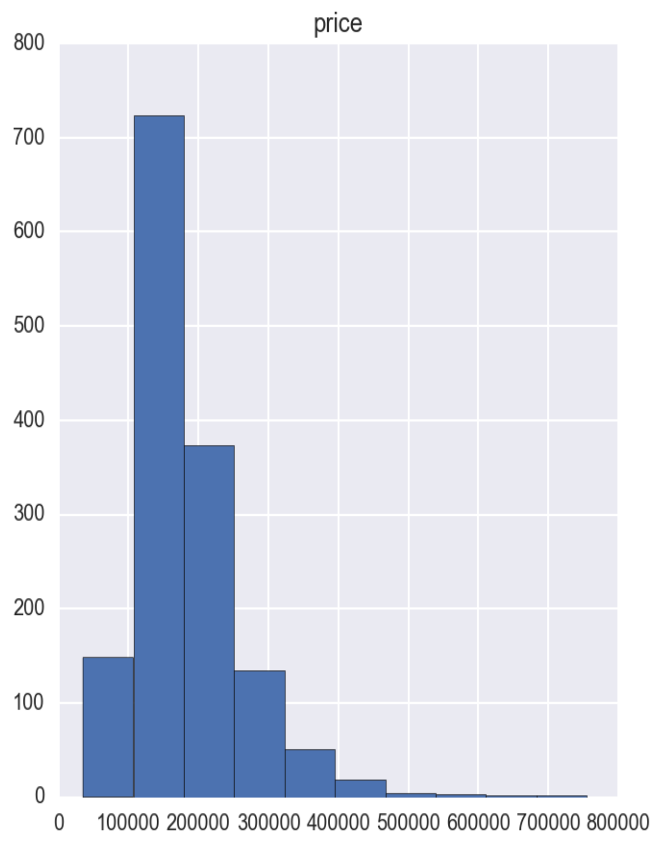
- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement
- **BsmtExposure**: Walkout or garden level basement walls
- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area
- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **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**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **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
- **Fence**: Fence quality

- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

Data preprocessing:

- First I'll transform the skewed numeric features by taking log (feature + 1) - this will make the features more normal
- Then create Dummy variables for the categorical features
- Finally replace the numeric missing values (NaN's) with the mean of their respective columns



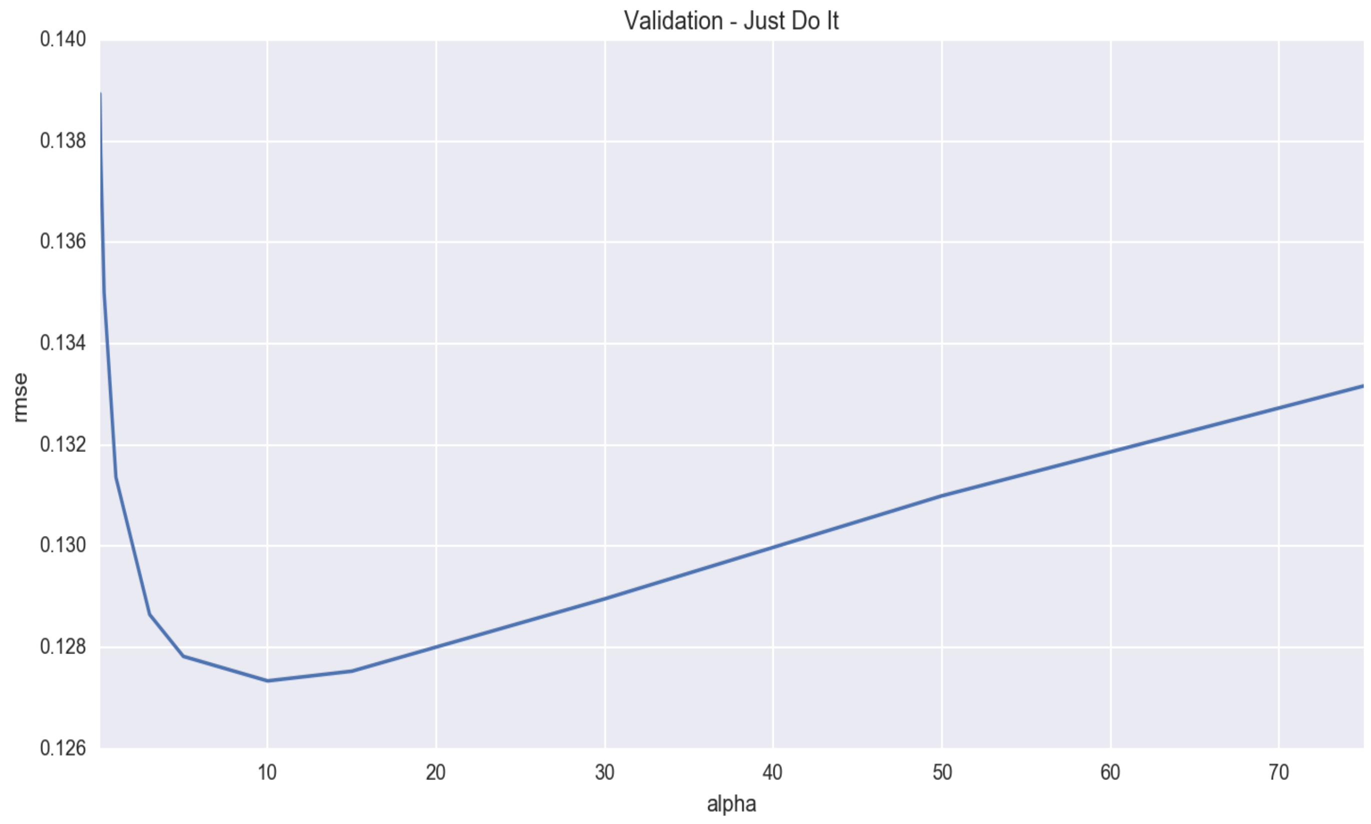


Models

Now I am going to use regularized linear regression models from the scikit learn module. I am going to try both l_1(Lasso) and l_2(Ridge) regularization. I will also define a function that returns the cross-validation(cv) rmse(root mean square error) so that I can evaluate the models and pick the best tuning par.

Ridge Regression is a remedial measure taken to alleviate multicollinearity amongst regression predictor variables in a model. Often predictor variables used in a regression are highly correlated. When they are, the regression coefficient of any one variable depend on which other predictor variables are included in the model, and which ones are left out. (So the predictor variable does not reflect any inherent effect of that particular predictor on the response variable, but only a marginal or partial effect, given whatever other correlated predictor variables are included in the model). Ridge regression adds a small bias factor to the variables in order to alleviate this problem.

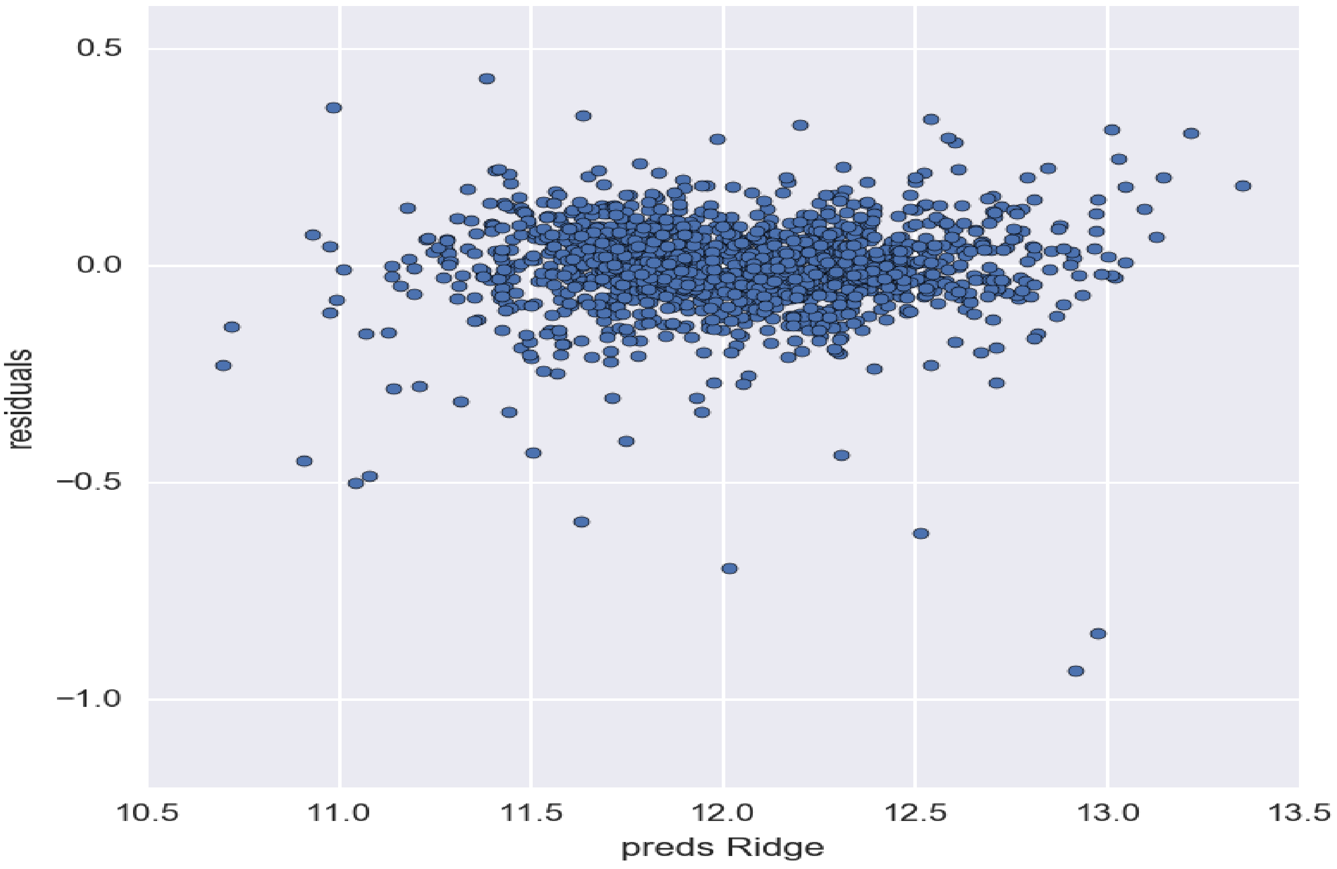
The main tuning parameter for the Ridge model is alpha - a regularization parameter that measures how flexible the model is. The higher the regularization the less prone my model will be to overfit. However, it will also lose flexibility and might not capture all the signal in the data.



Note the U-ish shaped curve above. When alpha is too large the regularization is too strong and the model cannot capture all the complexities in the data. If, however we let the model be too flexible (alpha small) the model begins to overfit. A value of alpha = 10 is about right based on the plot above. So, for the Ridge regression we get a rmse of about 0.127.

```
cv_ridge.min(): 0.12733734668670765
```

Let's look at the residuals of the model as well.



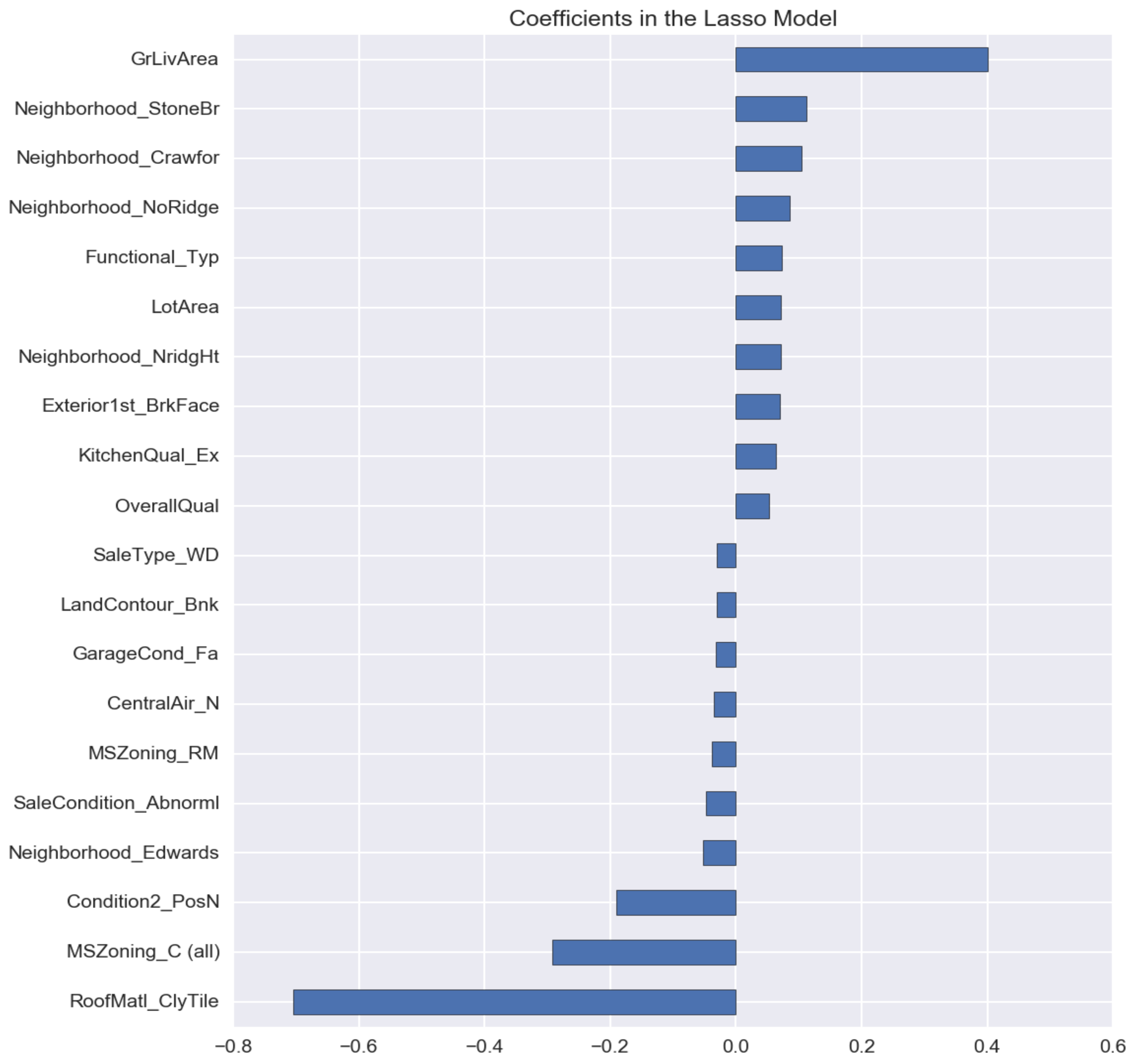
Now let's try out the Lasso model. In statistics and machine learning, Lasso (least absolute shrinkage and selection operator) is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the statistical model it produces.

I will do a slightly different approach here and use the built-in Lasso CV to figure out the best alpha for me. For some reason the alphas in Lasso CV are really the inverse of the alphas in Ridge.

```
rmse_cv(model_lasso).mean(): 0.12314421090977427
```

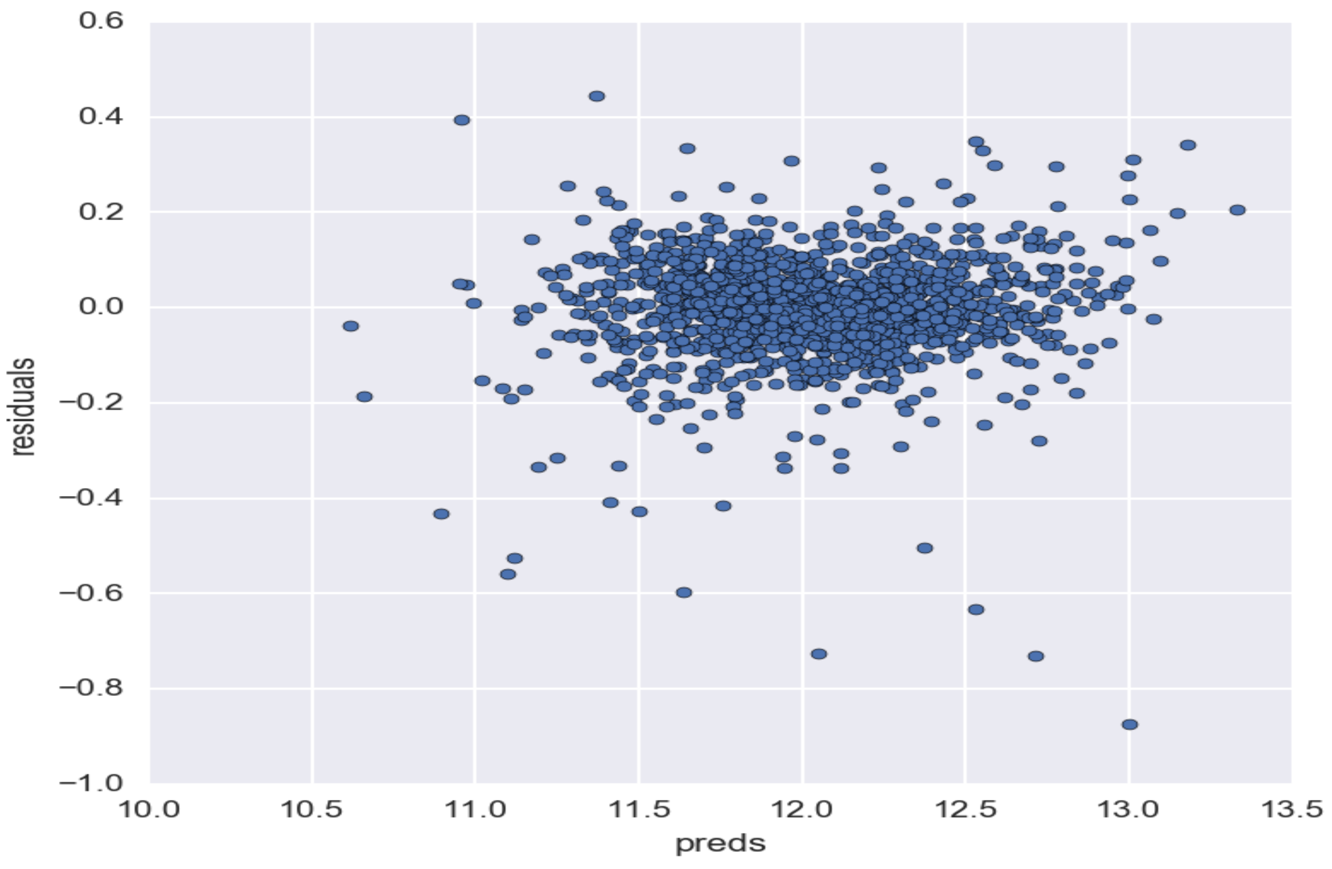
The Lasso performs even better than the Ridge so I will just use this one to predict on the test set. Another neat thing about the Lasso is that it does feature selection for me i.e. setting coefficients of features it deems unimportant to zero. One thing to note here however is that the features selected are not necessarily the "correct" ones - especially since there are a lot of collinear features in this dataset. One idea to try here is run Lasso a few times on bootstrapped samples and see how stable the feature selection is.

I can also look at what the most important coefficients are:



The most important positive feature is GrLivArea - the above ground area by area square feet. This makes sense. Then a few other location and quality features contributed positively. Some of the negative features make less sense and would be worth considering more - it seems like they might have come from unbalanced categorical variables.

Also, note that unlike the feature importance I'd get from a random forest, these are actual coefficients in my model - so I can say precisely why the predicted price is what it is. The only issue here is that we log transformed both the target and the numeric features so the actual magnitudes are a bit hard to interpret.



The above residual plot looks good.

To wrap it up let's predict on the test set after implementing Elastic net regularization model.

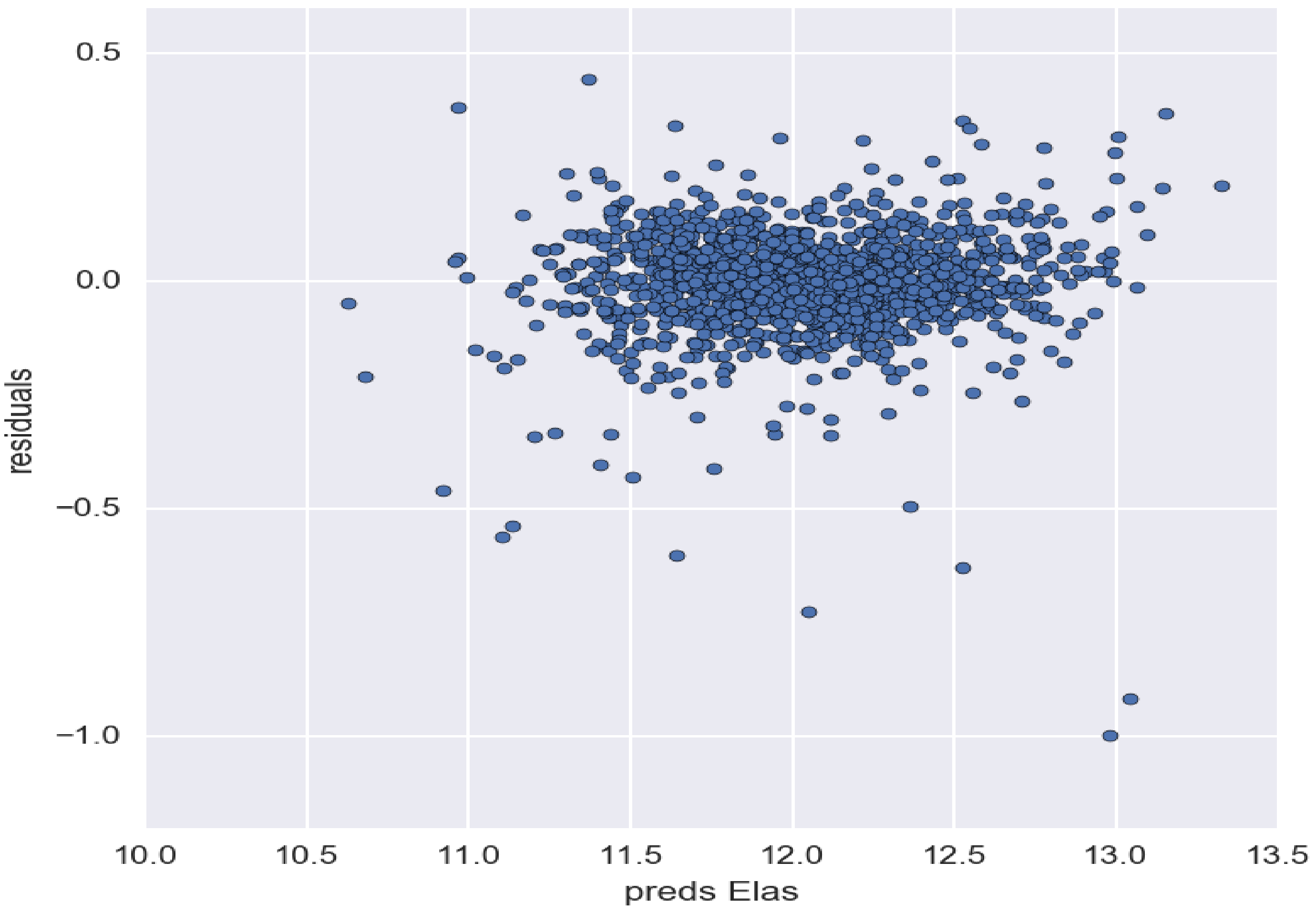
In statistics and in the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods.

```
model_elas = ElasticNet (alpha=0.001, l1_ratio=0.5, fit_intercept=True, normalize=False,
precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False,
positive=False, random_state=None, selection='cyclic').fit(X_train, y)
```

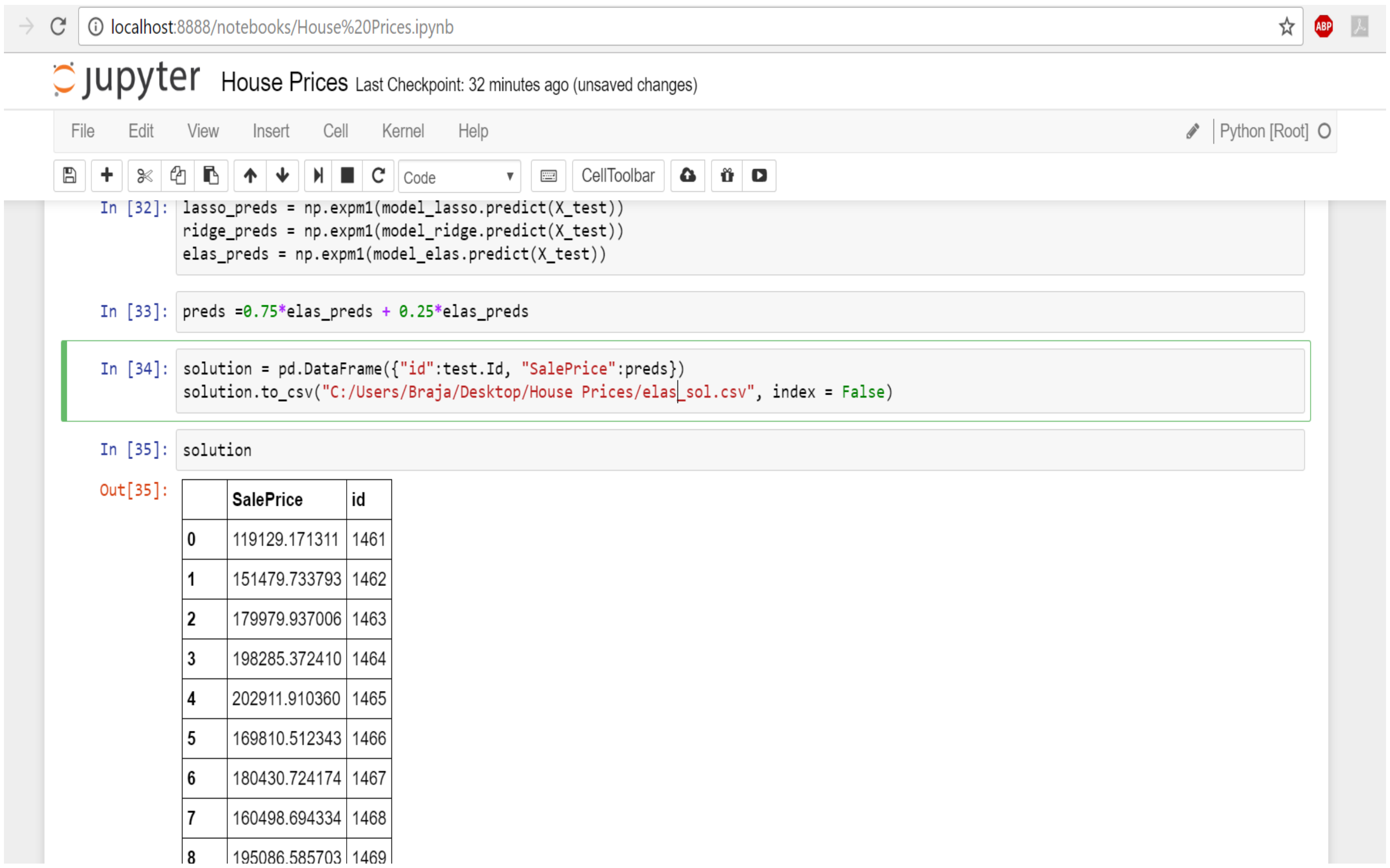
```
    rmse_cv(model_elas).mean(): 0.12289939197911479
```

```
elas_preds = np.expm1(model_elas.predict(X_test))

preds =0.75*elas_preds + 0.25*elas_preds

solution = pd.DataFrame({"id":test.Id, "SalePrice":preds})
```

Let's look at the residuals as well:



Finally, the predicted prices of each home in Ames, Iowa is given by the dataframe "solution" as shown below. The file containing SalePrice and id is exported in csv format.


localhost:8888/notebooks/House%20Prices.ipynb
jupyter House Prices Last Checkpoint: 32 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Help
Python [Root]
Code
CellToolbar

```
In [32]: lasso_preds = np.expm1(model_lasso.predict(X_test))
         ridge_preds = np.expm1(model_ridge.predict(X_test))
         elas_preds = np.expm1(model_elas.predict(X_test))

In [33]: preds =0.75*elas_preds + 0.25*elas_preds

In [34]: solution = pd.DataFrame({"id":test.Id, "SalePrice":preds})
         solution.to_csv("C:/Users/Braja/Desktop/House Prices/elas_sol.csv", index = False)

In [35]: solution

Out[35]:
```

	SalePrice	id
0	119129.171311	1461
1	151479.733793	1462
2	179979.937006	1463
3	198285.372410	1464
4	202911.910360	1465
5	169810.512343	1466
6	180430.724174	1467
7	160498.694334	1468
8	195086.585703	1469

Appendix:

```
import pandas as pd

import numpy as np

import seaborn as sns


import matplotlib

import matplotlib.pyplot as plt

from scipy.stats import skew

from scipy.stats.stats import pearsonr


%config InlineBackend.figure_format = 'retina' #set 'png' here when working on notebook

%matplotlib inline


train = pd.read_csv("C:/Users/Braja/Desktop/House Prices/train.csv")

test = pd.read_csv("C:/Users/Braja/Desktop/House Prices/test.csv")


all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'],

                  test.loc[:,'MSSubClass':'SaleCondition']))

all_data.head()


matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)

prices = pd.DataFrame({"price":train["SalePrice"], "log(price + 1)":np.log1p(train["SalePrice"])})

prices.hist()
```

```
#log transform the target:
train["SalePrice"] = np.log1p(train["SalePrice"])


#log transform skewed numeric features:
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index
skewed_feats = train[numeric_feats].apply(lambda x: skew(x.dropna())) #compute skewness
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index
all_data[skewed_feats] = np.log1p(all_data[skewed_feats])


all_data = pd.get_dummies(all_data)


#filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())


#creating matrices for sklearn:
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.SalePrice
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, LassoCV, LassoLarsCV
from sklearn.cross_validation import cross_val_score


def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, y, scoring="neg_mean_squared_error", cv = 5))
    return(rmse)
model_ridge = Ridge()
alphas = [0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 50, 75]
cv_ridge = [rmse_cv(Ridge(alpha = alpha)).mean()
            for alpha in alphas]
cv_ridge = pd.Series(cv_ridge, index = alphas)
cv_ridge.plot(title = "Validation - Just Do It")
plt.xlabel("alpha")
plt.ylabel("rmse")

cv_ridge.min()
model_ridge = Ridge(alpha = 5).fit(X_train, y)

#let's look at the residuals as well:
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)
preds_ridge = pd.DataFrame({"preds Ridge":model_ridge.predict(X_train), "true":y})
preds_ridge["residuals"] = preds_ridge["true"] - preds_ridge["preds Ridge"]
preds_ridge.plot(x = "preds Ridge", y = "residuals",kind = "scatter")

model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(X_train, y)
rmse_cv(model_lasso).mean()
coef = pd.Series(model_lasso.coef_, index = X_train.columns)

print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other " +
str(sum(coef == 0)) + " variables")
```

Output: Lasso picked 110 variables and eliminated the other 178 variables

```
imp_coef = pd.concat([coef.sort_values().head(10),
                     coef.sort_values().tail(10)])


matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)

imp_coef.plot(kind = "barh")

plt.title("Coefficients in the Lasso Model")
```

#let's look at the residuals as well:

```
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)

preds = pd.DataFrame({"preds":model_lasso.predict(X_train), "true":y})

preds["residuals"] = preds["true"] - preds["preds"]

preds.plot(x = "preds", y = "residuals",kind = "scatter")
```

model_elas = ElasticNet(alpha=0.001, l1_ratio=0.5, fit_intercept=True, normalize=False, precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic').fit(X_train, y)

rmse_cv(model_elas).mean()

#let's look at the residuals as well:

```
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)

preds_elas = pd.DataFrame({"preds Elas":model_elas.predict(X_train), "true":y})

preds_elas["residuals"] = preds_elas["true"] - preds_elas["preds Elas"]

preds_elas.plot(x = "preds Elas", y = "residuals",kind = "scatter")
```

```
lasso_preds = np.expm1(model_lasso.predict(X_test))

ridge_preds = np.expm1(model_ridge.predict(X_test))

elas_preds = np.expm1(model_elas.predict(X_test))


preds =0.75*elas_preds + 0.25*elas_preds

solution = pd.DataFrame({"id":test.Id, "SalePrice":preds})

solution.to_csv("C:/Users/Braja/Desktop/House Prices/elas_sol.csv", index = False)
```

Reference:

https://www.kaggle.com/c/house-prices-advanced-regression-techniques

https://www.kaggle.com/zyedpls/house-prices-advanced-regression-techniques/regularized-linear-models