EE379K: Lab 2

Group Members: Irfan Hasan (ih3976), Peter Zhang (yz7724)

Question 1 (Irfan)

In [3]:
```
import numpy as np
import sympy
import sys

v1 = np.array([1, 2, 3, 4])
v2 = np.array([0, 1, 0, 1])
v3 = np.array([1, 4, 3, 6])
v4 = np.array([2, 11, 6, 15])
matrix = np.vstack((v1, v2, v3, v4))

print('\n-------- Q1 --------\n')
print('A vector inside S is any vector that is a linear combination of v1, v
print('Thus [0,2,0,2] is a vector inside S')
print('A vector not inside S is [1,1,1,1]\n')
print('To find a perpendicular vector, calculate the nullspace of the columr
print('Nullspace of column space:')
perp = sympy.Matrix([v1, v2, v3, v4]).nullspace()
for v in perp:
    for e in v:
        sys.stdout.write(str(e) + " ")
    sys.stdout.write("\n")
    sys.stdout.flush()

print('\nTo check if a new vector is in S we can see if the vector is a line

print('\n-------- Q2 --------\n')
rank = np.linalg.matrix_rank(matrix)
print('The dimension of S is {}'.format(rank))

print('\n-------- Q3 --------\n')
print('A QR decomposition provides an orthogonal basis for the column space
print('If the rank of A is n, then the first n columns of q form a basis for

q,r = np.linalg.qr(matrix.T) #need column space

orth = q[:, :rank]
print('Orthonormal basis for S: \n')
print(orth)

print('\n-------- Q4 --------\n')
print('When minimizing the distance between x in S and z [1, 0, 0, 0] is the
print('as finding the projection of z onto subspace S.')
z = np.array([1, 0, 0, 0])
v1 = orth[:, 0]
v2 = orth[:, 1]
x = (np.dot(z,v1) / np.dot(v1,v1)) * v1 + (np.dot(z,v2)/np.dot(v2,v2))*v2
print('x is {0}'.format(x))
```

```
-------- Q1 --------

A vector inside S is any vector that is a linear combination of v1, v2, v
3, v4
Thus [0,2,0,2] is a vector inside S
A vector not inside S is [1,1,1,1]
```

```
To find a perpendicular vector, calculate the nullspace of the column spa
ce.
Nullspace of column space:
-3 0 1 0
-2 -1 0 1

To check if a new vector is in S we can see if the vector is a linear com
bination of the vectors in S


-------- Q2 --------

The dimension of S is 2

-------- Q3 --------

A QR decomposition provides an orthogonal basis for the column space of A
If the rank of A is n, then the first n columns of q form a basis for the
column space of A.

Orthonormal basis for S:

[[-0.18257419  0.2236068 ]
 [-0.36514837 -0.67082039]
 [-0.54772256  0.67082039]
 [-0.73029674 -0.2236068 ]]

-------- Q4 --------

When minimizing the distance between x in S and z [1, 0, 0, 0] is the sam
e
as finding the projection of z onto subspace S.
x is [ 0.08333333 -0.08333333  0.25        0.08333333]
```

Question 2 (Peter)

1. Top 10 common words in ICML papers

Our machines runs out of memory when processing such large number of PDFs; therefore, all answers below are based on the first 50 PDFs.

In [29]:
```
import urllib2
import re
from bs4 import BeautifulSoup

from cStringIO import StringIO
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
import os
import sys, getopt

import operator

#converts pdf, returns its text content as a string
def convert(fname, pages=None):
    if not pages:
        pagenums = set()
    else:
        pagenums = set(pages)

    output = StringIO()
    manager = PDFResourceManager()
    converter = TextConverter(manager, output, laparams=LAParams())
    interpreter = PDFPageInterpreter(manager, converter)

    infile = file(fname, 'rb')
    for page in PDFPage.get_pages(infile, pagenums):
        interpreter.process_page(page)
    infile.close()
    converter.close()
    text = output.getvalue()
    output.close
    return text

words = {}  # word frequency
response = urllib2.urlopen('http://proceedings.mlr.press/v70')
html = BeautifulSoup(response.read(), 'html.parser')
links = html.find_all('a', text=re.compile('Download PDF'))

for i in range(50):
    link = links[i]
    fname = link['href'].split('/')[-1]
    fname = "./pdfs/" + fname

    # uncomment this block to download the pdfs
    ''' # download all pdfs/
    pdf = urllib2.urlopen(link['href']).read()
    with open('./pdfs/'+fname, 'wb') as f:
        f.write(pdf)
    '''

    data = convert(fname)
    for word in re.findall(r"[a-zA-Z]+", data):
        word = word.lower()
        if word in words:
```

```
            words[word] += 1
        else:
            words[word] = 1
```

In [30]:
```
top10 = dict(sorted(words.iteritems(), key=operator.itemgetter(1), reverse=T
top10
```

Out[30]:
```
{'a': 7836,
 'and': 7152,
 'cid': 4819,
 'for': 4259,
 'in': 6076,
 'is': 4510,
 'of': 8353,
 'the': 16374,
 'to': 5504,
 'we': 3669}
```

2. Calculate the entropy of Z, a randomly selected word in randomly selected ICML paper

In [36]:
```
import scipy as sc

entropy = sc.stats.entropy(words.values())

print 'Entropy is {0}'.format(entropy)
```

```
Entropy is 7.04991170588
```

3. Synthesize a random paragraph using the marginal distribution over words

Due to amount of information to process in all pdfs, only one PDF is used.

In [37]:
```
import operator
import random

sorted_w = sorted(words.items(), key=operator.itemgetter(1), reverse=True)
total = sum(words.values())
```

In [38]:
```
import sys

for i in range(50):
    n = random.randint(0, total)
    count = 0
    for t in sorted_w:
        count += t[1]
        if count >= n:
            sys.stdout.write(t[0] + " ")
            break;
sys.stdout.flush()
```

```
that a network be mic nement adapted operators regret from ve same and re
gions be this the an of of other proximation loss do ing replacement uai
see correspond by gence and with mse the neg and iteration effectiveness
class project models we parameters c hind application al previously size
```

Question 3 (Irfan & Peter)

Follow the data preprocessing steps from https://www.kaggle.com/apapiu/house-prices-advanced-regression-techniques/regularized-linear-models (https://www.kaggle.com/apapiu/house-prices-advanced-regression-techniques/regularized-linear-models)

In [2]:
```
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
%matplotlib inline
```

In [3]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

In [4]:
```
train.head()
```

Out[4]:

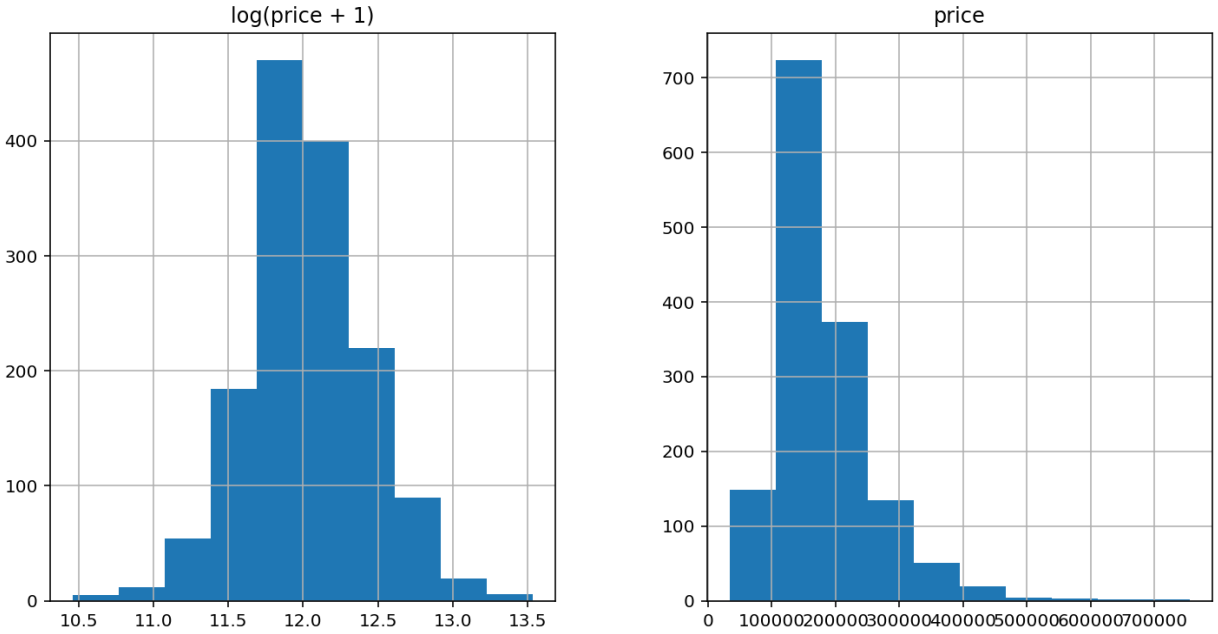
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilitie
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPu
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPu
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPu
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPu
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPu

5 rows × 81 columns

In [5]:
```
all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'],
                      test.loc[:,'MSSubClass':'SaleCondition']))
```

In [6]:
```
matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)
prices = pd.DataFrame({"price":train["SalePrice"], "log(price + 1)":np.log1p
prices.hist()
```

Out[6]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a0dfd3110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a0ec30410>]],
dtype=object)
```



In [7]:
```
#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())) #compu
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index

all_data[skewed_feats] = np.log1p(all_data[skewed_feats])
```

In [8]:
```
all_data = pd.get_dummies(all_data)
```

In [9]:
```
#filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())
```

In [10]:
```
#creating matrices for sklearn:
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.SalePrice
```

Models

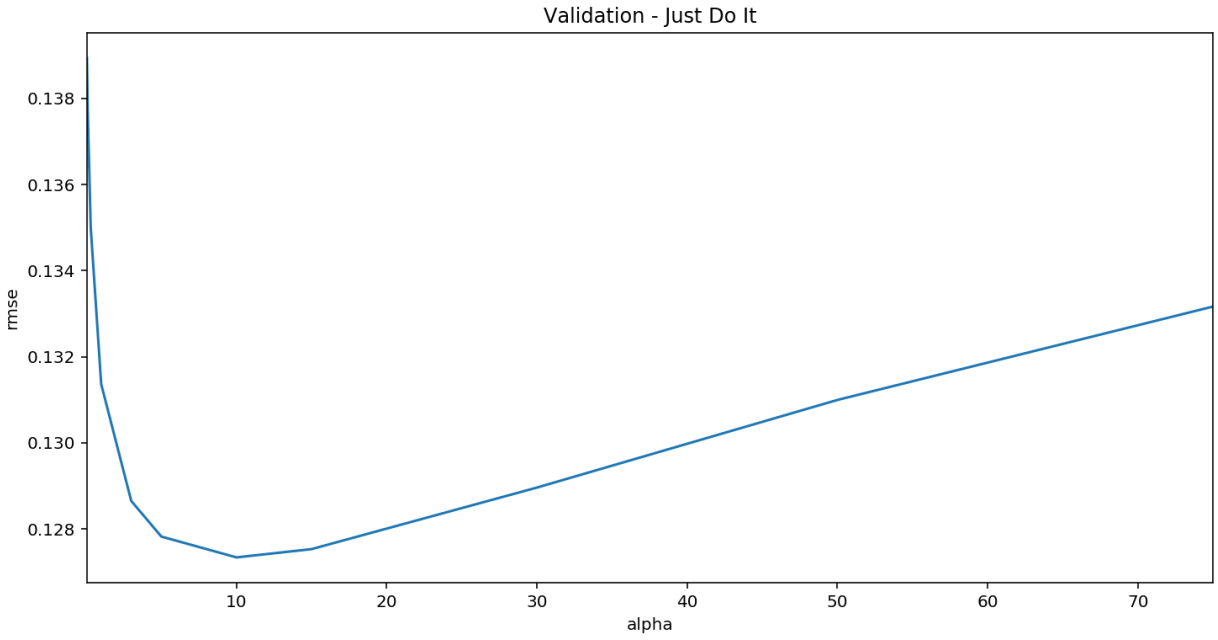
In [11]:
```
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, LassoCV, LassoL
from sklearn.model_selection import cross_val_score

def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, y, scoring="neg_mean_squa
    return(rmse)
```

In [12]:
```
model_ridge = Ridge()
```

In [13]:
```
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]
```

In [14]:
```
cv_ridge = pd.Series(cv_ridge, index = alphas)
cv_ridge.plot(title = "Validation - Just Do It")
plt.xlabel("alpha")
plt.ylabel("rmse")
```

Out[14]: Text(0,0.5,u'rmse')



In [15]:
```
cv_ridge.min()
```

Out[15]: 0.12733734668670763

In [16]:
```
model_lasso = LassoCV(alphas = [1, 0.1, 0.001, 0.0005]).fit(X_train, y)
```

In [17]:
```
rmse_cv(model_lasso).mean()
```

Out[17]: 0.12314421090977448

In [18]:
```
coef = pd.Series(model_lasso.coef_, index = X_train.columns)
```

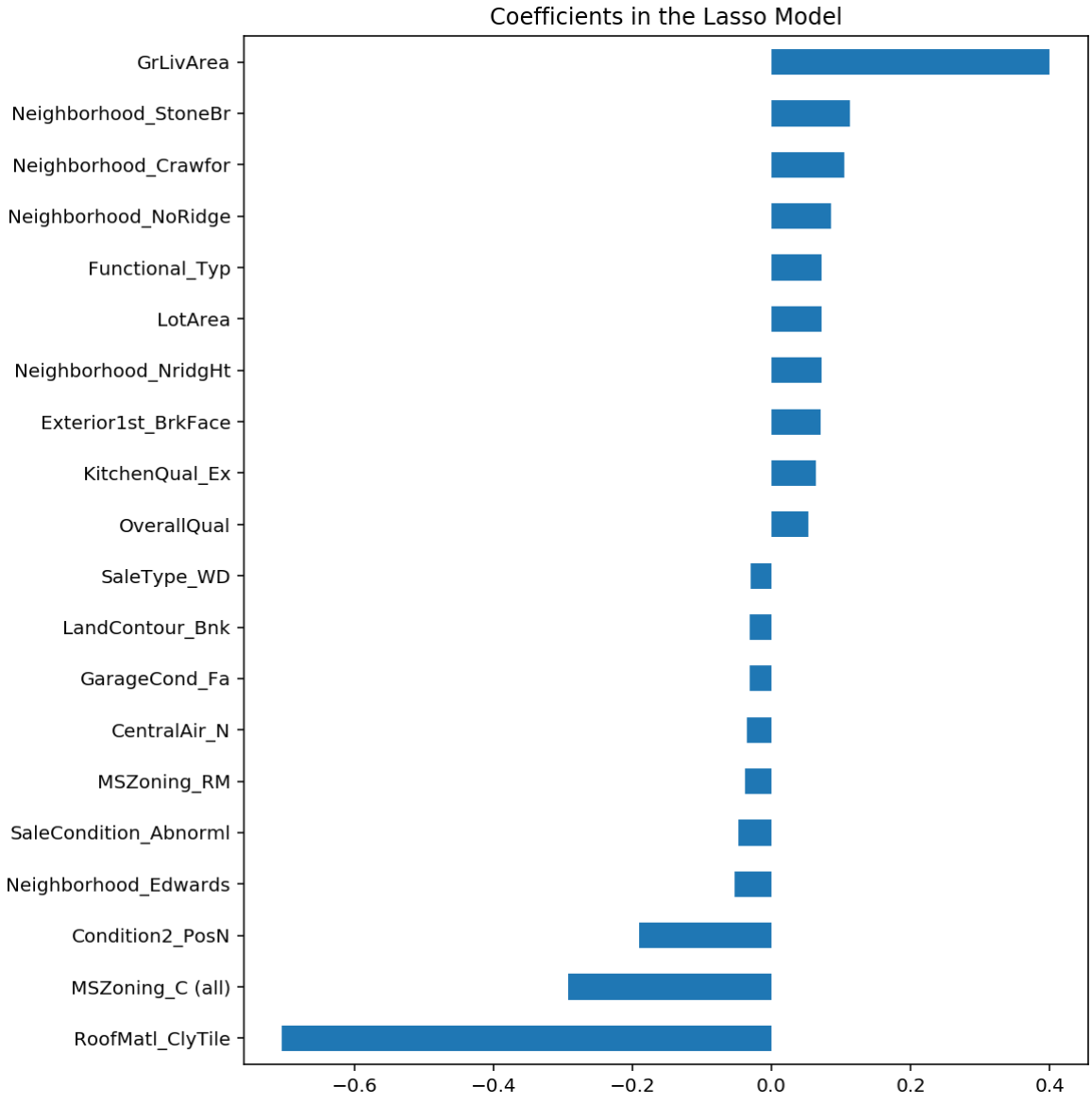
In [19]:
```
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the
```

```
Lasso picked 111 variables and eliminated the other 177 variables
```

In [20]:
```
imp_coef = pd.concat([coef.sort_values().head(10),
                      coef.sort_values().tail(10)])
```

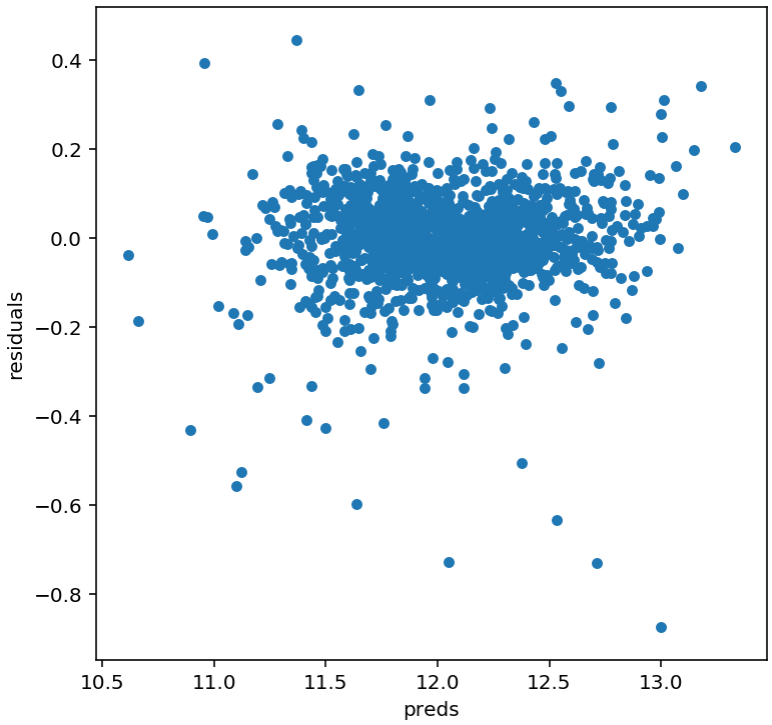
In [21]:
```
matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Lasso Model")
```

Out[21]: Text(0.5,1,u'Coefficients in the Lasso Model')



In [22]:
```
#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")
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1086c8dd0>

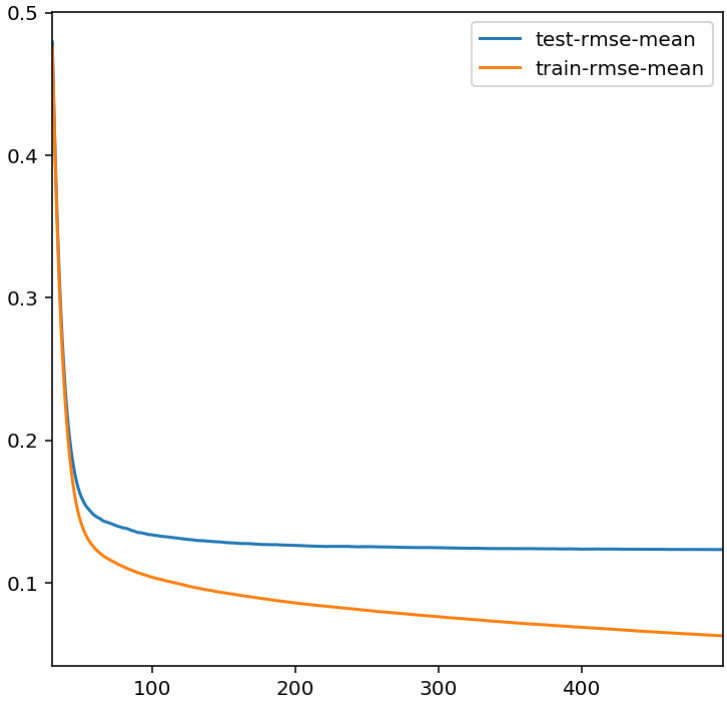


In [23]:
```
import xgboost as xgb
```

In [24]:
```
dtrain = xgb.DMatrix(X_train, label = y)
dtest = xgb.DMatrix(X_test)

params = {"max_depth":2, "eta":0.1}
model = xgb.cv(params, dtrain,  num_boost_round=500, early_stopping_rounds=1
```

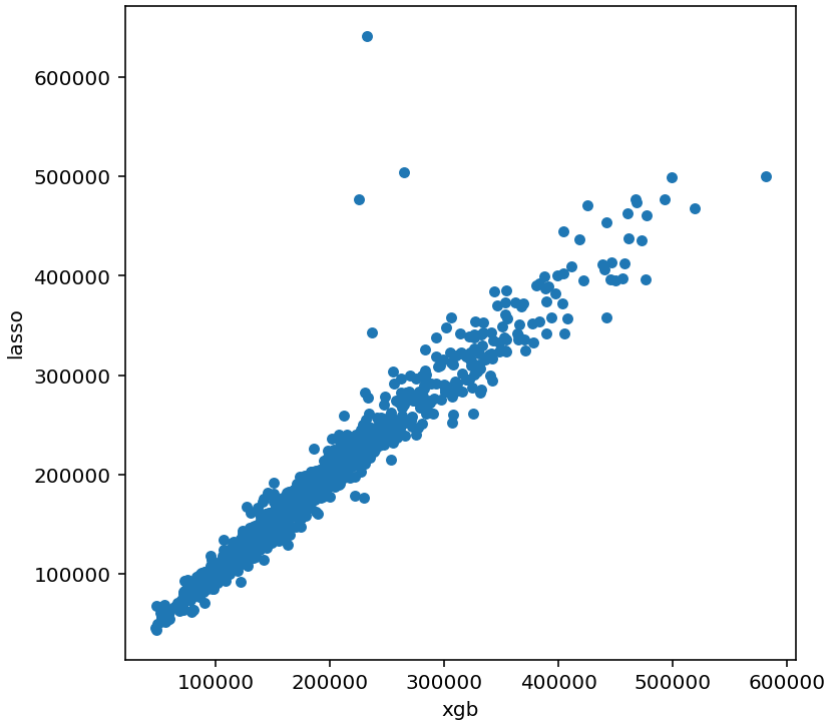
In [25]:
```
model.loc[30:,["test-rmse-mean", "train-rmse-mean"]].plot()
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a18d3e3d0>
```



In [64]:
```
model_xgb = xgb.XGBRegressor(n_estimators=360, max_depth=2, learning_rate=0.
model_xgb.fit(X_train, y)

xgb_preds = np.expm1(model_xgb.predict(X_test))
lasso_preds = np.expm1(model_lasso.predict(X_test))

predictions = pd.DataFrame({"xgb":xgb_preds, "lasso":lasso_preds})
predictions.plot(x = "xgb", y = "lasso", kind = "scatter")
```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x1a19e87950>



In [65]:
```
preds = 0.3*lasso_preds + 0.7*xgb_preds

solution = pd.DataFrame({"id":test.Id, "SalePrice":preds})
solution.to_csv("ridge_sol.csv", index = False)
```

Make alpha = 0.1 and submit the prediction.

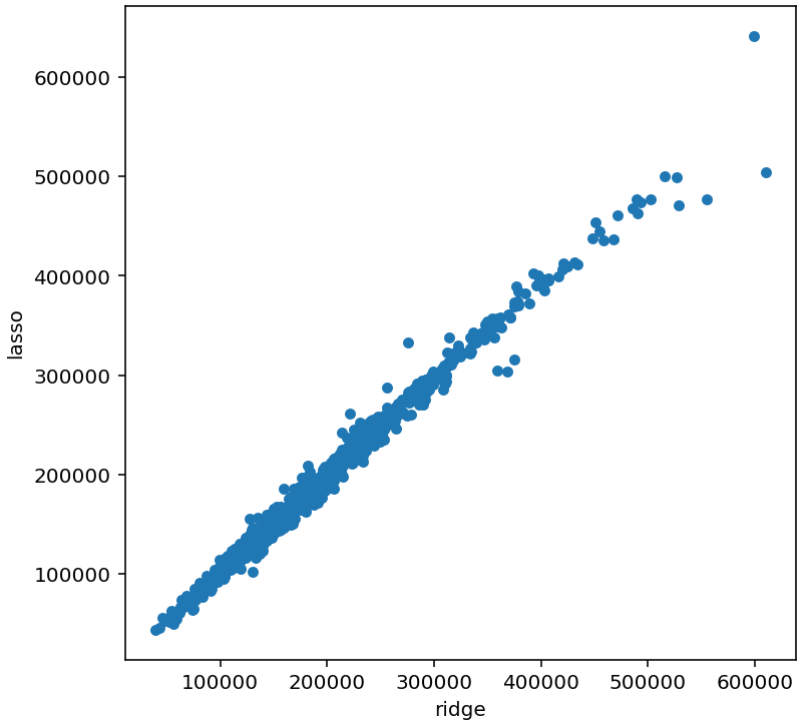
In [66]:
```
model_r = Ridge(alpha=0.1).fit(X_train, y)
pred_r = np.expm1(model_r.predict(X_test))

solution = pd.DataFrame({"id":test.Id, "SalePrice":pred_r})
solution.to_csv("0_1_ridge_sol.csv", index = False)
```

The ridge regession with alpha = 0.1 received a score of 0.13029.

Our Model

In [85]:
```
ridge_model = RidgeCV(alphas=[1, 0.1, 0.001, 0.0005]).fit(X_train, y)
lasso_model = LassoCV(alphas=[1, 0.1, 0.001, 0.0005]).fit(X_train, y)

ridge_pred = np.expm1(ridge_model.predict(X_test))
lasso_pred = np.expm1(lasso_model.predict(X_test))

predictions = pd.DataFrame({"ridge":ridge_pred, "lasso":lasso_preds})
predictions.plot(x='ridge', y='lasso', kind='scatter')
```

Out[85]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1bc3dad0>



In [87]:
```
final_pred = 0.8*lasso_pred + 0.2*ridge_pred

solution = pd.DataFrame({"id":test.Id, "SalePrice":pred_r})
solution.to_csv("mysol.csv", index = False)
```

The model received a score of 0.12120.