Homework 1

Javier Palomares

September 22, 2019

Problem 1

1. Watch: Vectors, what even are they? | Essence of linear algebra, Chapter 1 https://www.youtube.com/watch?v=fNk_zzaMoSs (https://www.youtube.com/watch?v=fNk_zzaMoSs)
2. Chapter 2 https://www.youtube.com/watch?v=k7RM-ot2NWY (https://www.youtube.com/watch?v=k7RM-ot2NWY) In this video, in Minute 2:51, two vectors are written by their coordinates. Compute the coordinates of the vector that is the sum of twice the first (left one) plus the second (right one).

In [92]:
```
import numpy as np
first = np.array([-.8,1.3]);
second = np.array([3.1,-2.9]);
coordinates = 2*first + second
print(np.asarray(coordinates))
```

```
[ 1.5 -0.3]
```

Are the vectors $[1, 1]$, $[1, 0]$, $[0, 1]$ linearly dependent? Write the third as a linear combination of the first two.

The vector are not linearly independent. $[1, 1]$ can be written as $[1, 0] + [0, 1]$ so it's a linear combination of the other two. The third can be written as
$[0, 1] = [1, 1] - [1, 0]$

3.Chapter 3 https://www.youtube.com/watch?v=kYB8IZa5AuE (https://www.youtube.com/watch?v=kYB8IZa5AuE) At minute 6:36, a Matrix and a vector is given. Write down how this matrix transforms this vector. Based on what you learned, write a matrix that rotates the 2D space by 90 degrees clockwise.

In [26]:
```
A = np.array([[3, 2],
              [-2, 1]])
v = np.array([5,7])
np.matmul(A,v)
```

Out[26]: `array([29, -3])`

The matrix transforms the vector $[5, 7]$ to $5 * [3, -2] + 7 * [2, 1]$ The matrix transforms vectors to linear combinations of $[3, -2]$ and $[2, 1]$

The matrix $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ rotates the 2D space by 90 degrees clockwise.

4.Chapter 4 https://www.youtube.com/watch?v=XkY2DOUCWMU (https://www.youtube.com/watch?v=XkY2DOUCWMU) At 3.34 a composition matrix is shown. Apply this Composition linear transformation to the vector [1; 2]T and write the transformed vector.

In [37]:
```
composition_matrix= np.array([[1, -1],
              [1, 0]]);
v = np.array([1,2]);
np.matmul(composition_matrix,v)
```

Out[37]:
```
array([-1,  1])
```

5.The determinant | Essence of linear algebra, chapter 6 https://www.youtube.com/watch?v=Ip3X9LOh2dk (https://www.youtube.com/watch?v=Ip3X9LOh2dk) In 9.40 there is a quiz question: Write your answer in one sentence.

$det(M_1 M_2) = det(M_1)det(M_2)$ because M_1 and M_2 are linear transformations, so M_1 scales the are of the unit square by $det(M_1)$, then M_2 scales the result by $det(M_2)$ so the unit square's area is scaled in total by $det(M_1)det(M_2)$.

6.Dot products and duality | Essence of linear algebra, chapter 9 https://www.youtube.com/watch?v=LyGKycYT2v0 (https://www.youtube.com/watch?v=LyGKycYT2v0)

Project the vector $[1, 2, 3]^T$ on the vector $[1, 1, 1]$. Write the projected vector.

In [81]:
```
a = np.array([1,2,3])
b = np.array([1,1,1])
proj_a_b = np.dot(a,b) / np.linalg.norm(b) * b
print(proj_a_b)
```

```
[3.46410162 3.46410162 3.46410162]
```

Project the vector $[1, 2, 3]^T$ on the span of the vectors $[1, 0, 0]^T$ and $[1, 1, 0]^T$. Write the projected vector.

$proj_v = \frac{v \cdot a}{a \cdot a} a + \frac{v \cdot b}{b \cdot b} b$

In [77]:
```
v = np.array([1,2,3])
a = np.array([1,0,0])
b = np.array([1,1,0])

print(np.dot(v,a)/ np.dot(a,a) * a + np.dot(v,b)/ np.dot(b,b) * b)
```

```
[ 2.5  1.5  0. ]
```

Problem 2: Linear Algebra in Python.

You can use all Python functions to solve this problem.

1.Consider the linear subspace $S = span(v1, v2, v3, v4)$ where
$v1 = [1; 2; 3; 4]; v2 = [0; 1; 0; 1]; v3 = [1; 4; 3; 6]; v4 = [2; 11; 6; 15]$.

In [42]:
```
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])
```

Create a vector inside S different from v1; v2; v3; v4.

In [45]:
```
v1+v2+4*v3
```

Out[45]:
```
array([ 5, 19, 15, 29])
```

Create a vector not in S.

$[0, 0, 0, 1]$ is a vector not in S. It cannot be writtend as a linear combination of $v1, v2, v3,$ and $v4$

How would you check if a new vector is in S?

A vector v is in S if there a consistent solution a, b, c, d to $v = a * v1 + b * v2 + c * v3 + d * v4$

2.Find the dimension of the subspace S.

In [53]:
```
M = np.matrix([v1,v2,v3,v4]).transpose()
```

In [57]:
```
import sympy
_, inds = sympy.Matrix(M).T.rref()
inds
```

Out[57]:
```
(0, 1)
```

The first 2 vectors are linearly independent. The other 2 are linear combinations of the first 2. So the dimension of S is 2.

3.Find an orthonormal basis for the subspace S.

Since the first 2 vectors are independent, they form the basis for S. To find an orthonormal basis, we need to decompose v_1 and v_2 into orthonormal vectors n_1 and n_2

I'll let $n_1 = \frac{v_2}{|v2|}$ then we need $n_2 = av_1 + bv_2$ so that $n_1^T n_2 = 0$

In [61]:
```
n1 = v2 /np.linalg.norm(v2)
```

Since we need $n_1^T n_2 = 0$ we end up with needing a solution to $3a + b = 0$. I'll use $a = -1, b = 3$ then normalize n_2

In [70]:
```
a = -1
b = 3
n2 = a *v1 + b * v2
n2 = n2/ np.linalg.norm(n2)
```

An orthonal basis for S is given by span(n1,n2)

In [78]:
```
print(n1)
print(n2)
```

```
[0.         0.70710678 0.         0.70710678]
[-0.28867513  0.28867513 -0.8660254  -0.28867513]
```

4.Solve the optimization problem $min_{x \epsilon S} ||x - z^*||_2$ where $z^* = [1, 0, 0, 0]$.

Since x is in S, we can express x as $x = an_1 + bn_2$ Then the optimization is to minimize

$$f = ||an_1 + bn_2 - z^*||_2$$

$$f=|| a \left(\begin{array}{c} 0 \\ 1 \\ 0 \\ 1 \end{array}\right)

- b \left(\begin{array}{c} -1 \ 1 \ -3 \ -1

  ## \end{array}\right)

  \left(

$$\begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix}$$

  \right) || _2 $$

$$f = || \begin{pmatrix} -b-1 \\ a+b \\ -3b \\ a-b \end{pmatrix} ||_2$$

We can optimize $F = f^2$ to also find the optimum of f. Taking the gradient of F, we find

$$\nabla F = \begin{pmatrix} 4a \\ 22b+2 \end{pmatrix}$$

Setting equal to zero, we find $a = 0, b = \frac{-1}{11}$ So the optimal solution is at

$x = \frac{-1}{11} \begin{pmatrix} -1 \\ 1 \\ -3 \\ -1 \end{pmatrix}$

5.(Tricky) Is there a relation of this optimization problem with linear regression? Discuss.

The optimization done in problem 4 is exactly the optimization we would solve when optimizing the least squares loss function for linear regression.

Problem 3: Starting supervised learning in Kaggle.

1. Lets start with our first Kaggle submission in a playground regression competition. Make an account to Kaggle and find [https://www.kaggle.com/c/house-prices-advanced-regression-techniques/ (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/)](https://www.kaggle.com/c/house-prices-advanced-regression-techniques/)

I've created an account in the past. My user name is javierpalomares

2.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). Then, split the data into a train and test set and fit a linear regression model (without any regularization) to make predictions. Report the performance (RMSE) of this model on the train set, the test set and on the Kaggle private leader board. (Hint: remember to exponentiate np.expm1(ypred) your predictions).

In [38]:
```
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
```

In [39]:
```
train = pd.read_csv("./input/train.csv")
test = pd.read_csv("./input/test.csv")
```

In [40]:
```
train.head()
```

Out[40]:

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

5 rows × 81 columns

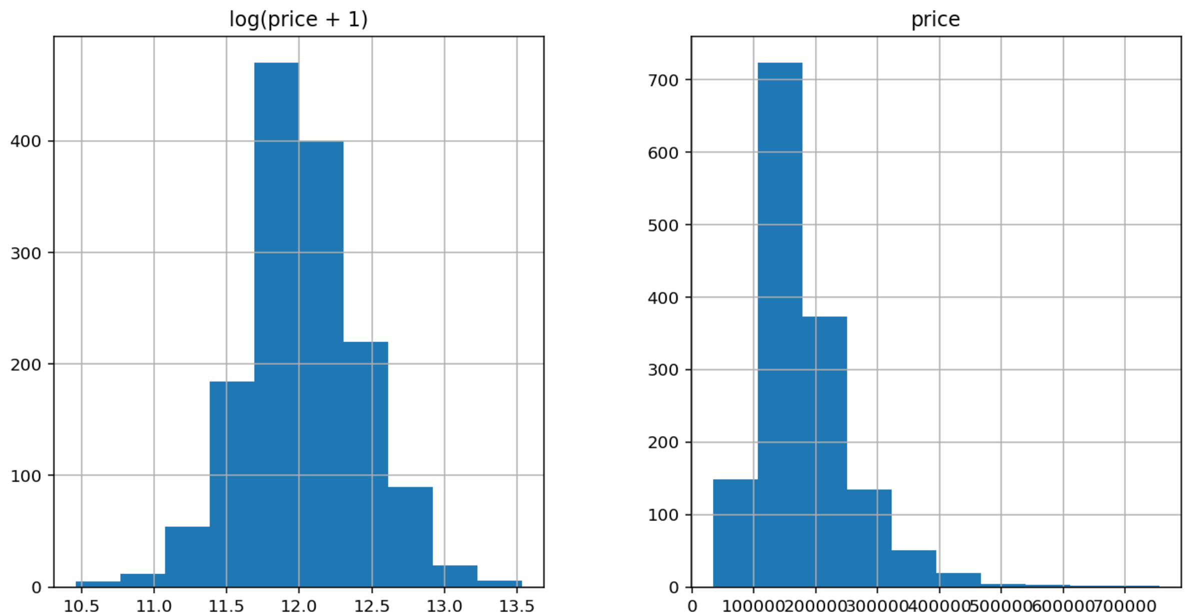
In [41]:
```
all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'],
                      test.loc[:,'MSSubClass':'SaleCondition']))
```

Data preprocessing: We're not going to do anything fancy here:

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

In [42]:
```
matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)
prices = pd.DataFrame({"price":train["SalePrice"], "log(price + 1)":np.log1p(train["SalePrice"])})
prices.hist()
```

Out[42]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001DB4771DC50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001DB47B7D518
>]], dtype=object)
```



In [43]:
```
#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])
```

In [44]:
```
#one hot encode categorical columns
all_data = pd.get_dummies(all_data)
```

In [45]:
```
#filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())
```

In [46]:
```
#creating matrices for sklearn:
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y_train = train.SalePrice
```

Models

First I'll train a linear regression without any regularization

In [47]:
```
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, LassoCV, LassoLar
sCV,LinearRegression
from sklearn.model_selection import cross_val_score

# function to compute the root mean squared error of the cross validation
def rmse_cv(model,X_train,y):
    rmse= np.sqrt(-cross_val_score(model, X_train, y, scoring="neg_mean_square
d_error", cv = 5))
    return(rmse)

def rms(predicted,target):
    return np.sqrt(np.mean((predicted-target)**2))
```

In [48]:
```
linear_model = LinearRegression()
# Train the model using the training sets
linear_model.fit(X_train, y_train)
```

Out[48]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [49]:
```
# Make predictions using the training set
y_pred = linear_model.predict(X_train)
# we took a log of the sale price in the pre processing, so we need to exponen
tiate y_pred and y to get the true sale prices
y_pred_exp = np.expm1(y_pred)
y_exp = np.expm1(y_train)
rmse_linear = rms(y_pred_exp,y_exp)
print("A Linear Regression produced an rmse of ${} on the train set".format(rm
se_linear))
```

```
A Linear Regression produced an rmse of $17717.980852150613 on the train set
```

In [50]:
```
# Now make predictions using the test set
y_pred = linear_model.predict(X_test)
# we took a log of the sale price in the pre processing, so we need to exponen
tiate y_pred and y to get the true sale prices
y_pred_exp = np.expm1(y_pred)
```

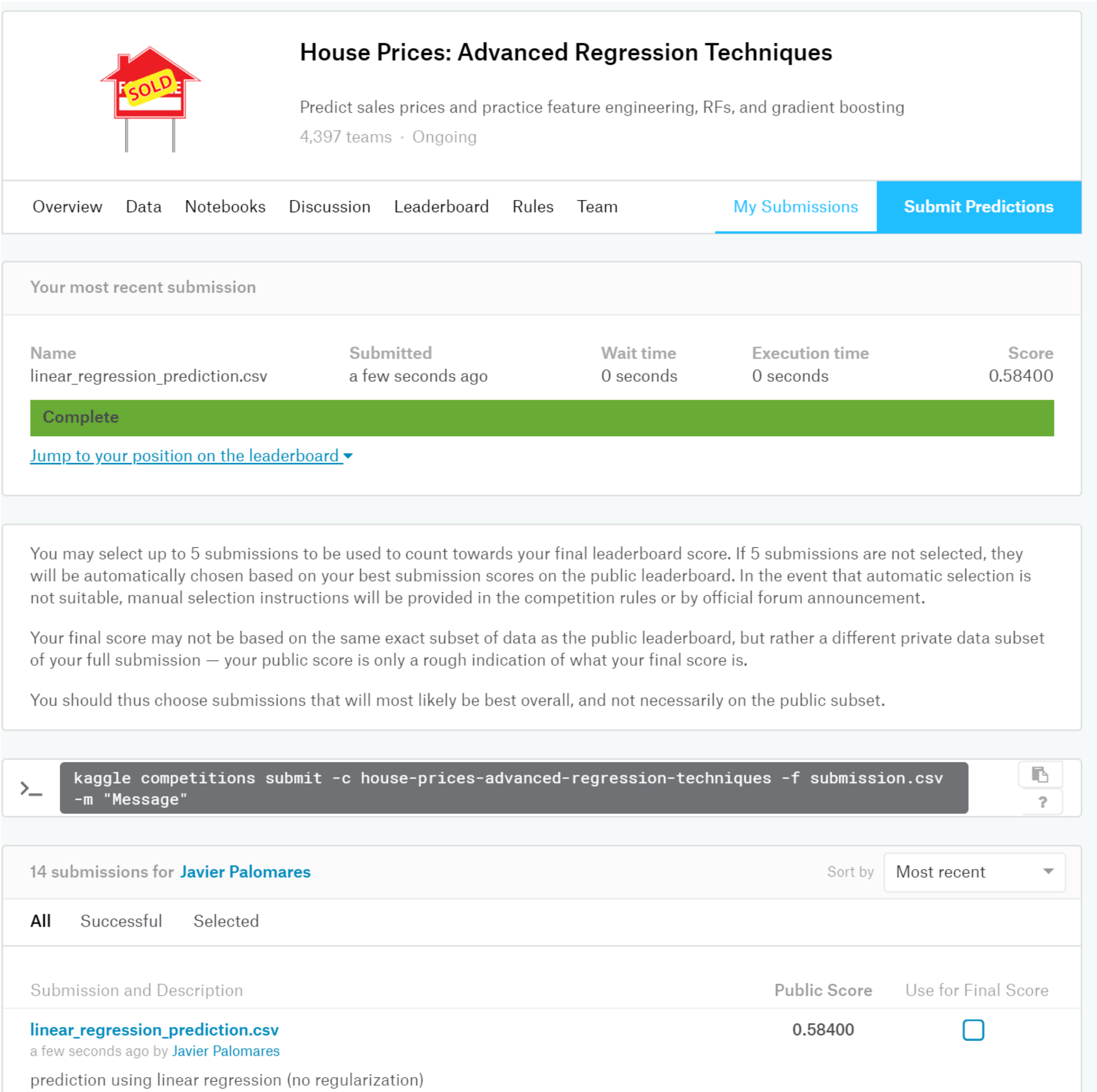
In [51]:
```
def print_predictions(filename,header,ids,y_pred):
    f = open(filename,'w')
    numRows = len(ids)
    f.write(header)
    for i in range(numRows):
        idNum = ids[i]
        y = y_pred[i]
        f.write("{},{}\n".format(idNum,y))
    f.close()
```

In [52]:
```
ids = test['Id']
print_predictions("linear_regression_prediction.csv",'Id,SalePrice\n',ids,y_pr
ed_exp)
```

In [53]:
```
y_pred = linear_model.predict(X_train)
rmse_log = rms(y_pred,y_train)
print(rmse_log)
```

```
0.0916376890882
```

This submission had an rmse on the logs of the sale prices of .5840. For comparison, the rms on the log of the training data was .0916.



House Prices: Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

4,397 teams · Ongoing

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions **Submit Predictions**

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
linear_regression_prediction.csv	a few seconds ago	0 seconds	0 seconds	0.58400

Complete

Jump to your position on the leaderboard

You may select up to 5 submissions to be used to count towards your final leaderboard score. If 5 submissions are not selected, they will be automatically chosen based on your best submission scores on the public leaderboard. In the event that automatic selection is not suitable, manual selection instructions will be provided in the competition rules or by official forum announcement.

Your final score may not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission — your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

```
kaggle competitions submit -c house-prices-advanced-regression-techniques -f submission.csv -m "Message"
```

14 submissions for **Javier Palomares** Sort by Most recent

All Successful Selected

Submission and Description	Public Score	Use for Final Score
linear_regression_prediction.csv a few seconds ago by Javier Palomares prediction using linear regression (no regularization)	0.58400	☐

3.Fit a ridge regression (i.e. using l_2 regularization) using α = 0.1. Make a submission of this prediction. Again, report train error, test error and your score/position on Kaggle LB.

In [54]:
```
alpha = .1
ridge_model = Ridge(alpha=alpha)
y = train.SalePrice
# fit the model then get the y predicted
ridge_model.fit(X_train,y)
y_pred = ridge_model.predict(X_train)
# we took a log of the sale price in the pre processing, so we need to exponen
tiate y_pred and y to get the true sale prices
y_pred_exp = np.expm1(y_pred)
y_exp = np.expm1(y)
rmse_ridge = rms(y_pred_exp,y_exp)
print("using alpha = 0.1, Ridge regression had an rmse of {}".format(rmse_ridg
e))
```

```
using alpha = 0.1, Ridge regression had an rmse of 17965.974938885814
```

The ridge model gives a rmse in the Sale price of approximately 17965.97 when $\alpha = .1$

In [55]:
```
ids = test['Id']
print_predictions("ridge_prediction_alpha1.csv",'Id,SalePrice\n',ids,y_pred_ex
p)
```

House Prices: Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

4,401 teams · Ongoing

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions **Submit Predictions**

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
ridge_prediction_alpha1.csv	just now	0 seconds	0 seconds	0.56958

Complete

Jump to your position on the leaderboard

You may select up to 5 submissions to be used to count towards your final leaderboard score. If 5 submissions are not selected, they will be automatically chosen based on your best submission scores on the public leaderboard. In the event that automatic selection is not suitable, manual selection instructions will be provided in the competition rules or by official forum announcement.

Your final score may not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission — your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

```
kaggle competitions submit -c house-prices-advanced-regression-techniques -f submission.csv
-m "Message"
```

15 submissions for **Javier Palomares** Sort by Most recent

All Successful Selected

Submission and Description	Public Score	Use for Final Score
ridge_prediction_alpha1.csv just now by Javier Palomares ridge regression using alpha=.1	0.56958	☐

This submission had an rmse on the logs of the sale prices of .5695. Better than linear regression

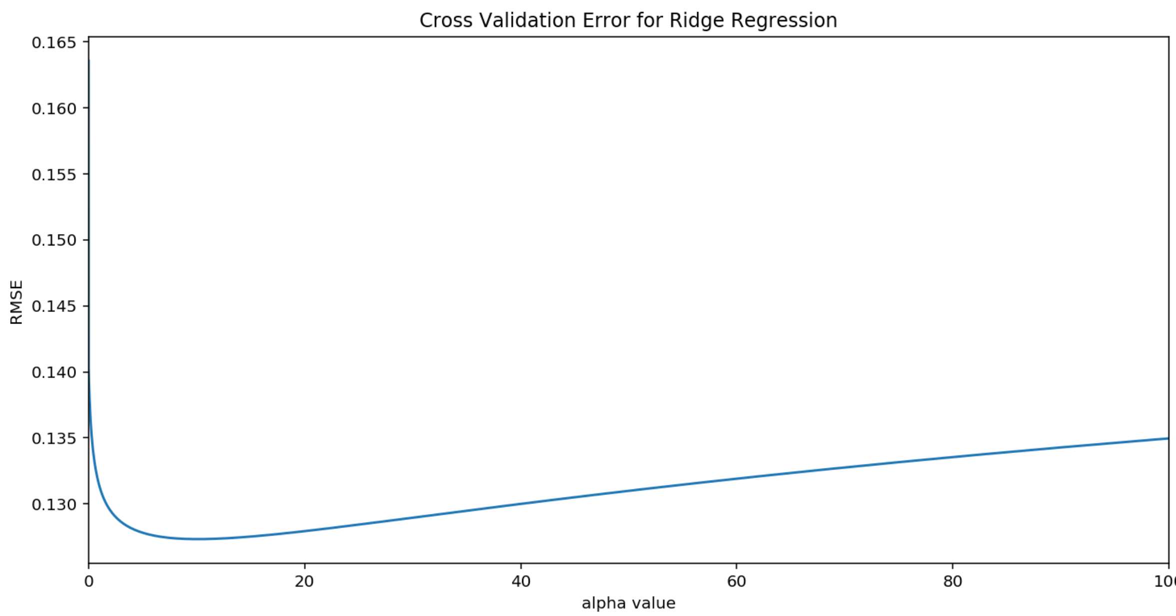
4.Train a ridge regression and a lasso regression model. Optimize the alphas using cross validation. What is the best score you can get from a single ridge regression model and from a single lasso model ? Report also the best hyperparameter α that you find.

Ridge regresion

In [56]:
```
n_alphas = 200
alphas = np.logspace(-5, 3, n_alphas).tolist()
cv_ridge_rmse = [rmse_cv(Ridge(alpha = alpha),X_train,y).mean()
            for alpha in alphas]
```

In [57]:
```
cv_ridge_rmse = pd.Series(cv_ridge_rmse, index = alphas)
cv_ridge_rmse.plot(title = "Cross Validation Error for Ridge Regression")
plt.xlabel("alpha value")
plt.xlim(0,100)
plt.ylabel("RMSE")
```

Out[57]:
```
Text(0,0.5,'RMSE')
```



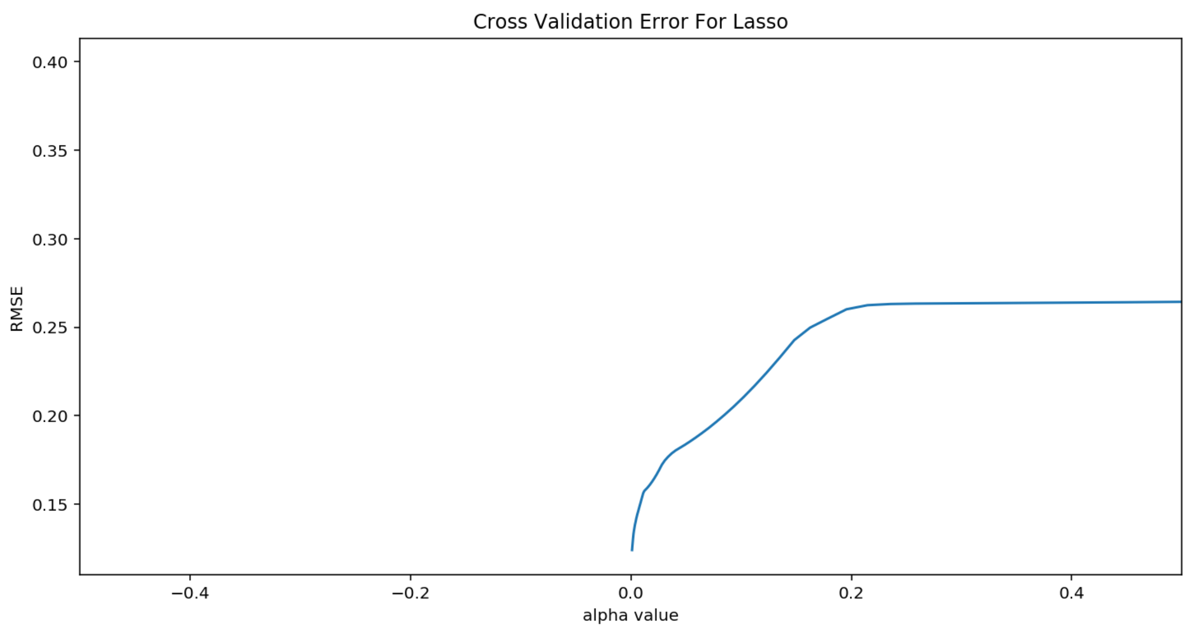
In [58]:
```
min_rmse_ridge = cv_ridge_rmse.min()
best_alpha_ridge = cv_ridge_rmse[cv_ridge_rmse==min_rmse_ridge].index[0]
print("The lowest cross validation RMSE for Ridge Regression is {} when alpha=
{}".format(min_rmse_ridge,best_alpha_ridge))
```

```
The lowest cross validation RMSE for Ridge Regression is 0.12733847319204633
when alpha=9.771241535346501
```

Lasso

In [59]:
```
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, Lasso,LassoCV, La
ssoLarsCV
# the alphas in lasso are inverted from ridge
inv_alphas = np.ones(len(alphas)) /np.array(alphas)
cv_lasso_rmse = [rmse_cv(Lasso(alpha = alpha),X_train,y).mean()
            for alpha in inv_alphas]
cv_lasso_rmse = pd.Series(cv_lasso_rmse, index = inv_alphas)
cv_lasso_rmse.plot(title = "Cross Validation Error For Lasso")
plt.xlabel("alpha value")
plt.ylabel("RMSE")
plt.xlim(-.5,.5)
```

Out[59]:
```
(-0.5, 0.5)
```



In [60]:
```
min_rmse_lasso = cv_lasso_rmse.min()
best_alpha_lasso = cv_lasso_rmse[cv_lasso_rmse==min_rmse_lasso].index[0]
print("The lowest cross validation RMSE for Ridge Regression is {} when alpha=
{}".format(min_rmse_lasso,best_alpha_lasso))
```

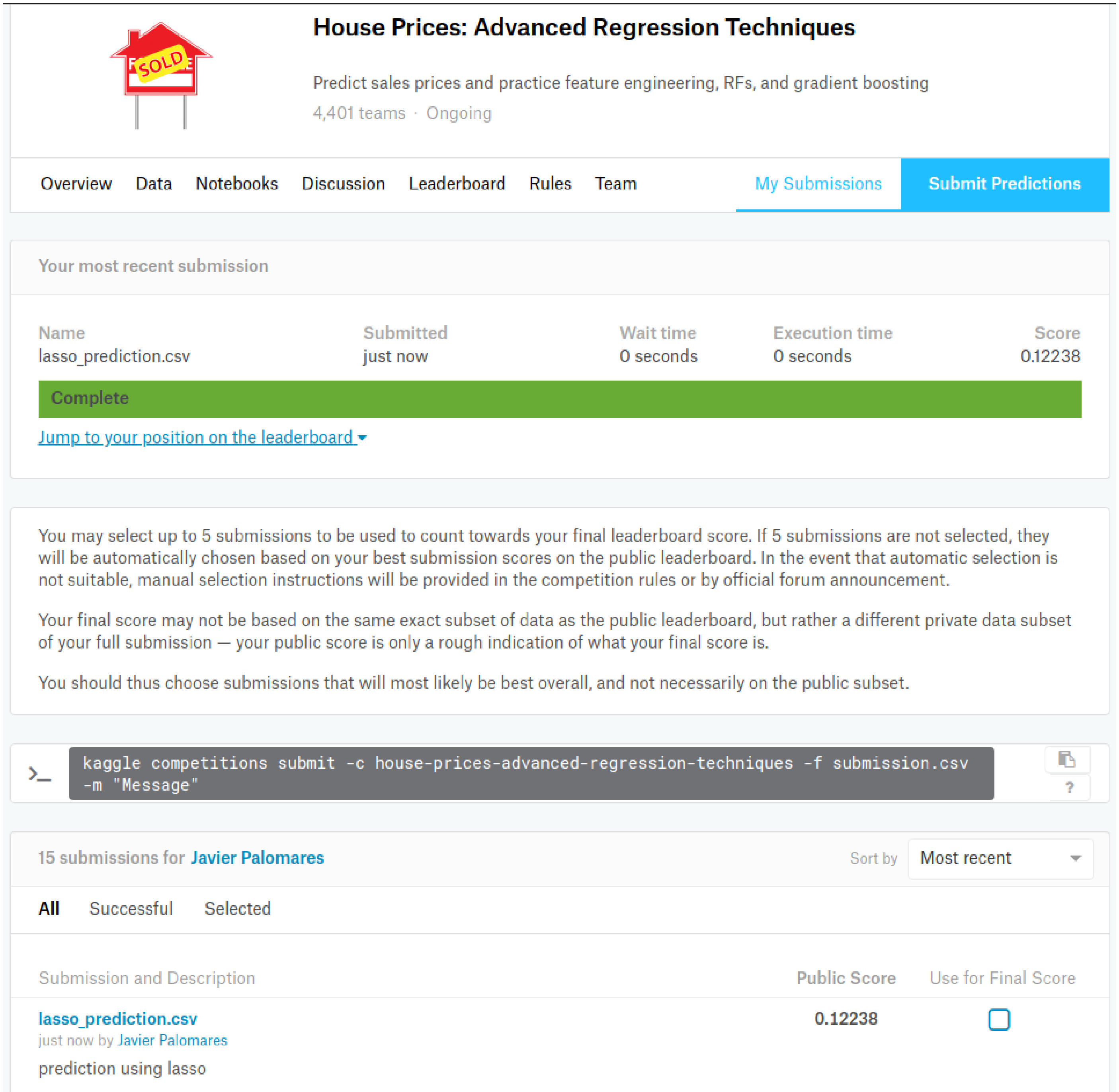
```
The lowest cross validation RMSE for Ridge Regression is 0.1241949894226694 w
hen alpha=0.001
```

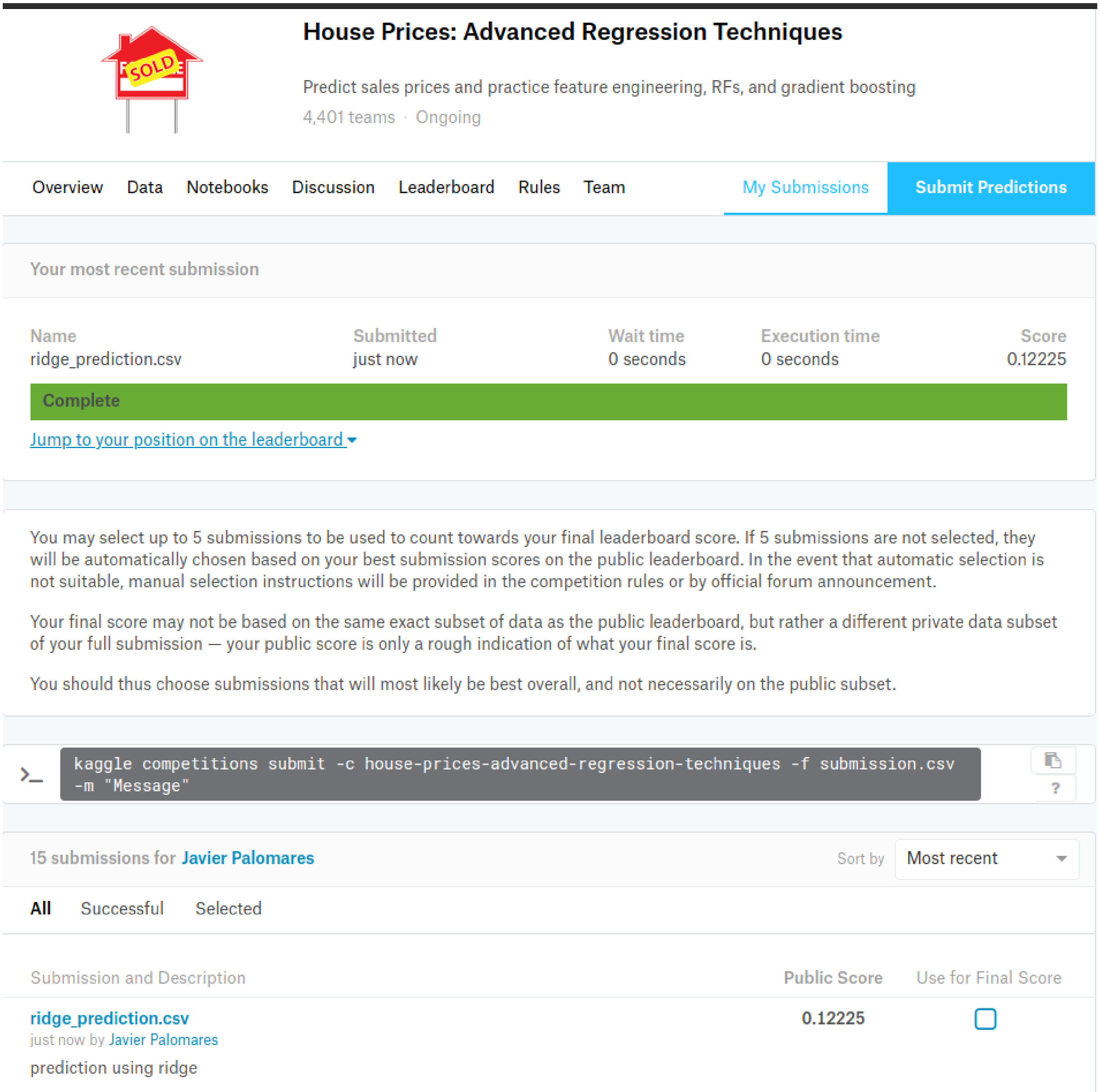
In [61]:
```
# Create predictions with the best alpha scores for lasso and ridge
best_ridge = Ridge(alpha=best_alpha_ridge).fit(X_train,y)
best_lasso = Lasso(alpha=best_alpha_lasso).fit(X_train,y)
```

In [62]:
```
# exponentiate the prediction since we took the log of the sale price in the t
raining data
y_pred_ridge = np.expm1(best_ridge.predict(X_test))
y_pred_lasso = np.expm1(best_lasso.predict(X_test))
ids = test['Id']
```

```
In [31]: print_predictions("ridge_prediction.csv",'Id,SalePrice\n',ids,y_pred_ridge)
         print_predictions("lasso_prediction.csv",'Id,SalePrice\n',ids,y_pred_lasso)
```

My lasso submission scored 0.12238. My ridge regression scored 0.12225.

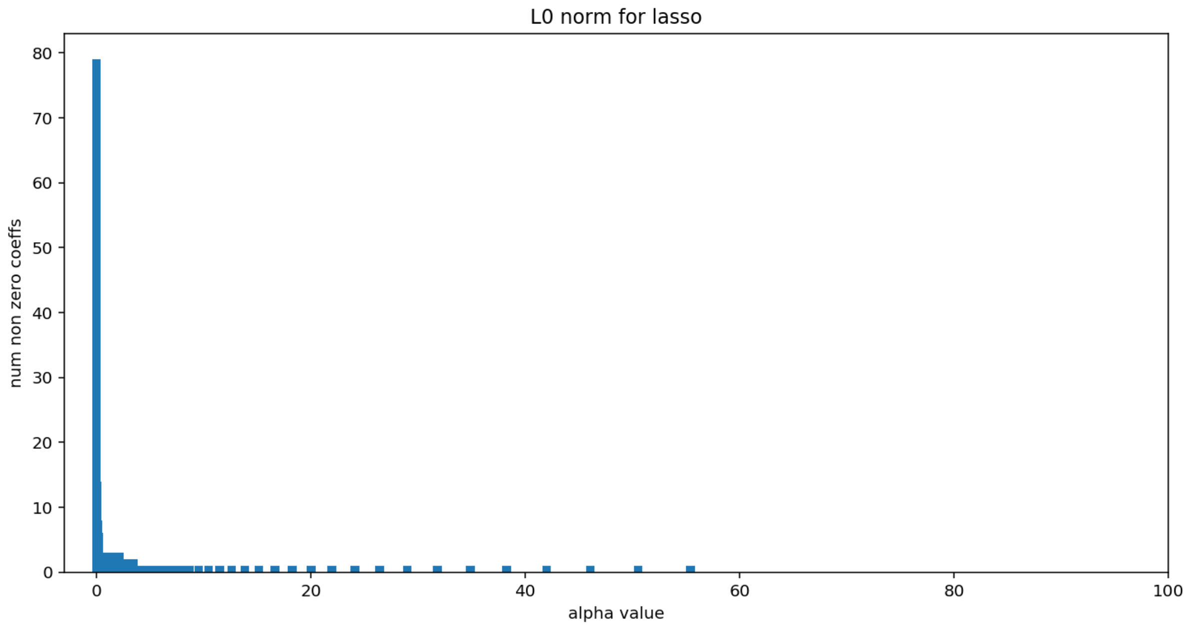




5.Plot the l_0 norm (number of nonzeros) of the coefficients that lasso produces as you vary the regularization hyperparameter alpha.

In [63]:
```
import numpy as np
l0_norm = np.zeros(len(inv_alphas))
for i in range(len(inv_alphas)):
    alpha = inv_alphas[i]
    lasso_model = Lasso(alpha=alpha)
    lasso_model.fit(X_train,y)
    coeff = lasso_model.coef_
    num_nzero_coeff = sum(coeff != 0)
    l0_norm[i] = num_nzero_coeff
```

In [64]:
```
plt.bar(inv_alphas,l0_norm)
plt.xlim(-3,100)
plt.title("L0 norm for lasso")
plt.xlabel('alpha value')
plt.ylabel('num non zero coeffs')
plt.show()
```



6.Add the outputs of your models as features and train a ridge regression on all the features plus the model outputs (This is called Ensembling and Stacking). Be careful not to overfit. What score can you get?

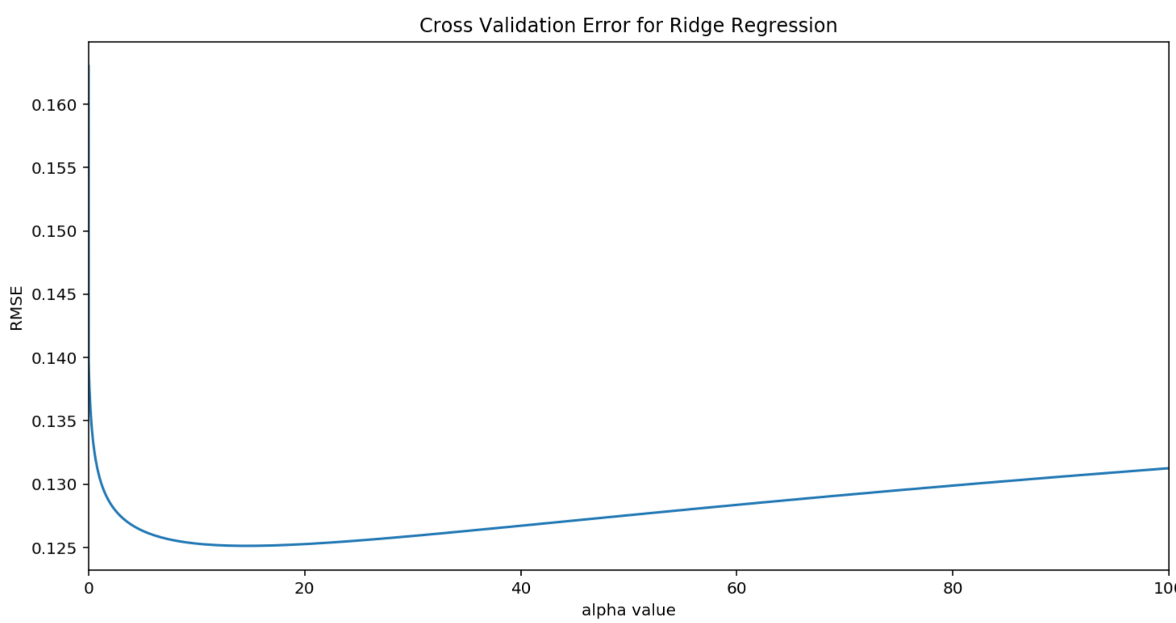
In [66]:
```
ridge_model_output = best_ridge.predict(X_train).tolist()
lasso_model_output = best_lasso.predict(X_train).tolist()
```

In [67]:
```
# add the output of the ridge and lasso models as features of X_train
X_train['ridge_pred'] = ridge_model_output
X_train['lasso_pred'] = lasso_model_output
```

```
C:\Users\javie\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy

C:\Users\javie\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

In [68]:
```
# now train a ridge regression with the new features
n_alphas = 200
alphas = np.logspace(-5, 3, n_alphas).tolist()
# Now retrain a ridge regression.
cv_ridge_stacking = [rmse_cv(Ridge(alpha = alpha),X_train,y).mean()
            for alpha in alphas]
cv_ridge_stacking = pd.Series(cv_ridge_stacking, index = alphas)
cv_ridge_stacking.plot(title = "Cross Validation Error for Ridge Regression")
plt.xlabel("alpha value")
plt.xlim(0,100)
plt.ylabel("RMSE")
```

Out[68]: Text(0,0.5,'RMSE')



In [69]:
```
min_rmse_ridge_stacking = cv_ridge_stacking.min()
best_alpha_ridge_stacking = cv_ridge_stacking[cv_ridge_stacking==min_rmse_ridg
e_stacking].index[0]
print("The lowest cross validation RMSE after stacking for Ridge Regression is
{} when alpha={}".format(min_rmse_ridge_stacking,best_alpha_ridge_stacking))
```

```
The lowest cross validation RMSE after stacking for Ridge Regression is 0.125
13383313691065 when alpha=14.149912974345758
```

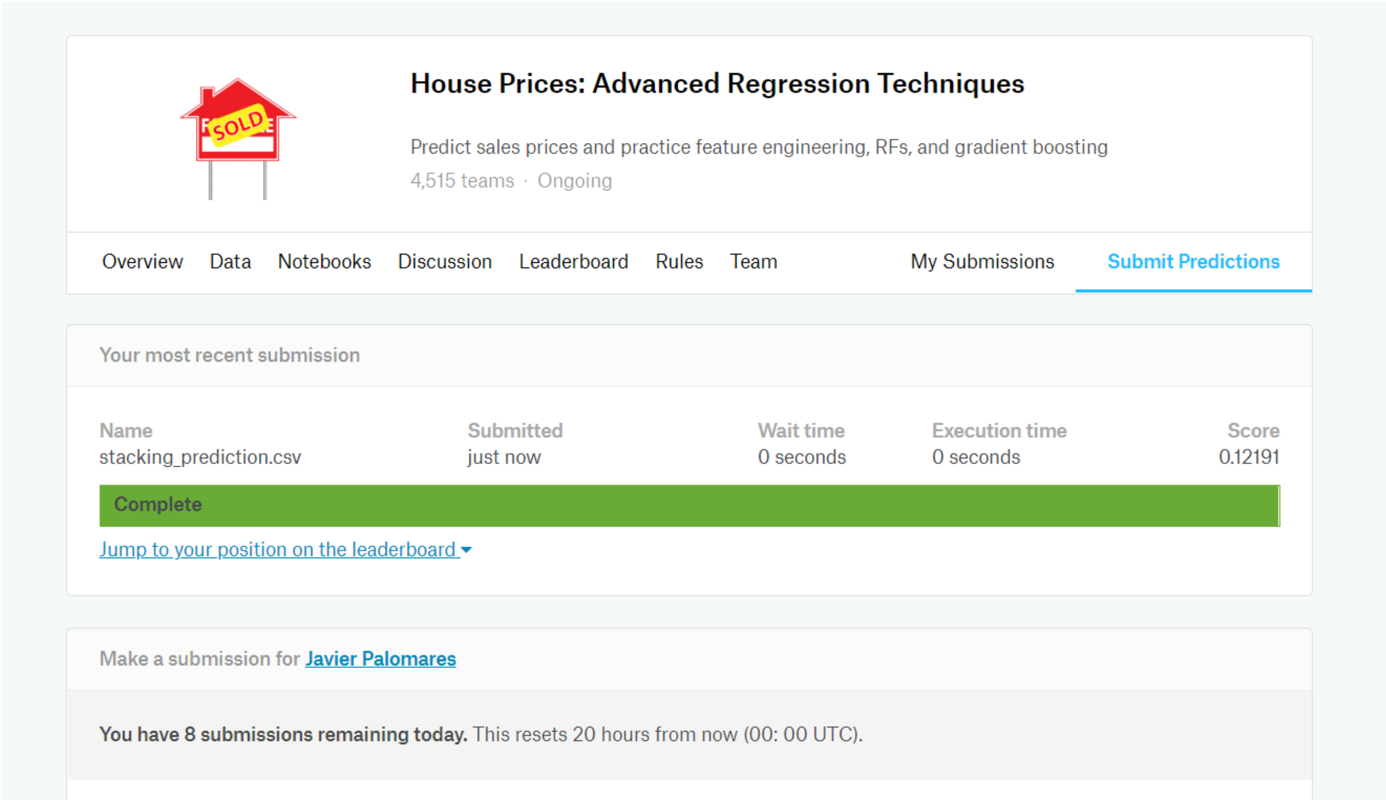
In [70]:
```
best_ridge_stacking = Ridge(alpha=best_alpha_ridge_stacking).fit(X_train,y)
```

In [71]:
```
# Add the predictions from ridge and lasso to the test data as well
y_pred_ridge = best_ridge.predict(X_test).tolist()
y_pred_lasso = best_lasso.predict(X_test).tolist()
X_test['ridge_pred'] = y_pred_ridge
X_test['lasso_pred'] = y_pred_lasso
```

```
C:\Users\javie\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
C:\Users\javie\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  """
```

In [72]:
```
# don't forget to exponentiate the outputs
y_pred_stacking = np.expm1(best_ridge_stacking.predict(X_test))
print_predictions("stacking_prediction.csv",'Id,SalePrice\n',ids,y_pred_stacki
ng)
```

Stacking had a score of 0.12191 which is an improvement!



7.Improve your perfomance by running any model or method you would like to try. Experiment with feature engineering and stacking many models. You are allowed to use any public tool in python. No nonpython tools allowed. Read the Kaggle forums and kernels to get ideas. Include in your report if you find something in the forums you like, or if you made a post or code, especially if other Kagglers used it afterwards.

I will try xgboost.

In [ ]:
```
import xgboost
from xgboost import plot_importance
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score,KFold
from sklearn.model_selection import  train_test_split
train = pd.read_csv("./input/train.csv")
test = pd.read_csv("./input/test.csv")
all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'],
                      test.loc[:,'MSSubClass':'SaleCondition']))
all_data = pd.get_dummies(all_data)
# filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.SalePrice
ids = test['Id']
parameters_for_testing = {
    'colsample_bytree':[0.4,0.6,0.8],
    'min_child_weight':[1.5,6,10],
    'learning_rate':[0.1,0.07,0.01],
    'max_depth':[3,5,7],
    'subsample':[0.6,0.95]
}

xgb_model = xgboost.XGBRegressor(learning_rate =0.1, n_estimators=1000, max_de
pth=5,
                                 min_child_weight=1, gamma=0, subsample=0.8, c
olsample_bytree=0.8, nthread=6, scale_pos_weight=1, seed=27)

gsearch = GridSearchCV(estimator = xgb_model, param_grid = parameters_for_test
ing, n_jobs=6,iid=False, verbose=10,scoring='neg_mean_squared_error')
gsearch.fit(X_train,y)
```

```
Fitting 3 folds for each of 162 candidates, totalling 486 fits

/usr/local/lib/python3.7/site-packages/sklearn/model_selection/_split.py:205
3: FutureWarning: You should specify a value for 'cv' instead of relying on t
he default value. The default value will change from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
[Parallel(n_jobs=6)]: Using backend LokyBackend with 6 concurrent workers.
[Parallel(n_jobs=6)]: Done   1 tasks      | elapsed:   12.3s
[Parallel(n_jobs=6)]: Done   6 tasks      | elapsed:   12.6s
[Parallel(n_jobs=6)]: Done  13 tasks      | elapsed:   32.9s
[Parallel(n_jobs=6)]: Done  20 tasks      | elapsed:   45.8s
[Parallel(n_jobs=6)]: Done  29 tasks      | elapsed:   59.8s
[Parallel(n_jobs=6)]: Done  38 tasks      | elapsed:  1.5min
```

In [ ]:
```
print (gsearch.cv_results_)
print('best params')
print (gsearch.best_params_)
print('best score')
print (gsearch.best_score_)
```

The grid search found the best parameter values:
{'colsample_bytree': 0.4, 'learning_rate': 0.07, 'max_depth': 3, 'min_child_weight': 1.5, 'subsample': 0.6}

In [ ]:
```
xgb_model = xgboost.XGBRegressor(learning_rate =0.07, n_estimators=1000, max_depth=3,
                                 min_child_weight=1.5, gamma=0, subsample=0.6,
colsample_bytree=0.4, nthread=6, scale_pos_weight=1, seed=27,eval_metric='rmse')
xgb_model.fit(X_train,y)
y_pred_xgb = xgb_model.predict(X_test)
print_predictions("xgb_prediction.csv",'Id,SalePrice\n',ids,y_pred_xgb)
```

This xgboost regression had a score of 0.13129, which is actually not an improvement over what I saw with stacking. Note that I did I reloaded the data, so the preprocessing steps were skipped. I'll apply the preprocessing and stack on the lasso and ridge predictions.

In [44]:
```
## Read the data
train = pd.read_csv("./input/train.csv")
test = pd.read_csv("./input/test.csv")
all_data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'],
                      test.loc[:,'MSSubClass':'SaleCondition']))

## Preprocessing
# log transform the sale price
train["SalePrice"] = np.log1p(train["SalePrice"])

# get the numberical features
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index

# compute the skeweness factor of features
# take features with factor greater than .75
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

# take the log of all skewed features
all_data[skewed_feats] = np.log1p(all_data[skewed_feats])
# Convert categorical variable into dummy/indicator variables
all_data = pd.get_dummies(all_data)
# filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())
```

In [45]:
```
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.SalePrice
# stack the lasso and ridge predictions
ridge_model_output = best_ridge.predict(X_train).tolist()
lasso_model_output = best_lasso.predict(X_train).tolist()
X_train['ridge_pred'] = ridge_model_output
X_train['lasso_pred'] = lasso_model_output

# Add the predictions from ridge and lasso to the test data as well
y_pred_ridge = best_ridge.predict(X_test).tolist()
y_pred_lasso = best_lasso.predict(X_test).tolist()
X_test['ridge_pred'] = y_pred_ridge
X_test['lasso_pred'] = y_pred_lasso
```

```
/usr/local/Cellar/ipython/7.7.0/libexec/vendor/lib/python3.7/site-packages/ip
ykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  import sys
/usr/local/Cellar/ipython/7.7.0/libexec/vendor/lib/python3.7/site-packages/ip
ykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy

/usr/local/Cellar/ipython/7.7.0/libexec/vendor/lib/python3.7/site-packages/ip
ykernel_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  del sys.path[0]
/usr/local/Cellar/ipython/7.7.0/libexec/vendor/lib/python3.7/site-packages/ip
ykernel_launcher.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

import xgboost from xgboost import plot_importance from sklearn.model_selection import GridSearchCV from sklearn.model_selection import cross_val_score,KFold from sklearn.model_selection import train_test_split xgb_stack_model = xgboost.XGBRegressor(learning_rate =0.07, n_estimators=1000, max_depth=3, min_child_weight=1.5, gamma=0, subsample=0.6, colsample_bytree=0.4, nthread=6, scale_pos_weight=1, seed=27,eval_metric='rmse') xgb_stack_model.fit(X_train,y)

In [48]:
```
y_pred_xgb_stack = np.expm1(xgb_stack_model.predict(X_test))
```

In [49]:
```
print_predictions("xgb_stacking_prediction.csv",'Id,SalePrice\n',ids,y_pred_xg
b_stack)
```

This prediction had a score of 0.12396.

I want to try one more model: A random forest regressor

In [50]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(X_train,y)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/ensemble/forest.py:246: Future
Warning: The default value of n_estimators will change from 10 in version 0.2
0 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[50]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', 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=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

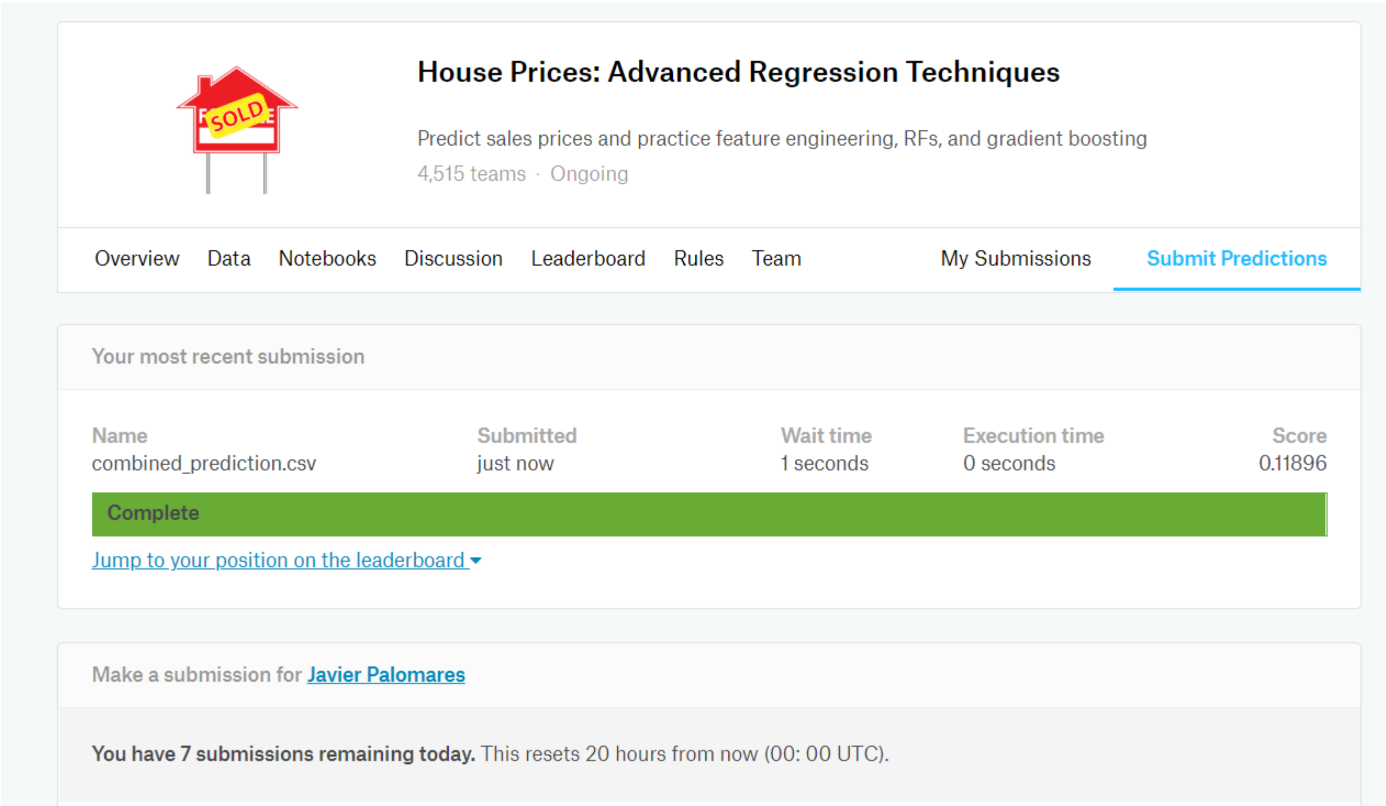
In [51]:
```
y_rf_pred = np.expm1(rf.predict(X_test))
```

In [52]:
```
print_predictions("rf_prediction.csv",'Id,SalePrice\n',ids,y_rf_pred)
```

Random forest had a good score of 0.13225, but not better than the score achieved by stacking. I want to try one more thing: combinining models. I'll combine the stacking model, the random forest, and the xgb regressor.

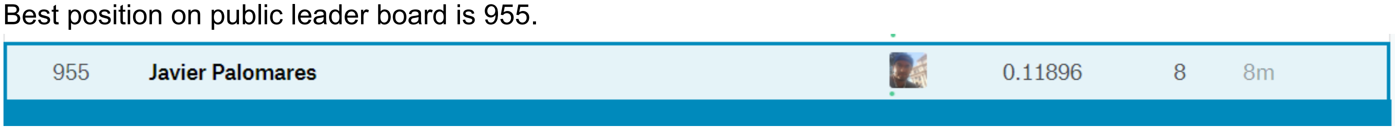
In [ ]:
```
y_pred_combined = .33*y_pred_xgb + .33*y_rf_pred + .34*y_pred_stacking
print_predictions("combined_prediction.csv",'Id,SalePrice\n',ids,y_pred_combin
ed)
```

This prediction has a better score of 0.11896 and advanced me 302 places. This looks like the best I can do for now.



8.Report the best RMSE you got and the position on the private Kaggle leader board, and how you got it

The best RMSE I have so far is .11896. The competition is not closed yet so I can't see the private leaderboard. Best position on public leader board is 955.



I achieved my best score by ensembling 3 models with equal weight: xg_boost model with hyperparameter tuning, random forest, and a ridge model that took in the output of ridge and lasso as stacked features.