Importing Libraries

In [4]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression,Ridge, Lasso
from sklearn.model_selection import cross_val_score
import matplotlib
%pylab inline
pd.options.display.max_columns = 300
```

```
Populating the interactive namespace from numpy and matplotlib
```

Populating the interactive namespace from numpy and matplotlib

In [5]:

```
train = pd.read_csv("C:/Users/ddalv/Documents/Courses/ML & Stats/HW/train.csv")
target = train["SalePrice"]
train = train.drop("SalePrice",1) # remove the target variable
test = pd.read_csv("C:/Users/ddalv/Documents/Courses/ML & Stats/HW/test.csv")
combi = pd.concat((train,test)) # we combine both the dataframes which do not have targ
et variable
```

In [6]:

```
print(shape(train))
print(shape(test))
print(shape(combi))
```

```
(1460, 80)
(1459, 80)
(2919, 80)
```

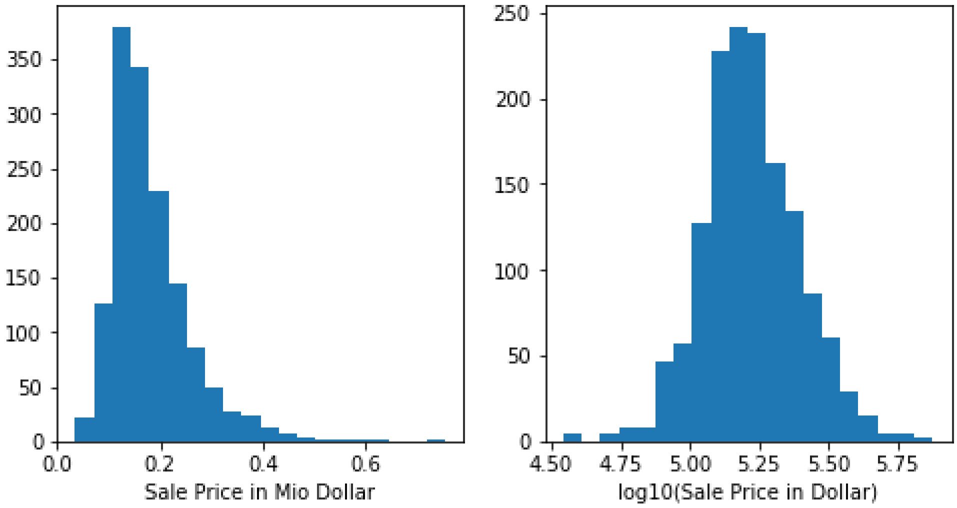
We look at the distribution of sales price in Linear and Log space

In [7]:

```
figure(figsize(8,4))
subplot(1,2,1)
hist(target*1e-6,20);
xlabel("Sale Price in Mio Dollar")
subplot(1,2,2)
hist(log10(target),20);
xlabel("log10(Sale Price in Dollar)")
```

Out[7]:

```
Text(0.5,0,'log10(Sale Price in Dollar)')
```



In [8]:

```
target = log10(target)
```

In [9]:

```
combi.head(10)
```

Out[9]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandC
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	6	50	RL	85.0	14115	Pave	NaN	IR1	Lvl
6	7	20	RL	75.0	10084	Pave	NaN	Reg	Lvl
7	8	60	RL	NaN	10382	Pave	NaN	IR1	Lvl
8	9	50	RM	51.0	6120	Pave	NaN	Reg	Lvl
9	10	190	RL	50.0	7420	Pave	NaN	Reg	Lvl

We observe that there are a lot of categorical features and NaN values.

In [10]:

```
# create new features from categorical data:
combi = pd.get_dummies(combi)
# fill missing entries with the mean of the column:
combi = combi.fillna(combi.mean())
# create new train and test arrays:
train = combi[:train.shape[0]]
test = combi[train.shape[0]:]
```

In [11]:

```
combi.head(10)
```

Out[11]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearR
0	1	60	65.000000	8450	7	5	2003	2003
1	2	20	80.000000	9600	6	8	1976	1976
2	3	60	68.000000	11250	7	5	2001	2002
3	4	70	60.000000	9550	7	5	1915	1970
4	5	60	84.000000	14260	8	5	2000	2000
5	6	50	85.000000	14115	5	5	1993	1995
6	7	20	75.000000	10084	8	5	2004	2005
7	8	60	69.305795	10382	7	6	1973	1973
8	9	50	51.000000	6120	7	5	1931	1950
9	10	190	50.000000	7420	5	6	1939	1950

Linear Regression Model

In [12]:

```
model = LinearRegression()
score = mean(sqrt(-cross_val_score(model, train, target,scoring="neg_mean_squared_erro
r", cv = 5)))
print("linear regression score: ", score)
```

```
linear regression score:  0.0682956085468
```

Lasso and Ridge regresion model

In [15]:

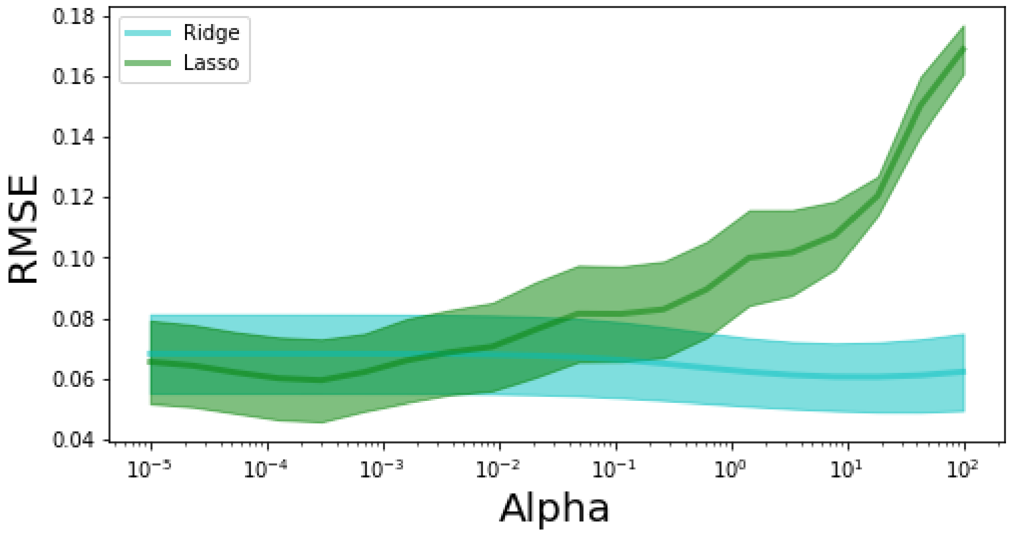
```
cv = 5 #number of folds in cross-validation
alphas = np.logspace(-5,2,20)
scores = np.zeros((len(alphas),cv))
scores_mu = np.zeros(len(alphas))
scores_sigma = np.zeros(len(alphas))
for i in range(0,len(alphas)):
    model = Ridge(alpha=alphas[i])
    scores[i,:] = sqrt(-cross_val_score(model, train, target,scoring="neg_mean_squared_
error", cv = cv))
    scores_mu[i] = mean(scores[i,:])
    scores_sigma[i] = std(scores[i,:])

#for i in range(0,cv):
figure(figsize(8,4))
# Now, we plot the Ridge model and print the best score in Ridge RegressionModel
#plot(alphas,scores[:,i], 'b--', alpha=0.5)
plot(alphas,scores_mu,'c-',lw=3, alpha=0.5, label = "Ridge")
fill_between(alphas,np.array(scores_mu)-np.array(scores_sigma),np.array(scores_mu)+np.a
rray(scores_sigma),color="c",alpha=0.5)
print("best score in Ridge: ",min(scores_mu))
for i in range(0,len(alphas)):
    model = Lasso(alpha=alphas[i])
    scores[i,:] = sqrt(-cross_val_score(model, train, target,scoring="neg_mean_squared_
error", cv = cv))
    scores_mu[i] = mean(scores[i,:])
    scores_sigma[i] = std(scores[i,:])

# Now, we plot the Lasso model and print the best score in Lasso RegressionModel
plot(alphas,scores_mu,'g-',lw=3, alpha=0.5, label="Lasso")
fill_between(alphas,np.array(scores_mu)-np.array(scores_sigma),np.array(scores_mu)+np.a
rray(scores_sigma),color="g",alpha=0.5)
xscale("log")
plt.xlabel("Alpha", size=20)
plt.ylabel("RMSE", size=20)
legend(loc=2)
print("best score in Lasso: ",min(scores_mu))
```

```
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 2.5508727885589203e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 2.922534561235701e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 2.6864527616164448e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 2.324552779902582e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 2.8311873077391874e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 5.957758740464754e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 6.764998579736344e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 6.354543681900229e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 5.4504692032253564e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)
C:\Users\ddalv\Anaconda3\lib\site-packages\scipy\linalg\basic.py:223: Runt
imeWarning: scipy.linalg.solve
Ill-conditioned matrix detected. Result is not guaranteed to be accurate.
Reciprocal condition number: 6.573651146590225e-17
  ' condition number: {}'.format(rcond), RuntimeWarning)

best score in Ridge:  0.060652516145
best score in Lasso:  0.0595313605364
```



Conclusion : Lasso Performs Better than Ridge Regression