In [329]:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
import pandas as pd
import seaborn as sns
import matplotlib
```

In [330]:

```
train = pd.read_csv("house-prices-advanced-regression-techniques/train.csv")
test = pd.read_csv("house-prices-advanced-regression-techniques/test.csv")
```

In [331]:

```
train.head()
```

Out[331]:

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

5 rows × 81 columns

In [332]:

```
data = pd.concat((train.loc[:,'MSSubClass':'SaleCondition'], test.loc[:,'MSSub
Class':'SaleCondition']))
```

In [333]:

```
train["SalePrice"] = np.log1p(train["SalePrice"])

#log transform skewed numeric features:
indeces = data.dtypes[data.dtypes != "object"].index

skewed_feats = train[indeces].apply(lambda x: skew(x.dropna())) #compute skewn
ess
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index

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

In [334]:

```
data = pd.get_dummies(data)
```

In [335]:

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

In [336]:

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

In [337]:

```
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import cross_val_score
```

In [338]:

```
model_ridge = Ridge(alpha=.1)
model_ridge.fit(X_train, Y_train)
```

Out[338]:

```
Ridge(alpha=0.1)
```

In [339]:

```
preds = model_ridge.predict(X_test)
preds = np.expm1(preds)
```

In [340]:

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

In [341]:

```
#After submitting to Kaggle, we get a RMSE of .1377
```

In [342]:

```
def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, Y_train, scoring="neg_mean_squared_error", cv = 5))
    return(rmse)
```

In [343]:

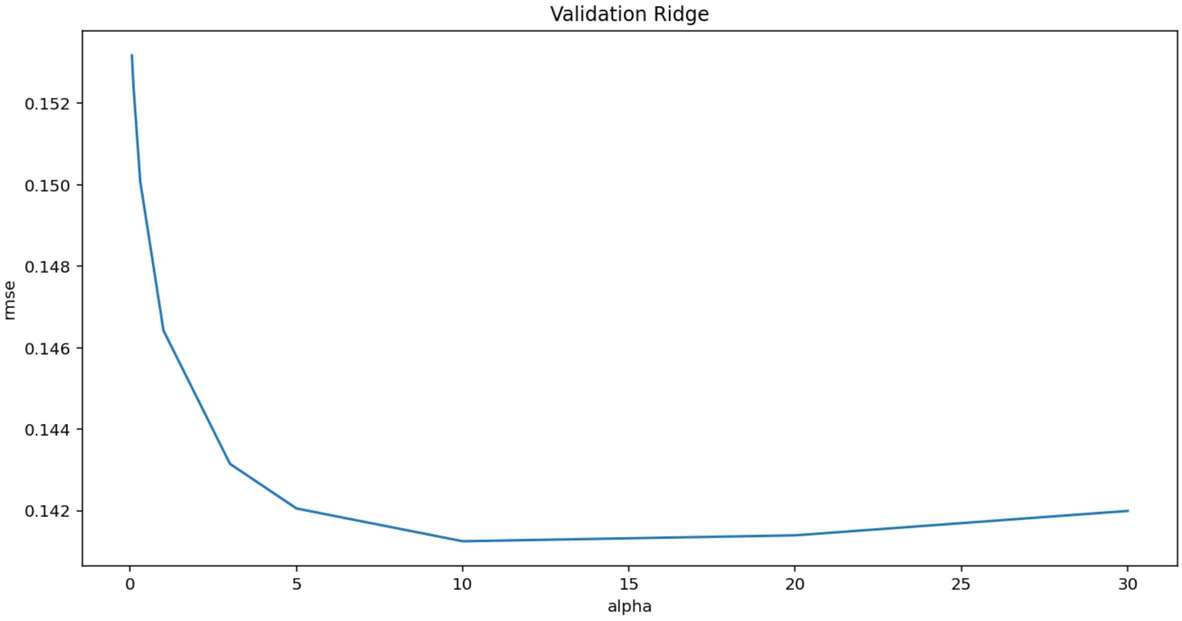
```
alphas = [0.05, 0.1, 0.3, 1, 3, 5, 10, 20, 30]
cv_ridge = [rmse_cv(Ridge(alpha = alpha)).mean()
            for alpha in alphas]
```

In [344]:

```
cv_ridge = pd.Series(cv_ridge, index = alphas)
cv_ridge.plot(title = "Validation Ridge")
plt.xlabel("alpha")
plt.ylabel("rmse")
```

Out[344]:

```
Text(0, 0.5, 'rmse')
```



In [345]:

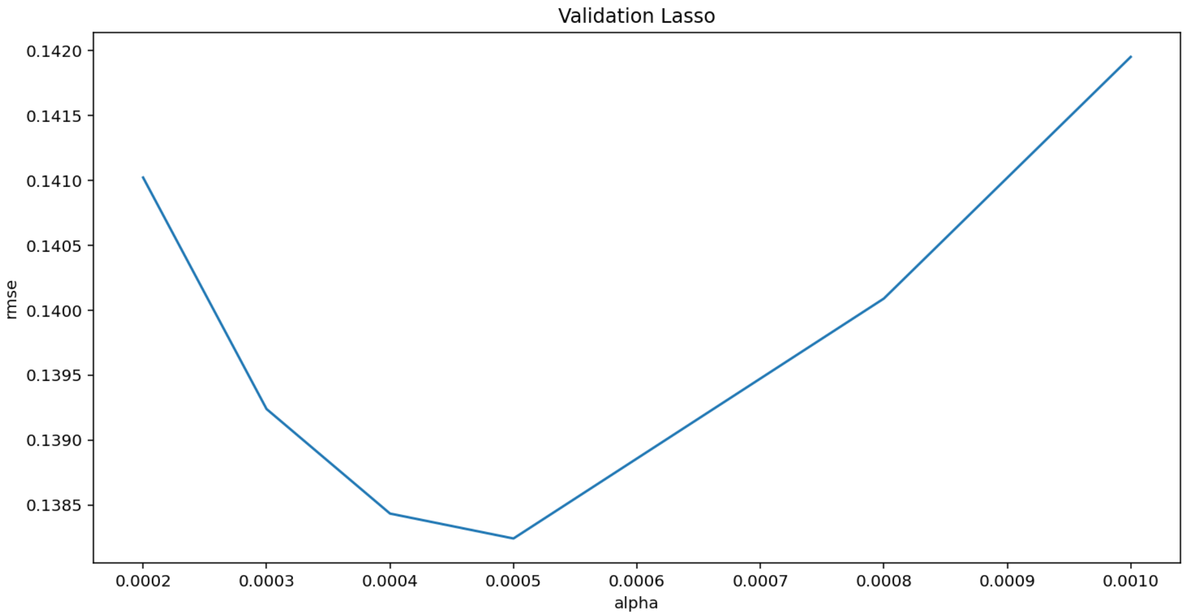
```
alphasl = [.001, .0008, .0005, .0004, .0003, .0002]
cv_lasso = [rmse_cv(Lasso(alpha = alpha)).mean()
            for alpha in alphasl]
```

In [346]:

```
cv_lasso = pd.Series(cv_lasso, index = alphasl)
cv_lasso.plot(title = "Validation Lasso")
plt.xlabel("alpha")
plt.ylabel("rmse")
```

Out[346]:

```
Text(0, 0.5, 'rmse')
```



In [347]:

```
# For a single LASSO Model, we can get to a RMSE of ~.138
# For a single Ridge Model, we can get to a RMSE of ~.141
```

In [348]:

```
models_lasso = [Lasso(alpha = alpha).fit(X_train, Y_train) for alpha in alphas
l]
```

In [349]:

```
coefs = [pd.Series(models_lasso[i].coef_, index = X_train.columns) for i in ra
nge(0, len(alphasl))]
```

In [350]:

```
l0 = np.zeros_like(alphasl)
for i in range(0, len(alphasl)):
    l0[i] = sum(coefs[i] != 0)
```

In [351]:

```
print(l0)
```

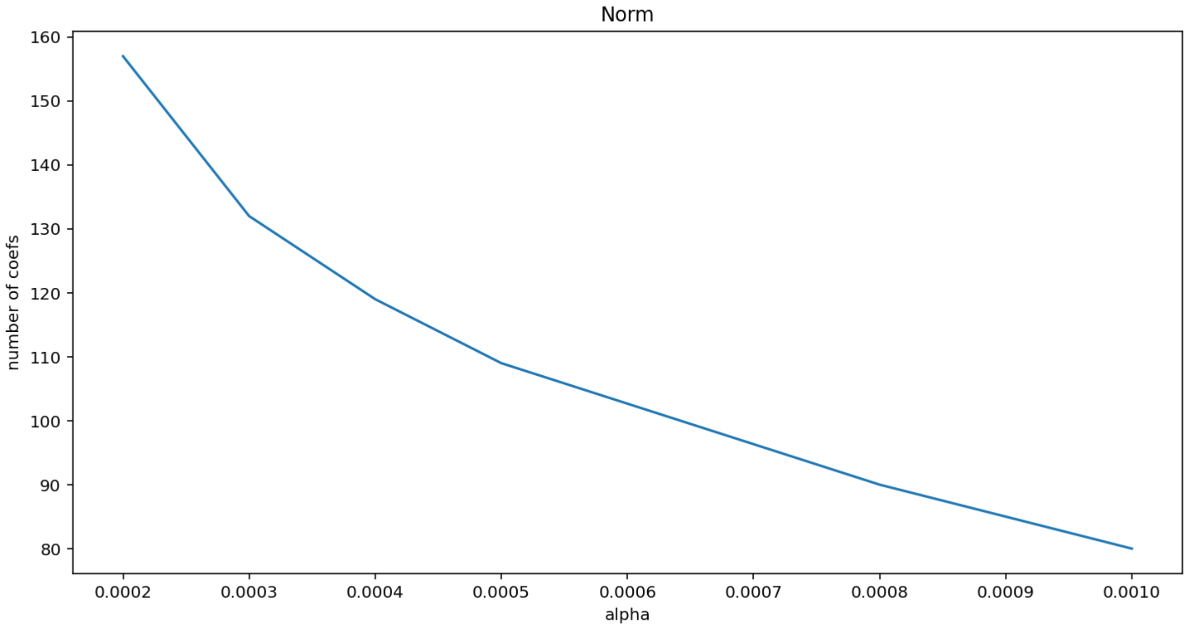
```
[ 80.  90. 109. 119. 132. 157.]
```

In [352]:

```
l0 = pd.Series(l0, index = alphasl)
l0.plot(title = "Norm")
plt.xlabel("alpha")
plt.ylabel("number of coefs")
```

Out[352]:

```
Text(0, 0.5, 'number of coefs')
```



In [353]:

```
predictions_lasso = [models_lasso[i].predict(X_train) for i in range(0, len(al
phasl))]
```

In [358]:

```
X_train.loc[:]["lasso1"] = predictions_lasso[0]
X_train.loc[:]["lasso2"] = predictions_lasso[1]
X_train.loc[:]["lasso3"] = predictions_lasso[2]
X_train.loc[:]["lasso4"] = predictions_lasso[3]
X_train.loc[:]["lasso5"] = predictions_lasso[4]
X_train.loc[:]["lasso6"] = predictions_lasso[5]
```

In [363]:

```
model_ridge_es = Ridge(alpha=10)
model_ridge_es.fit(X_train, Y_train)
```

Out[363]:

```
Ridge(alpha=10)
```

In [364]:

```
def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, Y_train, scoring="neg_mean_
squared_error", cv = 5))
    return(rmse)
```

In [366]:

```
cv_ridge_es = rmse_cv(Ridge(alpha = 10)).mean()
print(cv_ridge_es)
```

```
0.13445149400996287
```

In [ ]:

```
# We can get down to a RMSE score of .134 which is better than both the LASSO
 and the Ridge Models
```