# #loading CSV data with PD

import pandas as pd

url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/bikeshare.csv'

bikes = pd.read\_csv(url, index\_col='datetime', parse\_dates=True)

# Correlation and head map

import seaborn as sns

bikes.corr()

sns.heatmap(bikes.corr())

# Linear regression with one variable

#create X and y

feature\_cols = ['temp']

X=bikes[feature\_cols]

y =bikes.total

#instantiate and fit

from sklearn.linear\_model import LinearRegression

linreg = LinearRegression()

linreg.fit(X, y)

#print the coefficients

print linreg.intercept\_

print linreg.coef\_

# pair the feature names with the coefficients

zip(feature\_cols, linreg.coef\_)

# #linear regression normal with multiple variables

# create X and y

feature\_cols = ['temp', 'season', 'weather', 'humidity']

X = bikes[feature\_cols]

y = bikes.total

# instantiate and fit

linreg = LinearRegression()

linreg.fit(X, y)

# print the coefficients

print linreg.intercept\_

print linreg.coef\_

# pair the feature names with the coefficients

zip(feature\_cols, linreg.coef\_)

# #splitting training set.

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=123)

# # Regularization with RidgeCV (different reg parms🡺alphas)

Regularized cost function as mentioned below.

\underset{w}{min\,} {{|| X w - y||_2}^2 + \alpha {||w||_2}^2}

>>> from sklearn import linear\_model

>>> clf = linear\_model.RidgeCV(alphas=[0.1, 1.0, 10.0])

>>> clf.fit(X\_train, y\_train)

RidgeCV(alphas=[0.1, 1.0, 10.0], cv=None, fit\_intercept=True, scoring=None,

normalize=False)

pred=clf.predict(X\_test)

>>> clf.alpha\_

0.1

# #regularization with LassoCV(different reg parms🡺alphas)

Regularized cost function as mentioned below.

\underset{w}{min\,} { \frac{1}{2n_{samples}} ||X w - y||_2 ^ 2 + \alpha ||w||_1}

>>> from sklearn import linear\_model

>>> clf = linear\_model.LassoCV(alphas=[0.1, 1.0, 10.0])

>>> clf.fit(X\_train, y\_train)

Lasso(alpha=0.1, copy\_X=True, fit\_intercept=True, max\_iter=1000,

normalize=False, positive=False, precompute=False, random\_state=None,

selection='cyclic', tol=0.0001, warm\_start=False)

>>>pred= clf.predict(X\_test)

array([ 0.8])

# # calculate metrics!

from sklearn import metrics

import numpy as np

print 'MAE:', metrics.mean\_absolute\_error(true, pred)

print 'MSE:', metrics.mean\_squared\_error(true, pred)

print 'RMSE:', np.sqrt(metrics.mean\_squared\_error(true, pred))

**RMSE:155.649459131---With Ridge**

**RMSE:155.643749947---With Losso**

One variable temp

**RMSE:166.175955908---With normal**

**RMSE:166.175913119---With RidgeCV**

**RMSE:166.175581741---With LASSO**

# #Normalize the features

LinearRegression(fit\_intercept=True, normalize=False, copy\_X=True, n\_jobs=1)

LassoCV(*eps=0.001*, *n\_alphas=100*, *alphas=None*, *fit\_intercept=True*,*normalize=False*, *precompute='auto'*, *max\_iter=1000*, *tol=0.0001*, *copy\_X=True*, *cv=None*, *verbose=False*, *n\_jobs=1*,*positive=False*, *random\_state=None*, *selection='cyclic'*)

*RidgeCV(*alphas=(0.1*,*1.0*,*10.0)*,*fit\_intercept=True*,*normalize=False*,*scoring=None*,*cv=None*,*gcv\_mode=None*,*store\_cv\_values=False*)*

***normalize****: boolean, optional, default False*

*If True, the regressors X will be normalized before regression.*

**fit\_intercept** : boolean

Whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations (e.g. data is expected to be already centered)

**alphas** : numpy array, optional

List of alphas where to compute the models. If None alphas are set automatically

# #difference between Ridge and LASSO

Ridge and Lasso regression uses two different penalty functions. Ridge uses l2 where as lasso go with l1. In ridge regression, the penalty is the sum of the squares of the coefficients and for the Lasso, it's the sum of the absolute values of the coefficients.