**R – code to clean the data**

library(RCurl)

url = getURL("http://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv")

forest\_fires <- read.csv(text = url)

head(forest\_fires)

# get a model matrix

ff\_clean = model.matrix(~., data=forest\_fires)

ff\_clean = as.data.frame(ff\_clean)

# write the csv file to my coursera folder

library(readr)

write\_csv(x=ff\_clean, path="E:/forestfire/datafile/forest\_fires.csv")

# get column print out for python (line 19)

library(pystr)

cols = colnames(ff\_clean)

dput(pystr\_upper(cols))

**Python code and Lasso implementation**

#importing Series and DataFrame from pandas  
import pandas as pd  
import numpy as np  
import matplotlib.pylab as plt  
from sklearn.cross\_validation import train\_test\_split  
from sklearn.linear\_model import LassoLarsCV  
  
#Loading dataset  
data = pd.read\_csv(r"E:\forestfire\datafile\forest\_fires.csv")  
  
#upper-case all DataFrame column names  
data.columns = map(str.upper, data.columns)  
  
#removing missing values  
data\_clean = data.dropna()  
  
#select predictor variables and target variable as separate data sets  
predvar= data\_clean[["X", "Y", "MONTHAUG", "MONTHDEC", "MONTHFEB",  
"MONTHJAN", "MONTHJUL", "MONTHJUN", "MONTHMAR", "MONTHMAY", "MONTHNOV",  
"MONTHOCT", "MONTHSEP", "DAYMON", "DAYSAT", "DAYSUN", "DAYTHU",  
"DAYTUE", "DAYWED", "FFMC", "DMC", "DC", "ISI", "TEMP", "RH",  
"WIND", "RAIN"]]  
  
target = data\_clean.AREA  
  
# standardize predictors to have mean=0 and sd=1  
predictors=predvar.copy()  
from sklearn import preprocessing  
predictors['X']=preprocessing.scale(predictors['X'].astype('float64'))  
predictors['Y']=preprocessing.scale(predictors['Y'].astype('float64'))  
predictors['MONTHAUG']=preprocessing.scale(predictors['MONTHAUG'].astype('float64'))  
predictors['MONTHDEC']=preprocessing.scale(predictors['MONTHDEC'].astype('float64'))  
predictors['MONTHFEB']=preprocessing.scale(predictors['MONTHFEB'].astype('float64'))  
predictors['MONTHJAN']=preprocessing.scale(predictors['MONTHJAN'].astype('float64'))  
predictors['MONTHJUL']=preprocessing.scale(predictors['MONTHJUL'].astype('float64'))  
predictors['MONTHJUN']=preprocessing.scale(predictors['MONTHJUN'].astype('float64'))  
predictors['MONTHMAR']=preprocessing.scale(predictors['MONTHMAR'].astype('float64'))  
predictors['MONTHMAY']=preprocessing.scale(predictors['MONTHMAY'].astype('float64'))  
predictors['MONTHNOV']=preprocessing.scale(predictors['MONTHNOV'].astype('float64'))  
predictors['MONTHOCT']=preprocessing.scale(predictors['MONTHOCT'].astype('float64'))  
predictors['MONTHSEP']=preprocessing.scale(predictors['MONTHSEP'].astype('float64'))  
predictors['DAYMON']=preprocessing.scale(predictors['DAYMON'].astype('float64'))  
predictors['DAYSAT']=preprocessing.scale(predictors['DAYSAT'].astype('float64'))  
predictors['DAYSUN']=preprocessing.scale(predictors['DAYSUN'].astype('float64'))  
predictors['DAYTHU']=preprocessing.scale(predictors['DAYTHU'].astype('float64'))  
predictors['DAYTUE']=preprocessing.scale(predictors['DAYTUE'].astype('float64'))  
predictors['DAYWED']=preprocessing.scale(predictors['DAYWED'].astype('float64'))  
predictors['FFMC']=preprocessing.scale(predictors['FFMC'].astype('float64'))  
predictors['DMC']=preprocessing.scale(predictors['DMC'].astype('float64'))  
predictors['DC']=preprocessing.scale(predictors['DC'].astype('float64'))  
predictors['ISI']=preprocessing.scale(predictors['ISI'].astype('float64'))  
predictors['TEMP']=preprocessing.scale(predictors['TEMP'].astype('float64'))  
predictors['RH']=preprocessing.scale(predictors['RH'].astype('float64'))  
predictors['WIND']=preprocessing.scale(predictors['WIND'].astype('float64'))  
predictors['RAIN']=preprocessing.scale(predictors['RAIN'].astype('float64'))  
  
# spliting data into train and test sets  
pred\_train, pred\_test, tar\_train, tar\_test = train\_test\_split(predictors, target,  
 test\_size=.3, random\_state=123)  
  
#lasso regression implementation  
model=LassoLarsCV(cv=10, precompute=False).fit(pred\_train,tar\_train)  
  
# print variable names and regression coefficients  
dict(zip(predictors.columns, model.coef\_))  
  
# plot coefficient progression  
m\_log\_alphas = -np.log10(model.alphas\_)  
ax = plt.gca()  
plt.plot(m\_log\_alphas, model.coef\_path\_.T)  
plt.axvline(-np.log10(model.alpha\_), linestyle='--', color='k',  
 label='alpha CV')  
plt.ylabel('Regression Coefficients')  
plt.xlabel('negative log(alpha)')  
plt.title('Regression Coefficients Progression for Lasso Paths')  
  
# plot mean square error for each fold  
m\_log\_alphascv = -np.log10(model.cv\_alphas\_)  
plt.figure()  
plt.plot(m\_log\_alphascv, model.cv\_mse\_path\_, ':')  
plt.plot(m\_log\_alphascv, model.cv\_mse\_path\_.mean(axis=-1), 'k',  
 label='Average across the folds', linewidth=2)  
plt.axvline(-np.log10(model.alpha\_), linestyle='--', color='k',  
 label='alpha CV')  
plt.legend()  
plt.xlabel('negative log(alpha)')  
plt.ylabel('Mean square error')  
plt.title('Mean square error on each fold')  
  
  
# MSE from training and test data  
from sklearn.metrics import mean\_squared\_error  
train\_error = mean\_squared\_error(tar\_train, model.predict(pred\_train))  
test\_error = mean\_squared\_error(tar\_test, model.predict(pred\_test))  
print ('training data MSE')  
print(train\_error)  
print ('test data MSE')  
print(test\_error)  
  
# R-square from training and test data  
rsquared\_train=model.score(pred\_train,tar\_train)  
rsquared\_test=model.score(pred\_test,tar\_test)  
print ('training data R-square')  
print(rsquared\_train)  
print ('test data R-square')  
print(rsquared\_test)

**Output Parameters :**

**training data MSE**

**5383.211576492841**

**test data MSE**

**704.656477226023**

**training data R-square**

**0.027785253229491436**

**test data R-square**

**-0.26831916949360446**