```
# -*- coding: utf-8 -*-
"""
Created on Mon March 6 14:22:00 2017

@author: EAmankwah
"""

# -*- coding: utf-8 -*-

#from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import scipy.stats
import seaborn as sns
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV

# bug fix for display formats to avoid run time errors
pd.set_option('display.float_format', lambda x:'%.2f'%x)

#Load the dataset

ad = pd.read_csv('worldbank.csv')

#Set PANDAS to show all columns in DataFrame
pd.set_option('display.max_columns', None)

#Set PANDAS to show all rows in DataFrame
pd.set_option('display.max_rows', None)

# convert variables to numeric format using convert_objects function
```

```
ad['x12_2013'] = pd.to_numeric(ad['x12_2013'], errors='coerce')
ad['x18_2013'] = pd.to_numeric(ad['x18_2013'], errors='coerce')
ad['x21_2013'] = pd.to_numeric(ad['x21_2013'], errors='coerce')
ad['x121_2013'] = pd.to_numeric(ad['x121_2013'], errors='coerce')
ad['x129_2013'] = pd.to_numeric(ad['x129_2013'], errors='coerce')
ad['x140_2013'] = pd.to_numeric(ad['x140_2013'], errors='coerce')
ad['x154_2013'] = pd.to_numeric(ad['x154_2013'], errors='coerce')
ad['x161_2013'] = pd.to_numeric(ad['x161_2013'], errors='coerce')
ad['x222_2013'] = pd.to_numeric(ad['x222_2013'], errors='coerce')
ad['x283_2013'] = pd.to_numeric(ad['x283_2013'], errors='coerce')

#upper-case all DataFrame column names
ad.columns = map(str.upper, ad.columns)

#clean data
data_clean = ad[['X12_2013','X18_2013','X21_2013','X121_2013','X129_2013','X140_2013',
'X154_2013','X161_2013','X222_2013','X283_2013']].dropna()

#print(data_clean.dtypes)
print(data_clean.describe())
print('\n')

# categories one predictor variable, polityscore into binary variable
def GDPGRP (row):
   if row['X140_2013'] <=3.16:
      return 0
   if row['X140_2013']>3.16:
      return 1

#checking that recoding is dane
data_clean['GDPGRP'] = data_clean.apply(lambda row:GDPGRP(row), axis=1)
chkld = data_clean['GDPGRP'].value_counts(sort=False, dropna=False)
print(chkld)
```

```
"""
Modeling and Prediction
"""
#select predictor variables and target variable as separate data sets

predvar = data_clean[['X18_2013','X21_2013','X121_2013','X129_2013','GDPGRP',
'X154_2013','X161_2013','X222_2013','X283_2013']]

target = data_clean.X12_2013

#split data for scatterplots
train,test=train_test_split(data_clean, test_size=.4, random_state=123)

# better variable names and labels for plots
train['Carbon Dioxide Damage']=train['X12_2013']
train['Energy Depletion']=train['X18_2013']
train['Natural Res Depletion']=train['X21_2013']
train['Export of GS']=train['X121_2013']
train['Food Production']=train['X129_2013']
train['Import of GS']=train['X154_2013']
train['Industry Value Added']=train['X161_2013']
train['Working Population']=train['X222_2013']
train['Urbanization']=train['X283_2013']

scat1 = sns.regplot(x="X18_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Energy Depletion')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Energy Depletion')
plt.show()
print ('Association Between Carbon Dioxide Damage and Energy Depletion')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X18_2013']))
print('\n')
```

```
scat2 = sns.regplot(x="X21_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Natural Resource Depletion')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Natural Res Depletion'
plt.show()
print ('Association Between Carbon Dioxide Damage and Natural Resource  Depletion')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X21_2013']))
print('\n')

scat3 = sns.regplot(x="X121_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Export of Goods and Services')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Export of Goods and Se
plt.show()
print ('Association Between Carbon Dioxide Damage and Export of Goods and Services')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X121_2013']))
print('\n')

scat4 = sns.regplot(x="X129_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Food Production')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Food Production')
plt.show()
print ('Association Between Carbon Dioxide Damage and Food Production')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X129_2013']))
print('\n')

scat5 = sns.regplot(x="X154_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Import of Goods and Services')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Import of Goods and Se
plt.show()
```

```
print ('Association Between Carbon Dioxide Damage and Import of Goods and Services')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X154_2013']))
print('\n')

scat6 = sns.regplot(x="X161_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Industry Value Added')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Industry Value Added')
plt.show()
print ('Association Between Carbon Dioxide Damage and Industry Value Added')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X161_2013']))
print('\n')

scat7 = sns.regplot(x="X222_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Working Population')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Working Population')
plt.show()
print ('Association Between Carbon Dioxide Damage and Working Population')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X222_2013']))
print('\n')

scat8 = sns.regplot(x="X283_2013", y="X12_2013", fit_reg=False, data=ad)
plt.xlabel('Urbanization')
plt.ylabel('Carbon Dioxide Damage')
plt.title('Scatterplot for the Association Between Carbon Dioxide Damage and Urbanization')
plt.show()
print ('Association Between Carbon Dioxide Damage and Urbanization')
print (scipy.stats.pearsonr(data_clean['X12_2013'], data_clean['X283_2013']))
print('\n')

# standardize predictors to have mean=0 and sd=1
predictors=predvar.copy().dropna()
```

```
from sklearn import preprocessing

#looping through
for col in ['X18_2013','X21_2013','X121_2013','X129_2013','GDPGRP','X154_2013',
'X161_2013','X222_2013','X283_2013']:
    preprocessing.scale(predictors[col].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target,
                                                              test_size=.4, random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
print(dict(zip(predictors.columns, model.coef_)))
print('\n')

# 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('-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('-log(alpha)')

plt.ylabel('Mean squared error')
plt.title('Mean squared 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('\n')
print ('test data MSE')
print(test_error)
print('\n')

# 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('\n')
print ('test data R-square')
print(rsquared_test)
```