```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict
from sklearn.multioutput import MultiOutputRegressor
from sklearn.linear_model import (LinearRegression, Ridge, RidgeCV,
    MultiTaskLassoCV, SGDClassifier, SGDRegressor, TheilSenRegressor,
    RANSACRegressor, HuberRegressor)
from sklearn.neural_network import (MLPClassifier, MLPRegressor)
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

from collections import namedtuple

countries = {
    'Australia' : [-35.3, 149.12],
    'Brazil' : [-15.75, -47.95],
    'Tanzania' : [-6.17, 35.74],
    'Indonesia' : [-6.17, 106.82],
    'Kenya' : [-1.26, 36.8],
    'Ethiopia' : [9.03, 38.74],
    'Cambodia' : [11.55, 104.91],
    'Mali' : [12.65, -8],
    'Thailand' : [13.75, 100.48],
    'Senegal' : [14.66, -17.41],
    'Cape Verde' : [14.91, -23.51],
    'Belize' : [17.25, -88.76],
    'Jamaica' : [17.98, -76.8],
    'Myanmar' : [19.75, 96.1],
    'Taiwan' : [23.76, 121],
    'India' : [28.61, 77.2],
    'Egypt' : [30.03, 31.21],
    'Pakistan' : [33.66, 73.16],
    'Morocco' : [34.03, -6.85],
    'Iran' : [35.68, 51.41],
    'Japan' : [35.7, 139.71],
    'Algeria' : [36.7, 3.21],
    'Greece' : [38, 23.71],
    'Turkey' : [39.91, 32.83],
    'China' : [39.91, 116.38],
    'Uzbekistan' : [41.26, 69.21],
    'Albania' : [41.33, 19.8],
    'Georgia' : [41.71, 44.78],
    'Italy' : [41.9, 12.48],
    'Kyrgyzstan' : [42.86, 74.6],
    'Romania' : [44.41, 26.1],
    'UK' : [52.5, -0.12],
    'Lithuania' : [54.68, 25.31]
}

def genCSV4(name, index, latitude, longitude):
    '''
    Not a general function, just tacks together the specific case for 
this Kaggle
    '''
    result = np.zeros((index.shape[0], 2))
    index = index.A1
    result[:,0] = np.array(latitude[:,0])
    result[:,1] = np.array(longitude[:,0])

    columns = {'lat', 'long'}
    df = pd.DataFrame(result, columns=columns, index=index)
    df.index.name = 'index'
    name = 'outputs/' + name + '.csv'
    df.to_csv(name)

def genCSV(name, index, prediction):
    '''
    Not a general function, just tacks together the specific case for 
this Kaggle
    '''
    index = index.A1

    columns = {'lat', 'long'}
    df = pd.DataFrame(prediction, columns=columns, index=index)
    df.index.name = 'index'
    name = 'outputs/' + name + '.csv'
    df.to_csv(name)

def genCSV_predtest(name, prediction, truth):
    predcopy = prediction.copy()
    truthcopy = truth.copy()
    content = np.hstack((prediction, truth))
    name = 'testOutputs/' + name + '.csv'
    np.savetxt(name, content, delimiter=",")

class Models:

    def __init__(self, X_train, X_test, Y_train, Y_test, X_final, 
index_final):
        self.X_train = X_train
        self.X_test = X_test
        self.Y_train = Y_train
        self.Y_test = Y_test
        self.X_final = X_final
        self.index_final = index_final

    def ridge(self, name):
        '''
        Ridge
        '''
        sciRidge = Ridge(
            alpha= 300, #tested alpha values, alpha=302 has lowest
score
            fit_intercept=True,
            normalize=False,
            max_iter=None )
        sciRidge.fit(self.X_train, self.Y_train[:,:2])
        predict_test = sciRidge.predict(self.X_test)
        MSE = mean_squared_error(predict_test, self.Y_test[:,:2])
        s = "Sci Ridge              (MSE: %f)" % (MSE)
        print s
        predict_final = sciRidge.predict(self.X_final)
        genCSV( (name + '_MSE' + str(MSE)), self.index_final,
predict_final )

    def ridgeCV(self, name):
        '''
        RidgeCV
        '''
        sciRidgeCV = RidgeCV(
            alphas=(0.001, 0.01, 0.1, 1, 2, 5, 20, 40, 60, 80, 100,
120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320, 340), #tested
alpha values, 321 works best
            fit_intercept=True,
            cv = 11,
            normalize=False )
        sciRidgeCV.fit(self.X_train, self.Y_train[:,:2])
        predict_test = sciRidgeCV.predict(self.X_test)
        MSE = mean_squared_error(predict_test,self.Y_test[:,:2])
        s = "Sci RidgeCV            (MSE: %f)" % (MSE)
        print s
        predict_final = sciRidgeCV.predict(self.X_final)
        genCSV( name + '_MSE' + str(MSE), self.index_final,
predict_final )

    def lassoCV(self, name):
        '''
        Lasso Regression
        '''
        sciLasso = MultiTaskLassoCV(
            fit_intercept=True,
            normalize=False,
            cv=12,
            tol = 0.001 )
        sciLasso.fit(self.X_train, self.Y_train)
        predict_test = sciLasso.predict(self.X_test)
        MSE = mean_squared_error(predict_test,self.Y_test)
        s = "Sci LassoCV            (MSE: %f)" % (MSE)
        print s
        print  sciLasso.score(self.X_test, self.Y_test)
        predict_final = sciLasso.predict(self.X_final)
        genCSV( name + '_MSE' + str(MSE), self.index_final,
predict_final )

    def SGDClassifier(self):
        '''
        SGD Classifier
        '''
        sciSGD = SGDClassifier()

        sciSGD.fit(self.X_train, self.Y_train[:,2])

        predict_test = sciSGD.predict(self.X_test)#.astype(np.float)

    def MLPClassifier(self, name):
        '''
        MLP Classifier
        '''
        sciMLP = MLPClassifier(solver='lbfgs')

        sciMLP.fit(self.X_train, self.Y_train[:,2].A1)

        predict_class = sciMLP.predict(self.X_test)

        predict_test = np.zeros((len(predict_class), 2))

        for i in range(len(predict_class)):
            predict_test[i] = countries[predict_class[i]]

        MSE = mean_squared_error(predict_test, self.Y_test[:,:2])

        print MSE

    def MLPRegressor(self, name):
        sciMLP = MultiOutputRegressor(
            MLPRegressor(hidden_layer_sizes=(66,),
            activation='logistic', solver='adam', max_iter=200,
batch_size=50)
            )

        sciMLP.fit(self.X_train, self.Y_train[:,:2])

        predict_test = sciMLP.predict(self.X_test)

        MSE = mean_squared_error(predict_test, self.Y_test[:,:2])

        print MSE
    def ForestRegressor(self, name):
        sciForest = MultiOutputRegressor(
            RandomForestRegressor(n_estimators=33)
            )

        sciForest.fit(self.X_train, self.Y_train[:,:2])

        predict_test = sciForest.predict(self.X_test)

        MSE = mean_squared_error(predict_test, self.Y_test[:,:2])

        print MSE


def main():
    trainPredictors = pd.read_csv('InputData/trainPredictors.csv')
    trainTargets = pd.read_csv('InputData/trainTargets.csv')
    testPredictors = pd.read_csv('InputData/testPredictors.csv')

    pdIndex = trainPredictors.iloc[:,0]
    pdX = trainPredictors.iloc[:,1:]
    pdY = trainTargets.iloc[:,1:]
    mean = pdY.mean(axis=0)
    std = pdY.std(axis=0)
    pdX.insert(0, "Design", 1.0)

    pdTestIndex = testPredictors.iloc[:,0]
    pdTestX = testPredictors.iloc[:,1:]
    pdTestX.insert(0, "Design", 1.0)

    X_orig = np.matrix(pdX.values)
    Y = np.matrix(pdY.values)
    index = np.matrix(pdIndex.values).transpose()
    #print np.unique(np.array(Y[:,0]))
    #print np.unique(np.array(Y[:,1]))

    X_final_orig = np.matrix(pdTestX.values)
    index_final = np.matrix(pdTestIndex.values).transpose()

    '''
    Checking for bad parameters
    '''
    # F=open('blah.txt', 'w')#('parameterCheck(rm5rm7rm30)(0.4).txt',
'a')#

    X = X_orig #np.delete(X_orig, [5,7], axis=1)
    X_final = X_final_orig #np.delete(X_final_orig, [5,7], axis=1)

    for i in range(145,150): #140-145
```

```
        print i
        '''
        Training and Testing Data
        '''
        X_train, X_test, Y_train, Y_test = train_test_split( X, Y,
test_size=0.2, random_state=i)

        '''
        Testing Models
        '''
        models = Models(X_train, X_test, Y_train, Y_test, X_final,
index_final)
        #filename = 'Ridge_test0.3_rand' + str(i)
        #models.ridge(filename)
        #filename = 'RidgeCV_quad_test0.3_rand' + str(i)
        #models.ridgeCV(filename)
        filename = 'LassoCV_test0.2_rand' + str(i)
        models.lassoCV(filename)
        #filename = 'MLP_test0.2_rand' + str(i)
        #models.ForestRegressor(filename)

    '''
    Plotting
    '''
    #for i in range(X.shape[1]):
    #    plt.scatter(np.array(Y[:,0]),np.array(Y[:,1]), alpha=0.1)
    #plt.show()



main()
```