Data_parser.py

```
import re
import xlwt
from tempfile import TemporaryFile

# Forwards Dataset
forward = "Dataset/forward_json.txt"
# Discard Dataset
discard = "Dataset/discard_json.txt"
# Test Dataset
test_data = "Dataset/test_json.txt"
# Debug Variable
debug = False

# Dataset Excel Name
'''data_sheet = xlwt.Workbook()
sheet1 = data_sheet.add_sheet("test")'''

# Print to Excel
'''def print_to_excel(info_list, sheet_name):
    for i,e in enumerate(info_list):
        sheet1.write(i,0,e)

    data_sheet.save(sheet_name)'''

# Function to grab data from json
def get_data(file_name):

    data = open(file_name).read()

    # Souce IP Information
    src_ip = re.findall('"src":\s"(\d+.\d+.\d+.\d+)"', data)
    src_port = re.findall('"src_port":\s"(\d+)"', data)

    # Destination IP Information
    dest_ip = re.findall('"dest":\s"(\d+.\d+.\d+.\d+)"', data)
    dest_port = re.findall('"dest_port":\s"(\d+)"', data)

    # Geolocation Information
    city = re.findall('"city":\s"(.*)",\s"host"', data)
    subdivision = re.findall('"subdivision":\s"(.*)",\s"name"', data)
    lat = re.findall('"lat":\s"(-?\d+.\d+)",\s"country"', data)
    country = re.findall('"country":\s"(.*)",\s"postal"', data)
    postal = re.findall('"postal":\s"(.*)",\s"ASN"', data)
    long = re.findall('"long":\s"(-?\d+.\d+)"}', data)

    # ISP Information
    host = re.findall('"host":\s"(.*)",\s"subdivision"', data)
    host_name = re.findall('"name":\s"(.*)",\s"ip"', data)
    isp_ip = re.findall('"ip":\s"(\d+.\d+.\d+.\d+)",\s"lat"', data)
    asn = re.findall('"ASN":\s"(\d+)",\s"long"', data)

    # Debug Messages
    if debug:
        print("src_ip: " + str(len(src_ip)))
        print("src_port: " + str(len(src_port)))
        print("dest: " + str(len(dest_ip)))
        print("dest_port: " + str(len(dest_port)))
        print("city: " + str(len(city)))
        print("subdivision: " + str(len(subdivision)))
        print("lat: " + str(len(lat)))
        print("long: " + str(len(long)))
```

```
        print("country: " + str(len(country)))
        print("postal: " + str(len(postal)))
        print("host: " + str(len(host)))
        print("host_name: " + str(len(host_name)))
        print("isp_ip: " + str(len(isp_ip)))
        print("asn: " + str(len(asn)))

    return src_ip, src_port, dest_ip, dest_port, city, subdivision, lat, country,
postal, long, host, host_name, isp_ip, asn

# Gets all the data and assigns it to the appropriate variable for printing later on
src_ip, src_port, dest_ip, dest_port, city, subdivision, lat, country, postal, long,
host, host_name, isp_ip, asn = get_data(discard)

#data_types = [src_ip, src_port, dest_ip, dest_port, city, subdivision, lat, country,
postal, long, host, host_name, isp_ip, asn]

def get_unique(list):
    ulist = set(list)

    '''for value in ulist:
        print(str(value))'''
    new_a = []

    for ip in ulist:
        new_a.append(ip)


    return new_a

#get_unique(src_port)
unique_list = get_unique(src_port)

for i in range(len(src_port)):
    for j in range(len(unique_list)):
        if src_port[i] == unique_list[j]:
            src_port[i] = j*10

for stuff in src_port:
    print(stuff)
#--------------------------------#
# Print Variables
src_ip_print = False
src_port_print = False
dest_ip_print = False
dest_port_print = False
city_print = False
subdivision_print = False
lat_print = False
country_print = False
postal_print = False
long_print = False
host_print = False
host_name_print = False
isp_ip_print = False
asn_print = False

if src_ip_print:
    for ip in src_ip:
        print(ip)
if src_port_print:
    for port in src_port:
        print(port)
```

```
if dest_ip_print:
    for ip in dest_ip:
        print(ip)
if dest_port_print:
    for port in dest_port:
        print(port)
if city_print:
    for city_name in city:
        print(city_name)
if subdivision_print:
    for subdivision_name in subdivision:
        print(subdivision_name)
if lat_print:
    for lat_num in lat:
        print(lat_num)
if country_print:
    for country_name in country:
        print(country_name)
if postal_print:
    for postal_num in postal:
        print(postal_num)
if long_print:
    for long_num in long:
        print(long_num)
if host_print:
    with open("temp.txt", "w") as file:
        for host_url in host:
            file.writelines(host_url+"\n")
if host_name_print:
    with open("temp.txt", "w") as file:
        for name in host_name:
            file.writelines(name + "\n")
if isp_ip_print:
    for ip in isp_ip:
        print(ip)
if asn_print:
    for num in asn:
        print(num)
#--------------------------------#
# Print All Records
'''for i in range(len(data_types)):
    for records in data_types[i]:
        print(records)'''
```

Nearest_Neighbors.py

```
"""
================================
Nearest Neighbors Classification
================================
Modified in class by Dr. Rivas
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors
from numpy import genfromtxt
from sklearn.model_selection import KFold
import time


# read digits data & split it into X (training input) and y (target output)
X, y, ytrue = genfromtxt('Dataset/features.csv', delimiter=' ')

X = X.reshape(len(X),1)
y = y.reshape(len(y),1)
ytrue = ytrue.reshape(len(ytrue),1)

plt.plot(X, y, '.')
plt.plot(X, ytrue, 'rx')
plt.show()

bestk=[]
kc=0
for n_neighbors in range(1,101,2):
  kf = KFold(n_splits=10)
  #n_neighbors = 85
  kscore=[]
  k=0
  for train, test in kf.split(X):
    #print("%s %s" % (train, test))
    X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]

    #time.sleep(100)

    # we create an instance of Neighbors Classifier and fit the data.
    clf = neighbors.KNeighborsRegressor(n_neighbors, weights='distance')
    clf.fit(X_train, y_train)

    kscore.append(clf.score(X_test,y_test))
    #print kscore[k]
    k=k+1

  print (n_neighbors)
  bestk.append(sum(kscore)/len(kscore))
  print(bestk[kc])
  kc+=1

# to do here: given this array of E_outs in CV, find the max, its
# corresponding index, and its corresponding value of n_neighbors
print(bestk)
```