A3

November 21, 2020

1 Data Integration

[1]:
```
import pandas as pd
from bs4 import BeautifulSoup
try:
    import pandas_read_xml as pdx
    import tabula
except:
    !pip install pandas_read_xml xlrd tabula-py
    import pandas_read_xml as pdx
import tabula
from functools import reduce
import numpy as np
from math import radians, cos, sin, asin, sqrt
```

[2]:
```
def distance(p1, p2):
    lat1, lon1 = p1
    lat2, lon2 = p2
    lon1 = radians(lon1)
    lon2 = radians(lon2)
    lat1 = radians(lat1)
    lat2 = radians(lat2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2

    c = 2 * asin(sqrt(a))

    r = 6378
    return(c * r)

def between_times(time):
    time = [int(i) for i in time.split(':')]
    time_ = 3600 * time[0] + 60 * time[1] + time[2]
    if time_ > 3600 * 7 and time_ < 3600 * 9:
        return True
```

```
    return False
```

[3]:
```
hospitals = open('29893909/hospitals.html')
hospitals = hospitals.read()
hospitals = pd.read_html(hospitals)[0].drop(['Unnamed: 0'], axis=1)

h_id = list(hospitals.id)
lat = [round(i, 6) for i in list(hospitals.lat)]
lng = [round(i, 6) for i in list(hospitals.lng)]
h_coordinates = [np.array((la, ln)) for la, ln in zip(lat, lng)]
```

[4]:
```
supermarkets = pd.read_excel("29893909/supermarkets.xlsx").drop(['Unnamed: 0'],␣
↪axis=1)
sm_id = list(supermarkets.id)
lat = [round(i, 6) for i in list(supermarkets.lat)]
lng = [round(i, 6) for i in list(supermarkets.lng)]
sm_coordinates = [np.array((la, ln)) for la, ln in zip(lat, lng)]
```

[5]:
```
real_state = open('29893909/real_state.xml').read()[2:-1]
soup = BeautifulSoup(real_state)
tags = [item.name for item in soup.root.childGenerator()]
data = {}
for i, child in enumerate(soup.body.root.children):
    data[tags[i]] = []
    for grandchild in child.children:
        data[tags[i]].append(grandchild.text)
real_state_xml = pd.DataFrame(data, columns=tags)
real_state_json = pd.read_json('29893909/real_state.json')
real_state = real_state_json.append(real_state_xml)
```

[6]:
```
shoppingcenters = tabula.read_pdf('29893909/shopingcenters.pdf', pages='all')
shoppingcenters = reduce(lambda a, b: a.append(b), shoppingcenters)
```

[7]:
```
sc_id = list(shoppingcenters.sc_id)
lat = [round(i, 6) for i in list(shoppingcenters.lat)]
lng = [round(i, 6) for i in list(shoppingcenters.lng)]
sc_coordinates = [np.array((la, ln)) for la, ln in zip(lat, lng)]
```

[8]:
```
stops = pd.read_csv("GTFS - Melbourne Train Information/stops.txt")
ts_id = list(stops.stop_id)
lat = [round(i, 6) for i in list(stops.stop_lat)]
lng = [round(i, 6) for i in list(stops.stop_lon)]
ts_coordinates = [np.array((la, ln)) for la, ln in zip(lat, lng)]
```

[9]:
```
stop_times = pd.read_csv("GTFS - Melbourne Train Information/stop_times.txt")
calendar = pd.read_csv("GTFS - Melbourne Train Information/calendar.txt")
trips = pd.read_csv("GTFS - Melbourne Train Information/trips.txt")
```

```
routes = pd.read_csv("GTFS - Melbourne Train Information/routes.txt")
```

[13]:
```
def in_weekdays(service_id):
    temp = 0
    for i in ["monday", "tuesday", "wednesday", "thursday", "friday"]:
        temp += calendar[calendar.service_id==service_id][i].values[0]
    if temp > 0:
        return True
    return False
```

[14]:
```
def closest(location, id, coords):
    distances = [distance(location, coord) for coord in coords]
    return id[distances.index(min(distances))], min(distances)

def coord(df, index):
    return np.array((
        df.loc[index, 'lat'],
        df.loc[index, 'lng']
    ))
```

[15]:
```
lrs = len(real_state)
real_state['suburb'] = ["not available"]*lrs
real_state['Shopping_center_id'] = ["not available"]*lrs
real_state['Distance_to_sc'] = [0]*lrs
real_state['Train_station_id'] = ["not available"]*lrs
real_state['Distance_to_train_station'] = [0]*lrs
real_state['travel_min_to_CBD'] = [0]*lrs
real_state['Transfer_flag'] = [-1]*lrs
real_state['Hospital_id'] = ["not available"]*lrs
real_state['Distance_to_hospital'] = [0]*lrs
real_state['Supermarket_id'] = ["not available"]*lrs
real_state['Distance_to_supermarket'] = [0]*lrs
```

[21]:
```
# handling cases where the hour in time > 24
for index in stop_times[stop_times.arrival_time.str.match("24:\d{1,2}:
 ↪\d{1,2}")].index:
    time = stop_times.loc[index, 'arrival_time'].split(":")
    time = ":".join(["00", time[1], time[2]])
    stop_times.loc[index, 'arrival_time'] = time


for index in stop_times[stop_times.departure_time.str.match("24:\d{1,2}:
 ↪\d{1,2}")].index:
    time = stop_times.loc[index, 'departure_time'].split(":")
    time = ":".join(["00", time[1], time[2]])
    stop_times.loc[index, 'departure_time'] = time
```

```
for index in stop_times[stop_times.arrival_time.str.match("25:\d{1,2}:
↪\d{1,2}")].index:
    time = stop_times.loc[index, 'arrival_time'].split(":")
    time = ":".join(["01", time[1], time[2]])
    stop_times.loc[index, 'arrival_time'] = time

for index in stop_times[stop_times.departure_time.str.match("25:\d{1,2}:
↪\d{1,2}")].index:
    time = stop_times.loc[index, 'departure_time'].split(":")
    time = ":".join(["01", time[1], time[2]])
    stop_times.loc[index, 'departure_time'] = time

for index in stop_times[stop_times.arrival_time.str.match("26:\d{1,2}:
↪\d{1,2}")].index:
    time = stop_times.loc[index, 'arrival_time'].split(":")
    time = ":".join(["02", time[1], time[2]])
    stop_times.loc[index, 'arrival_time'] = time

for index in stop_times[stop_times.departure_time.str.match("26:\d{1,2}:
↪\d{1,2}")].index:
    time = stop_times.loc[index, 'departure_time'].split(":")
    time = ":".join(["02", time[1], time[2]])
    stop_times.loc[index, 'departure_time'] = time

stop_times['departure_time'] = pd.to_datetime(stop_times['departure_time'],␣
↪format='%H:%M:%S').dt.time
stop_times['arrival_time'] = pd.to_datetime(stop_times['arrival_time'],␣
↪format='%H:%M:%S').dt.time
```

[22]:
```
start = pd.to_datetime("07:00:00", format="%H:%M:%S").time()
end = pd.to_datetime("09:00:00", format="%H:%M:%S").time()

stop_times_7_9 = stop_times[(stop_times['departure_time'] > start) &␣
↪(stop_times['departure_time'] < end) &
          (stop_times['arrival_time'] > start) &  (stop_times['arrival_time']␣
↪< end)]
```

[23]:
```
trips_flinders_street = trips[trips.trip_headsign=='City (Flinders Street)']
```

[39]:
```
# Transfer_flag
transfer_flag = {}
stop_times_trips = pd.merge(left=stop_times_7_9, right=trips_flinders_street,␣
↪on='trip_id')
for stop in list(stops.stop_id):
    if not stop_times_trips[stop_times_trips.stop_id == stop].empty:
        transfer_flag[stop] = 1
```

```
    else:
        transfer_flag[stop] = 0
```

[48]:
```
for index in real_state.index:
    try:
        lat = real_state.loc[index, 'lat'].values[0]
        lng = real_state.loc[index, 'lng'].values[0]
        location = np.array((lat, lng))

        id, dist = closest(location, sc_id, sc_coordinates)
        real_state.loc[index, 'Shopping_center_id'] = id
        real_state.loc[index, 'Distance_to_sc'] = dist

        id, dist = closest(location, ts_id, ts_coordinates)
        real_state.loc[index, 'Train_station_id'] = id
        real_state.loc[index, 'Distance_to_train_station'] = dist

        real_state.loc[index, 'Transfer_flag'] = transfer_flag[id]

        id, dist = closest(location, h_id, h_coordinates)
        real_state.loc[index, 'Hospital_id'] = id
        real_state.loc[index, 'Distance_to_hospital'] = dist

        id, dist = closest(location, sm_id, sm_coordinates)
        real_state.loc[index, 'Supermarket_id'] = id
        real_state.loc[index, 'Distance_to_supermarket'] = dist
    except:
#         print(real_state.loc[index])
        pass
```

[49]:
```
real_state
```

[49]:

	property_id	lat	lng	\
0	41387	-37.7451	145.065	
1	72107	-37.8387	145.263	
2	51703	-37.7807	145.124	
3	37969	-37.7756	145.02	
4	92396	-37.9746	145.061	
…	…	…	…	
996	24209	-37.697565000000004	144.97390900000002	
997	80370	-37.986048	145.158073	
998	75144	-37.89724731	145.0753326	
999	80699	-37.97064972	145.2060394	
1000	54903	-37.848088	145.072983	

	addr_street	price	property_type	year	bedrooms	bathrooms	\
0	1/46 Hillside Road	7200000	house	2015	3	2	

```
1        19 CHURCH STREET  19550000         house  2008        3         2
2     84 Williamsons Road   9630000         house  2013        3         1
3       126 Arthur Street   7380000         house  2014        3         2
4       68 Latrobe street  12558000         house  2015        3         1
…                     …       …           …   …      …       …
996     24 Beccles Street   7684000         house  2011        3         1
997         56 Putt Grove   9975000         house  2016        4         2
998     4/26A Howe Street   3618000         house  2009        2         1
999        10 McNab Court   3036000         house  2008        3         1
1000     1078 Toorak Road   4160000         house  2014        3         1

     parking_space  … Shopping_center_id Distance_to_sc  Train_station_id  \
0                2  …              SC_017       1.231520             19936
1                2  …              SC_032       4.153883             19870
2                2  …              SC_091       0.726772             20042
3                0  …              SC_001       2.141175             19930
4                2  …              SC_003       2.474721             19865
…              …  …                   …              …                …
996              1  …              SC_041       2.354717             19941
997              2  …              SC_032       4.377967             19871
998              1  …              SC_017       4.460887             19984
999              1  …              SC_062       3.247932             45795
1000             1  …              SC_001       2.610461             19929

     Distance_to_train_station  travel_min_to_CBD  Transfer_flag  \
0                     0.273541                  0              1
1                     0.580542                  0              1
2                     4.290689                  0              1
3                     0.455132                  0              1
4                     0.879556                  0              1
…                        …                   …            …
996                   0.478876                  0              1
997                   2.760415                  0              1
998                   0.954687                  0              1
999                   1.624100                  0              1
1000                  0.968075                  0              1

       Hospital_id Distance_to_hospital  Supermarket_id  \
0     hospital_066             1.161168           S_100
1     hospital_002             1.924596           S_139
2     hospital_194             1.134131           S_188
3     hospital_029             1.586274           S_219
4     hospital_133             0.888969           S_011
…            …                 …              …
996   hospital_126             2.270870           S_107
997   hospital_179             0.669890           S_226
998   hospital_075             2.181000           S_020
```

```
999   hospital_053             2.839493          S_096
1000  hospital_035             0.999720          S_105

     Distance_to_supermarket
0                   1.597652
1                   0.597170
2                   0.697544
3                   2.120799
4                   0.781919
…                      …
996                 1.860598
997                 2.083430
998                 1.890235
999                 2.244863
1000                2.363383

[2011 rows x 21 columns]
```

[51]:
```
import shapefile
from Shapely.geometry import Point # Point class
from Shapely.geometry import shape # shape() is a function to convert geo␣
 ↪objects through the interface

# point_to_check = (1234,5678) # an x,y tuple
# shp = shapefile.Reader('path/to/shp') #open the shapefile
# all_shapes = shp.shapes() # get all the polygons
# all_records = shp.records()
# for i in len(all_shapes):
#     boundary = all_shapes[i] # get a boundary polygon
#     if Point(point_to_check).within(shape(boundary)): # make a point and see␣
 ↪if it's in the polygon
#         name = all_records[i][2] # get the second field of the corresponding␣
 ↪record
#         print "The point is in", name
```

```
      ␣
↪---------------------------------------------------------------------------

       ModuleNotFoundError                       Traceback (most recent call␣
↪last)

       <ipython-input-51-0f089f8fd33a> in <module>
         1 import shapefile
   ----> 2 from Shapely.geometry import Point # Point class
         3 from Shapely.geometry import shape # shape() is a function to␣
↪convert geo objects through the interface
```

```
4
5 # point_to_check = (1234,5678) # an x,y tuple
```

```
ModuleNotFoundError: No module named 'Shapely'
```

[ ]: