Assignment_week7

February 1, 2021

[1]:
```
import os
import json
from pathlib import Path
import gzip
import hashlib
import shutil
import pandas as pd
import pygeohash
import s3fs
endpoint_url='https://storage.budsc.midwest-datascience.com'
current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')
if results_dir.exists():
    shutil.rmtree(results_dir)
results_dir.mkdir(parents=True, exist_ok=True)
def read_jsonl_data():
    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )
    src_data_path = 'data/processed/openflights/routes.jsonl.gz'
    with s3.open(src_data_path, 'rb') as f_gz:
        with gzip.open(f_gz, 'rb') as f:
            records = [json.loads(line) for line in f.readlines()]
    return records
def flatten_record(record):
    flat_record = dict()
    for key, value in record.items():
        if key in ['airline', 'src_airport', 'dst_airport']:
            if isinstance(value, dict):
                for child_key, child_value in value.items():
                    flat_key = '{}_{}'.format(key, child_key)
                    flat_record[flat_key] = child_value
        else:
            flat_record[key] = value
```

```
    return flat_record
def create_flattened_dataset():
    records = read_jsonl_data()
    parquet_path = results_dir.joinpath('routes-flattened.parquet')
    return pd.DataFrame.from_records([flatten_record(record) for record in␣
 ↪records])
df = create_flattened_dataset()
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].
 ↪astype(str) + df['airline_iata'].astype(str)
```

[4]:
```
partitions = (
        ('A', 'A'), ('B', 'B'), ('C', 'D'), ('E', 'F'),
        ('G', 'H'), ('I', 'J'), ('K', 'L'), ('M', 'M'),
        ('N', 'N'), ('O', 'P'), ('Q', 'R'), ('S', 'T'),
        ('U', 'U'), ('V', 'V'), ('W', 'X'), ('Y', 'Z')
    )
```

[10]:
```
part_dict = {}
for k in partitions:
    if k[0] == k[1]:
        kv_key = k[0]
    else:
        kv_key = k[0]+'-'+k[1]
    part_dict[k] = kv_key
```

[15]:
```
part_dict
```

[15]:
```
{('A', 'A'): 'A',
 ('B', 'B'): 'B',
 ('C', 'D'): 'C-D',
 ('E', 'F'): 'E-F',
 ('G', 'H'): 'G-H',
 ('I', 'J'): 'I-J',
 ('K', 'L'): 'K-L',
 ('M', 'M'): 'M',
 ('N', 'N'): 'N',
 ('O', 'P'): 'O-P',
 ('Q', 'R'): 'Q-R',
 ('S', 'T'): 'S-T',
 ('U', 'U'): 'U',
 ('V', 'V'): 'V',
 ('W', 'X'): 'W-X',
 ('Y', 'Z'): 'Y-Z'}
```

[26]:
```
def get_key(str_key):
    for k,v in part_dict.items():
        if str_key[0] == k[0] or str_key[0] == k[1]:
```

```
            return v
    return ''
```

[27]:
```
df['kv_key'] = df['key'].apply(get_key)
```

[47]:
```
df.head()
```

[47]:

```
   airline_airline_id airline_name           airline_alias airline_iata  \
0                 410   Aerocondor  ANA All Nippon Airways           2B
1                 410   Aerocondor  ANA All Nippon Airways           2B
2                 410   Aerocondor  ANA All Nippon Airways           2B
3                 410   Aerocondor  ANA All Nippon Airways           2B
4                 410   Aerocondor  ANA All Nippon Airways           2B

  airline_icao airline_callsign airline_country  airline_active  \
0          ARD       AEROCONDOR        Portugal            True
1          ARD       AEROCONDOR        Portugal            True
2          ARD       AEROCONDOR        Portugal            True
3          ARD       AEROCONDOR        Portugal            True
4          ARD       AEROCONDOR        Portugal            True

   src_airport_airport_id              src_airport_name  … dst_airport_dst  \
0                  2965.0    Sochi International Airport  …               N
1                  2966.0             Astrakhan Airport  …               N
2                  2966.0             Astrakhan Airport  …               N
3                  2968.0  Chelyabinsk Balandino Airport  …               N
4                  2968.0  Chelyabinsk Balandino Airport  …               N

  dst_airport_tz_id dst_airport_type dst_airport_source  codeshare  equipment  \
0     Europe/Moscow          airport        OurAirports      False      [CR2]
1     Europe/Moscow          airport        OurAirports      False      [CR2]
2     Europe/Moscow          airport        OurAirports      False      [CR2]
3     Europe/Moscow          airport        OurAirports      False      [CR2]
4  Asia/Krasnoyarsk          airport        OurAirports      False      [CR2]

        key  kv_key hash_key src_airport_geohash
0  AERKZN2B       A        6        szsrjjzd02b3
1  ASFKZN2B       A        9        v04pk3t5gbjj
2  ASFMRV2B       A        1        v04pk3t5gbjj
3  CEKKZN2B     C-D        3        v3gdxs17du83
4  CEKOVB2B     C-D        1        v3gdxs17du83

[5 rows x 42 columns]
```

[30]:
```
import os
os.getcwd()
```

[30]:
```
'/home/jovyan/dsc650_bellevue_master/dsc650/assignments/assignment07'
```

[36]:
```
df.to_parquet(os.getcwd()+'/results/kv.parquet',partition_cols=['kv_key'])
```

[185]:
```
!ls '/home/jovyan/dsc650_bellevue_master/dsc650/assignments/assignment07/
 ↪results/kv.parquet'
```

```
'kv_key='     'kv_key=E-F'  'kv_key=M'    'kv_key=S-T'  'kv_key=Y-Z'
'kv_key=A'    'kv_key=G-H'  'kv_key=N'    'kv_key=U'
'kv_key=B'    'kv_key=I-J'  'kv_key=O-P'  'kv_key=V'
'kv_key=C-D'  'kv_key=K-L'  'kv_key=Q-R'  'kv_key=W-X'
```

[40]:
```
import hashlib

def hash_key(key):
    m = hashlib.sha256()
    m.update(str(key).encode('utf-8'))
    return m.hexdigest()[0]
```

[41]:
```
df['hash_key'] = df['key'].apply(hash_key)
```

[43]:
```
df.to_parquet(os.getcwd()+'/results/hash.parquet',partition_cols=['hash_key'])
```

[186]:
```
!ls '/home/jovyan/dsc650_bellevue_master/dsc650/assignments/assignment07/
 ↪results/hash.parquet'
```

```
'hash_key=0'  'hash_key=4'  'hash_key=8'  'hash_key=c'
'hash_key=1'  'hash_key=5'  'hash_key=9'  'hash_key=d'
'hash_key=2'  'hash_key=6'  'hash_key=a'  'hash_key=e'
'hash_key=3'  'hash_key=7'  'hash_key=b'  'hash_key=f'
```

[ ]:
```
* West * The Dalles, Oregon * Latitude: 45.5945645 * Longitude: -121.1786823
* Central * Papillion, NE * Latitude: 41.1544433 * Longitude: -96.0422378
* East * Loudoun County, Virginia * Latitude: 39.08344 * Longitude: -77.6497145
```

[104]:
```
df['src_airport_geohash'] = df.apply(
    lambda row: pygeohash.encode(row.src_airport_latitude, row.
 ↪src_airport_longitude), axis=1
)
def determine_location(src_airport_geohash):
    locations = dict(
        central=pygeohash.encode(41.1544433, -96.0422378),
        west = pygeohash.encode(45.5945645,-121.1786823 ),
        east = pygeohash.encode(39.08344,-77.6497145)
    )
    #distances = #TODO: a list of centers and distances using the
 ↪pygeohash.geohash_haversine_distance function
```

```
    #distances = list(pygeohash.
↪geohash_haversine_distance(src_airport_geohash,k) for k in locations.keys())
    distances = []
    for k in locations.keys():
        distances.append(pygeohash.
↪geohash_haversine_distance(k,src_airport_geohash))

    distances.sort()
    return distances[0][1]
df['location'] = df['src_airport_geohash'].apply(determine_location)
df.to_parquet(os.getcwd()+'/results/geo.parquet', partition_cols=['location'])
```

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

       KeyError                                  Traceback (most recent call␣
↪last)

       <ipython-input-104-756bb8e15a2f> in <module>
        16     distances.sort()
        17     return distances[0][1]
   ---> 18 df['location'] = df['src_airport_geohash'].apply(determine_location)
        19 df.to_parquet(os.getcwd()+'/results/geo.parquet',␣
↪partition_cols=['location'])


       /opt/conda/lib/python3.8/site-packages/pandas/core/series.py in␣
↪apply(self, func, convert_dtype, args, **kwds)
      3846             else:
      3847                 values = self.astype(object).values
   -> 3848                 mapped = lib.map_infer(values, f,␣
↪convert=convert_dtype)
      3849
      3850         if len(mapped) and isinstance(mapped[0], Series):


       pandas/_libs/lib.pyx in pandas._libs.lib.map_infer()


       <ipython-input-104-756bb8e15a2f> in␣
↪determine_location(src_airport_geohash)
        12     distances = []
        13     for k in locations.keys():
   ---> 14         distances.append(pygeohash.
↪geohash_haversine_distance(k,src_airport_geohash))
        15
```

```
        16     distances.sort()


      /opt/conda/lib/python3.8/site-packages/pygeohash/distances.py in␣
↪geohash_haversine_distance(geohash_1, geohash_2)
        79     """
        80
   ---> 81     lat_1, lon_1 = decode(geohash_1)
        82     lat_2, lon_2 = decode(geohash_2)
        83


      /opt/conda/lib/python3.8/site-packages/pygeohash/geohash.py in␣
↪decode(geohash)
        70     containing only relevant digits and with trailing zeroes removed.
        71     """
   ---> 72     lat, lon, lat_err, lon_err = decode_exactly(geohash)
        73     # Format to the number of decimals that are known
        74     lats = "%.*f" % (max(1, int(round(-log10(lat_err)))) - 1, lat)


      /opt/conda/lib/python3.8/site-packages/pygeohash/geohash.py in␣
↪decode_exactly(geohash)
        45     is_even = True
        46     for c in geohash:
   ---> 47         cd = __decodemap[c]
        48         for mask in [16, 8, 4, 2, 1]:
        49             if is_even: # adds longitude info


      KeyError: 'a'
```

[180]:
```
def balance_partitions(keys, num_partitions):
    partitions = []
    p =1
    kmax = max(keys)
    kmin = min(keys)
    k_range = kmax - kmin
    k_incr = math.ceil(k_range / num_partitions)
    for i,key in enumerate(keys):
        if i+1 > k_incr *p:
            p = p+1
        partitions.append(p)
    return partitions
```

[181]:
```
keys = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
num_partitions = 4
```

[182]:
```
balance_partitions(keys,num_partitions)
```

[182]:
```
[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4]
```