assignment7

July 23, 2021

[1]:

```
# 7.1 a
```

[2]:

```
import os
import json
from pathlib import Path
import gzip
import hashlib
import shutil
import pandas as pd
import pygeohash
import s3fs
```

[3]:

```
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

# flatten dataset
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

# key with dataframe
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')
    )
```

[5]:

```
# create this directory structure is to create a new key called kv_key from the␣
 ↪key column and use the to_parquet method
# with partition_cols=['kv_key'] to save a partitioned dataset

partition_dict = {}
for key in partitions:
    if key[0] == key[1]:
        kv_key = key[0]
    else:
        kv_key = key[0] + '-' + key[1]
    partition_dict[key] = kv_key
```

[6]:

```
partition_dict
```

[6]:

```
{('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'}
```

[7]:

```
def get_key(s_key):
    for key, value in partition_dict.items():
        if s_key[0] == key[0] or s_key[0] == key[1]:
            return value
    return ' '

# add kv_key column
df['kv_key'] = df['key'].apply(get_key)
```

[8]:

```
# test; successfully run
df.to_csv('test', sep = ',')
```

[9]:

```
# move df to results folder
df.to_parquet(os.getcwd() + '/results/kv.parquet', partition_cols = ['kv_key'])
```

[10]:

```
# 7.1 b
```

[11]:

```
import hashlib

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

[12]:

```
# create new hash column, hashed value of key column
df['key'] = df['src_airport_iata'].astype(str) + df['dst_airport_iata'].
 ↪astype(str) + df['airline_iata'].astype(str)

df['hashed'] = df.apply(lambda x: hash_key(x.key), axis=1)

df['hash_key'] = df['hashed'].str[:1]
```

[13]:

```
# test to csv; successful
df.to_csv('test1', sep = ',')
```

[14]:
```
# move new column to df in results folder
df.to_parquet(os.getcwd() + '/results/hash.parquet', partition_cols =␣
 ↪['hash_key'])
```

[15]:
```
# 7.1 c
```

[17]:
```
! pip install geolib
```

```
Collecting geolib
  Downloading geolib-1.0.7-py3-none-any.whl (5.5 kB)
Collecting future
  Downloading future-0.18.2.tar.gz (829 kB)
     |                    | 829 kB 4.7 MB/s eta 0:00:01
Building wheels for collected packages: future
  Building wheel for future (setup.py) … done
  Created wheel for future: filename=future-0.18.2-py3-none-any.whl
size=491059
sha256=3751c4d160ba61217bad401d1463d2ea322a0e7509ff28374f5f054c3eefb342
  Stored in directory: /home/jovyan/.cache/pip/wheels/8e/70/28/3d6ccd6e315f65f24
5da085482a2e1c7d14b90b30f239e2cf4
Successfully built future
Installing collected packages: future, geolib
Successfully installed future-0.18.2 geolib-1.0.7
```

[18]:
```
import pandas as pd
import numpy as np
import sklearn.neighbors
from geolib import geohash
```

[20]:
```
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),
        east = pygeohash.encode(39.08344, -77.6497145),
        west = pygeohash.encode(45.5945645, -121.1786823))

    distances = []
    for location, geohash in locations.items():
        hav = pygeohash.geohash_haversine_distance(src_airport_geohash, geohash)
        distances.append(tuple((hav, location)))

    distances.sort()
    return distances[0][1]
```

```
# new column in df
df['location'] = df['src_airport_geohash'].apply(determine_location)
```

[21]:

```
# test; successful
df.to_csv('geo_test', sep = ',')
```

[22]:

```
# df to results folder
df.to_parquet('results/geo', partition_cols=['location'])
```

[23]:

```
# 7.1 d
```

[24]:

```
def balance_partitions (keys, num_partitions):
    vals = sorted(set(keys))
    num_vals = len(vals)
    partition_counts = (num_vals / num_partitions)+1
    partitions = []
    x = 1
    y = 1
    for i in range(num_vals):
        key_val ={}
        if x <= partition_counts:
            key_val[vals[i]] = y
            x = x + 1
        else:
            x = 1
            y = y + 1
            key_val[vals[i]] = y
            x = x + 1
        partitions.append(key_val)
    return partitions
```

[25]:

```
# list of keys
keys = ['cat', 'duck', 'chicken', 'rabbit', 'horse', 'cow', 'donkey', 'dog',␣
 ↪'goose', 'mouse', 'sheep']
```

[26]:

```
# number of partitions
num_partitions = 3
```

[27]:

```
partitions = balance_partitions(keys, num_partitions)
print(partitions)
```

```
[{'cat': 1}, {'chicken': 1}, {'cow': 1}, {'dog': 1}, {'donkey': 2}, {'duck': 2},
{'goose': 2}, {'horse': 2}, {'mouse': 3}, {'rabbit': 3}, {'sheep': 3}]
```

[30]:

```
num_partitions = 4
```

[31]:

```
partitions = balance_partitions(keys, num_partitions)
print(partitions)
```

```
[{'cat': 1}, {'chicken': 1}, {'cow': 1}, {'dog': 2}, {'donkey': 2}, {'duck': 2},
{'goose': 3}, {'horse': 3}, {'mouse': 3}, {'rabbit': 4}, {'sheep': 4}]
```

[ ]: