Assignment 3

June 25, 2021

1 Assignment 3

Import libraries and define common helper functions

[53]:
```
import os
import sys
import gzip
import json
from pathlib import Path
import csv

import pandas as pd
import s3fs
import pyarrow as pa
from pyarrow.json import read_json
import pyarrow.parquet as pq
import fastavro
import pygeohash
import snappy
import jsonschema
from jsonschema.exceptions import ValidationError


endpoint_url='https://storage.budsc.midwest-datascience.com'

current_dir = Path(os.getcwd()).absolute()
schema_dir = current_dir.joinpath('schemas')
results_dir = current_dir.joinpath('results')
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
```

Load the records from https://storage.budsc.midwest-datascience.com/data/processed/openflights/routes.jsonl.gz

[54]:
```
records = read_jsonl_data()
```

[55]:
```
# view what records looks like
records[0:2]
```

[55]:
```
[{'airline': {'airline_id': 410,
   'name': 'Aerocondor',
   'alias': 'ANA All Nippon Airways',
   'iata': '2B',
   'icao': 'ARD',
   'callsign': 'AEROCONDOR',
   'country': 'Portugal',
   'active': True},
  'src_airport': {'airport_id': 2965,
   'name': 'Sochi International Airport',
   'city': 'Sochi',
   'country': 'Russia',
   'iata': 'AER',
   'icao': 'URSS',
   'latitude': 43.449902,
   'longitude': 39.9566,
   'altitude': 89,
   'timezone': 3.0,
   'dst': 'N',
   'tz_id': 'Europe/Moscow',
   'type': 'airport',
   'source': 'OurAirports'},
  'dst_airport': {'airport_id': 2990,
   'name': 'Kazan International Airport',
   'city': 'Kazan',
   'country': 'Russia',
   'iata': 'KZN',
   'icao': 'UWKD',
   'latitude': 55.606201171875,
   'longitude': 49.278701782227,
   'altitude': 411,
   'timezone': 3.0,
```

```
  'dst': 'N',
  'tz_id': 'Europe/Moscow',
  'type': 'airport',
  'source': 'OurAirports'},
 'codeshare': False,
 'equipment': ['CR2']},
{'airline': {'airline_id': 410,
  'name': 'Aerocondor',
  'alias': 'ANA All Nippon Airways',
  'iata': '2B',
  'icao': 'ARD',
  'callsign': 'AEROCONDOR',
  'country': 'Portugal',
  'active': True},
 'src_airport': {'airport_id': 2966,
  'name': 'Astrakhan Airport',
  'city': 'Astrakhan',
  'country': 'Russia',
  'iata': 'ASF',
  'icao': 'URWA',
  'latitude': 46.2832984924,
  'longitude': 48.0063018799,
  'altitude': -65,
  'timezone': 4.0,
  'dst': 'N',
  'tz_id': 'Europe/Samara',
  'type': 'airport',
  'source': 'OurAirports'},
 'dst_airport': {'airport_id': 2990,
  'name': 'Kazan International Airport',
  'city': 'Kazan',
  'country': 'Russia',
  'iata': 'KZN',
  'icao': 'UWKD',
  'latitude': 55.606201171875,
  'longitude': 49.278701782227,
  'altitude': 411,
  'timezone': 3.0,
  'dst': 'N',
  'tz_id': 'Europe/Moscow',
  'type': 'airport',
  'source': 'OurAirports'},
 'codeshare': False,
 'equipment': ['CR2']}]
```

1.1 3.1

1.1.1 3.1.a JSON Schema

[56]:
```
import csv
```

[57]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')
    with open(schema_path) as f:
        schema = json.load(f)

    # used code from Teams discussion thread
    # needed to put validation_csv_path in quotes because I kept getting␣
 ↪NameError
    with open('validation_csv_path', 'w', encoding='utf-8') as f:
        #create column names
        fieldnames = ['row_num', 'is_valid', 'msg']

        #assign CSV writer object
        csv_writer = csv.DictWriter(f, fieldnames=fieldnames, lineterminator =␣
 ↪'\n')
        csv_writer.writeheader()

        #iterate over all the records & verify they align with the schema
        for i, record in enumerate(records):
            try:
                result = dict(row_num=i, is_valid=True, msg=record)
            except ValidationError as e:
                result = dict(row_num=i, is_valid=False, msg=record)
            finally:
                csv_writer.writerow(result)

validate_jsonl_data(records)
```

1.1.2 3.1.b Avro

[58]:
```
from fastavro import writer, parse_schema
```

[59]:
```
def create_avro_dataset(records):
    schema_path = schema_dir.joinpath('routes.avsc')
    data_path = results_dir.joinpath('routes.avro')
    ## TODO: Use fastavro to create Avro dataset
    with open(schema_path, 'r') as f1:
        schema = json.loads(f1.read())
    parsed_schema = fastavro.parse_schema(schema)
    ## create dataset
    with open(data_path, 'wb') as out:
        fastavro.writer(out, parsed_schema, records)
```

```
create_avro_dataset(records)
```

[68]:
```
# Check if file was created successfully
# view contents
data_path = results_dir.joinpath('routes.avro')
with open(data_path, mode = 'rb') as f:
    reader = fastavro.reader(f)
    records = [r for r in reader]
    df = pd.DataFrame.from_records(records)
    print(df.head())
```

```
                                              airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
1  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
2  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
3  {'airline_id': 410, 'name': 'Aerocondor', 'ali…
4  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                         src_airport  \
0  {'airport_id': 2965, 'name': 'Sochi Internatio…
1  {'airport_id': 2966, 'name': 'Astrakhan Airpor…
2  {'airport_id': 2966, 'name': 'Astrakhan Airpor…
3  {'airport_id': 2968, 'name': 'Chelyabinsk Bala…
4  {'airport_id': 2968, 'name': 'Chelyabinsk Bala…

                                         dst_airport codeshare  stops  \
0  {'airport_id': 2990, 'name': 'Kazan Internatio…      False      0
1  {'airport_id': 2990, 'name': 'Kazan Internatio…      False      0
2  {'airport_id': 2962, 'name': 'Mineralnyye Vody…      False      0
3  {'airport_id': 2990, 'name': 'Kazan Internatio…      False      0
4  {'airport_id': 4078, 'name': 'Tolmachevo Airpo…      False      0

  equipment
0     [CR2]
1     [CR2]
2     [CR2]
3     [CR2]
4     [CR2]
```

1.1.3 3.1.c Parquet

[60]:
```
def create_parquet_dataset():
    src_data_path = 'data/processed/openflights/routes.jsonl.gz'
    parquet_output_path = results_dir.joinpath('routes.parquet')
    s3 = s3fs.S3FileSystem(
        anon=True,
```

```
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )

    with s3.open(src_data_path, 'rb') as f_gz:
        with gzip.open(f_gz, 'rb') as f:
            pass
            ## TODO: Use Apache Arrow to create Parquet table and save the␣
↪dataset
            table = read_json(f)
    pq.write_table(table, parquet_output_path)

create_parquet_dataset()
```

[69]:
```
# Check if file was created successfully
# view contents
parquet_output_path = results_dir.joinpath('routes.parquet')
pqFile = pq.ParquetFile(parquet_output_path)
pqFile.metadata
```

[69]:
```
<pyarrow._parquet.FileMetaData object at 0x7fce803ae5e0>
  created_by: parquet-cpp version 1.5.1-SNAPSHOT
  num_columns: 38
  num_rows: 67663
  num_row_groups: 1
  format_version: 1.0
  serialized_size: 7571
```

1.1.4 3.1.d Protocol Buffers

[61]:
```
sys.path.insert(0, os.path.abspath('routes_pb2'))

import routes_pb2

def _airport_to_proto_obj(airport):
    obj = routes_pb2.Airport()
    if airport is None:
        return None
    if airport.get('airport_id') is None:
        return None

    obj.airport_id = airport.get('airport_id')
    if airport.get('name'):
        obj.name = airport.get('name')
    if airport.get('city'):
        obj.city = airport.get('city')
```

```
    if airport.get('iata'):
        obj.iata = airport.get('iata')
    if airport.get('icao'):
        obj.icao = airport.get('icao')
    if airport.get('altitude'):
        obj.altitude = airport.get('altitude')
    if airport.get('timezone'):
        obj.timezone = airport.get('timezone')
    if airport.get('dst'):
        obj.dst = airport.get('dst')
    if airport.get('tz_id'):
        obj.tz_id = airport.get('tz_id')
    if airport.get('type'):
        obj.type = airport.get('type')
    if airport.get('source'):
        obj.source = airport.get('source')

    obj.latitude = airport.get('latitude')
    obj.longitude = airport.get('longitude')

    return obj



def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:
        route = routes_pb2.Route()
        ## TODO: Implement the code to create the Protocol Buffers Dataset
        airline = _airport_to_proto_obj(record.get('airline', {}))
        if airline:
            route.airline.CopyFrom(airline)
        src_airport = _airport_to_proto_obj(record.get('src_airport', {}))
        if src_airport:
            route.src_airport.CopyFrom(src_airport)
        dst_airport = _airport_to_proto_obj(record.get('dst_airport', {}))
        if dst_airport:
            route.dst_airport.CopyFrom(dst_airport)

        if record.get('codeshare'):
            route.codeshare = record.get('codeshare')
        else:
            route.codeshare = False

        if record.get('stops'):
            route.stops = record.get('stops')
```

```
        equipment = record.get('equipment')

        if len(equipment) > 1:
            for i, v in enumerate(equipment):
                route.equipment.append(v)
        else:
            equpiment = record.get('equipment')

        routes.route.append(route)

    data_path = results_dir.joinpath('routes.pb')

    with open(data_path, 'wb') as f:
        f.write(routes.SerializeToString())

    compressed_path = results_dir.joinpath('routes.pb.snappy')

    with open(compressed_path, 'wb') as f:
        f.write(snappy.compress(routes.SerializeToString()))

create_protobuf_dataset(records)
```

1.1.5 3.1 e Output Sizes

[72]:
```
# Path to each file
json_schema_path = schema_dir.joinpath('routes-schema.json')
avro_schema_path = schema_dir.joinpath('routes.avsc')
par_path = results_dir.joinpath('routes.parquet')
proto_schema_path = schema_dir.joinpath('routes.proto')

# Getting size for the paths defined above
json_schema_size = os.path.getsize(json_schema_path)
avro_schema_size = os.path.getsize(avro_schema_path)
par_size = os.path.getsize(par_path)
proto_schema_size = os.path.getsize(proto_schema_path)

# Create a dataframe with the sizes
df = pd.DataFrame({'JSON Schema': [json_schema_size], 'Avro':␣
↪[avro_schema_size], 'Parquet': [par_size], 'Protocol Buffer':␣
↪[proto_schema_size]})
df
```

[72]:

	JSON Schema	Avro	Parquet	Protocol Buffer
0	4	3191	1975469	1073

[73]:
```
## Write Result to Comparison.csv as described in directions
compare = results_dir.joinpath('comparison.csv')
with open (compare, 'w') as f:
    df.to_csv(f, header = True)
```

1.2 3.2

1.2.1 3.2.a Simple Geohash Index

[62]:
```
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)
    hashes = []
    ## TODO: Create hash index
    for record in records:
        src_airport = record.get('src_airport', {})
        if src_airport:
            latitude = src_airport.get('latitude')
            longitude = src_airport.get('longitude')
            if latitude and longitude:
                hashes.append(pygeohash.encode(latitude, longitude))
    hashes.sort()

    three_letter = sorted(list(set([entry[:3] for entry in hashes])))

    hash_index = {value: [] for value in three_letter}

    for record in records:
        geohash = record.get('geohash')
        if geohash:
            hash_index[geohash[:3]].append(record)

    for key, values in hash_index.items():
        output_dir = geoindex_dir.joinpath(str(key[:1])).joinpath(str(key[:2]))
        output_dir.mkdir(exist_ok=True, parents=True)
        output_path = output_dir.joinpath('{}.jsonl.gz'.format(key))
        with gzip.open(output_path, 'w') as f:
            json_output = '\n'.join([json.dumps(value) for value in values])
            f.write(json_output.encode('utf-8'))

create_hash_dirs(records)
```

1.2.2 3.2.b Simple Search Feature

[63]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport
    a = pygeohash.encode(latitude, longitude)
    dist = 0
    name = ''

    for i, record in enumerate(records):
        src_airport = record.get('src_airport', {})
        if src_airport:
            lat = src_airport.get('latitude')
            long = src_airport.get('longitude')
            airport_name = src_airport.get('name')
            if lat and long:
                a1 = pygeohash.encode(lat, long)

                dist_n = pygeohash.geohash_approximate_distance(a, a1)
                if i==0:
                    dist = dist_n
                    name = airport_name
                else:
                    if dist > dist_n:
                        dist = dist_n
                        name = airport_name
    print(name)
```

[64]:
```
airport_search(41.1499988, -95.91779)
```

```
Eppley Airfield
```

[65]:
```
# airport search for MKE airport
# google searched coordinates: 42.9476° N, -87.8966° W
airport_search(42.9476, -87.8966)
```

```
General Mitchell International Airport
```

[ ]: