Assignment 3

April 5, 2021

1 Assignment 3

Import libraries and define common helper functions

[1]:
```
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

[2]:
```
records = read_jsonl_data()
```

[1]:
```
#records
```

1.1 3.1

1.1.1 3.1.a JSON Schema

[26]:
```
def validate_jsonl_data(records):
    schema_path = schema_dir.joinpath('routes-schema.json')

    validation_csv_path = results_dir.joinpath('validation-results.csv')

    with open(schema_path) as f:
        schema = json.load(f)

    with open(validation_csv_path, 'w') as f:
        for i, record in enumerate(records):
            try:
                ## TODO: Validate record

                jsonschema.validate(instance = record, schema = schema)

                pass
            except ValidationError as e:
                ## Print message if invalid record

                f.write(str(e))

                pass


validate_jsonl_data(records)
```

1.1.2 3.1.b Avro

[19]:
```
import fastavro
from fastavro.schema import load_schema
from fastavro import writer
```

[25]:
```
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

    parsed_schema = load_schema(schema_path)
    with open(data_path, 'wb') as out:
        writer(out, parsed_schema, records)

create_avro_dataset(records)
```

[24]:
```
#  check load
import pandas as pd

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

with open (data_path, mode = 'rb') as fp:
    reader = fastavro.reader(fp)
    records = [r for r in reader]

    df = pd.DataFrame.from_records(records)
    print(df.head(1))
```

```
                                             airline  \
0  {'airline_id': 410, 'name': 'Aerocondor', 'ali…

                                          src_airport  \
0  {'airport_id': 2965, 'name': 'Sochi Internatio…

                                          dst_airport  codeshare  stops  \
0  {'airport_id': 2990, 'name': 'Kazan Internatio…      False      0

  equipment
0     [CR2]
```

1.1.3 3.1.c Parquet

[31]:
```
import numpy as np
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq
```

```
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:


            for i in f:
                pass









            pass
            ## TODO: Use Apache Arrow to create Parquet table and save the␣
 ↪dataset
            table = pa.Table.from_pandas(f)
            pa.write_table(f, parquet_output_path)

            #pq.write_table(table, 'example.parquet')



create_parquet_dataset()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-31-d15e70b0a9fa> in <module>
     35
     36
---> 37 create_parquet_dataset()

<ipython-input-31-d15e70b0a9fa> in create_parquet_dataset()
     28             pass
     29             ## TODO: Use Apache Arrow to create Parquet table and save␣
 ↪the dataset
---> 30             table = pa.Table.from_pandas(f)
     31             pa.write_table(f, parquet_output_path)
```

```
     32

/opt/conda/lib/python3.8/site-packages/pyarrow/table.pxi in pyarrow.lib.Table.
 ↪from_pandas()

/opt/conda/lib/python3.8/site-packages/pyarrow/pandas_compat.py in␣
 ↪dataframe_to_arrays(df, schema, preserve_index, nthreads, columns, safe)
    547       index_columns,
    548       columns_to_convert,
--> 549       convert_fields) = _get_columns_to_convert(df, schema,␣
 ↪preserve_index,
    550                                                 columns)
    551

/opt/conda/lib/python3.8/site-packages/pyarrow/pandas_compat.py in␣
 ↪_get_columns_to_convert(df, schema, preserve_index, columns)
    328
    329 def _get_columns_to_convert(df, schema, preserve_index, columns):
--> 330     columns = _resolve_columns_of_interest(df, schema, columns)
    331
    332     if not df.columns.is_unique:

/opt/conda/lib/python3.8/site-packages/pyarrow/pandas_compat.py in␣
 ↪_resolve_columns_of_interest(df, schema, columns)
    502         columns = [c for c in columns if c in df.columns]
    503     else:
--> 504         columns = df.columns
    505
    506     return columns

AttributeError: 'GzipFile' object has no attribute 'columns'
```

1.1.4 3.1.d Protocol Buffers

[32]:
```
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 _airline_to_proto_obj(airline):
    obj = routes_pb2.Airline()
    if not airline.get('name'):
        return None
    if not airline.get('airline_id'):
        return None
    if not airline.get('active'):
        return None

    obj.airline_id = airline.get('airline_id')
    obj.name = airline.get('name')
    if airline.get('alias'):
        obj.alias = airline.get('alias')

    ## TODO

    #obj.airline_id = airline.get('airline_id')
    #if airline.get('name'):
    #    obj.name = airline.get('name')
    #if airline.get('alias'):
    #    obj.alias = airline.get('alias')
```

```
    #if airline.get('iata'):
    #    obj.iata = airline.get('iata')
    #if airline.get('icao'):
    #    obj.icao = airline.get('icao')
    #if airline.get('callsign'):
    #    obj.callsign = airline.get('callsign')
    #if airline.get('country'):
    #    obj.country = airline.get('country')

    return obj



def create_protobuf_dataset(records):
    routes = routes_pb2.Routes()
    for record in records:

        route = routes_pb2.Route()
        airline = _airline_to_proto_obj(record.get('airline', {}))

        if airline:
            route.airline.CopyFrom(airline)

        src_airport = _airport_to_proto_obj(record.get('src_airport', {}))

        ## TODO: Implement the code to create the Protocol Buffers Dataset


        if src_airport:

            route.src_airport.CopyFrom(src_airport)
            routes.route.append(route)

        codeshare = _airline_to_proto_obj(record.get('codeshare'))

        if codeshare:
            route.codeshare.CopyFrom(codeshare)
            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)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-32-7110c1887dc6> in <module>
    110         f.write(snappy.compress(routes.SerializeToString()))
    111
--> 112 create_protobuf_dataset(records)

<ipython-input-32-7110c1887dc6> in create_protobuf_dataset(records)
     93             routes.route.append(route)
     94
---> 95         codeshare = _airline_to_proto_obj(record.get('codeshare'))
     96
     97         if codeshare:

<ipython-input-32-7110c1887dc6> in _airline_to_proto_obj(airline)
     40 def _airline_to_proto_obj(airline):
     41     obj = routes_pb2.Airline()
---> 42     if not airline.get('name'):
     43         return None
     44     if not airline.get('airline_id'):

AttributeError: 'bool' object has no attribute 'get'
```

[ ]:

[ ]:

[ ]:

[ ]:

1.2 3.2

1.2.1 3.2.a Simple Geohash Index

[84]:
```
def create_hash_dirs(records):
    geoindex_dir = results_dir.joinpath('geoindex')
    geoindex_dir.mkdir(exist_ok=True, parents=True)

    records = read_jsonl_data()
```

```
    recs = pd.DataFrame(records)

    df = recs.src_airport

    a = df[0]
    print(a)

    print(a['latitude'])



    hashes = []
    ## TODO: Create hash index




create_hash_dirs(records)
```

```
{'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'}
43.449902
43.449902
```

1.2.2 3.2.b Simple Search Feature

[ ]:
```
def airport_search(latitude, longitude):
    ## TODO: Create simple search to return nearest airport
    pass

airport_search(41.1499988, -95.91779)
```