File Ingestion and Schema Validation

Virtual Internship

10-Feb-2023

Contents

- Introduction
- Dataset
- Ways to read data
- YAML file conversion
- GZIP conversion

Introduction

- A data pipeline is the series of steps that allow data from one system to move to and become useful in another system, particularly analytics, data science, or AI and machine learning systems.
- At a high level, a data pipeline works by pulling data from the source, applying rules for transformation and processing, then pushing it to its destination

- File ingestion and schema validation: In this report different ways of reading large data files (pandas, dask, chunksize, dictread, datatables, pyspark) have been explored and the most optimal way is suggested.
- The data file is then converted to YAML file and further compressed to gzip.
- Two different ways of creating yaml file has been explored using dataframes directly and second after coverting to array.

Dataset

- The dataset for parking violation ticket issued in New York in 2017 has been used.
- Data source:

https://www.kaggle.com/datasets/new-york-city/nyc-parking-tickets?select=Parking_Violations_Issued_-_Fiscal_Year_2017.csv

Total number of observations	**10803028**
Total number of files	1
Total number of features	43
Base format of the file	.csv
Size of the data	2.09 GB

Different ways to read data

Different methods as listed below, of reading the big data file has been tested and time to read the data is analysed to obtain the fastest method:

1) Pandas

Time to read using pandas (1min 21s)

```
%%time
NYC_data = pd.read_csv("Parking_Violations_Issued_-_Fiscal_Year_2017.csv")
NYC_data.head()
```

```
CPU times: user 27.8 s, sys: 15.4 s, total: 43.2 s
Wall time: 1min 21s
```

2) Pandas Chunksize

Time to read using pandas chunks (46s)

```
%%time
data_chunks = pd.read_csv("Parking_Violations_Issued_-_Fiscal_Year_2017.csv", chunksize=100000)
pandas_chunks = pd.concat(data_chunks)
pandas_chunks.head()
```

```
CPU times: user 29.9 s, sys: 8.71 s, total: 38.6 s
Wall time: 46.5 s
```

3) DictReader

Time to read using dictreader(3.45ms)

```
%%time
data_dict = csv.DictReader(open('Parking_Violations_Issued_-_Fiscal_Year_2017.csv'))
i=0
for row in data_dict:
    print(row)
    i+=1
    if i is 5:
        break
```

```
CPU times: user 788 µs, sys: 1.16 ms, total: 1.95 ms
Wall time: 3.45 ms
```

Different ways to read data

4) Datatable

Time to read using datatable(6.5s)

```
%%time
data_dt = dt.fread("Parking_Violations_Issued_-_Fiscal_Year_2017.csv")
data_dt.head()
```

```
CPU times: user 16.3 s, sys: 2.71 s, total: 19 s
Wall time: 6.5 s
```

5) Pyspark

Time to read using pyspark (32.8s)

```
%%time
spark = SparkSession.builder.appName("EDA.com").getOrCreate()
df = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("Parking_Violations_Issued_-_Fiscal_Year_2017.cs
df.show(5)
```

```
CPU times: user 36.2 ms, sys: 78.9 ms, total: 115 ms
Wall time: 32.8 s
```

6) Dask

Time to read using DASK(2s)

```
%%time
data_dask = dd.read_csv("Parking_Violations_Issued_-_Fiscal_Year_2017.csv", dtype={'House Number': 'object',
       'Time First Observed': 'object'})
data_dask.head()
```

```
CPU times: user 882 ms, sys: 382 ms, total: 1.26 s
Wall time: 2.07 s
```

Most Optimal way

- Maximum time is taken by Pandas.
- Among several ways Dictreader takes the least time (3.45ms) followed by Dask (2s).
- However, Dask maintains the pandas dataframe structure hence Dask can be preferred as compared to other methods for reading large data files.

Using YAML

- The first 10000 rows have been converted to yaml file
- The headers has been taken as the columns name.
- Keys are the headers and values are the corresponding data in the rows.

Creating YML

In [4]:
```
data=[]
# headers=data_dask.columns.values.tolist()
rows = NYC_data.iloc[1:10000].values.tolist()
print(len(rows))
```

```
9999
```

In [7]:
```
header=data_dask.columns.values.tolist()
for row in rows:
    data.append({header[i]:row[i] for i in range(len(header))})
with open('data2yml.yml','w') as f:
    yaml.dump(data,f,default_flow_style=False)
```

GZIP

- The data has been zipped using gzip.

GZIP

In [11]:

```
with open('data2yml.yml', 'rb') as f_in:
    with gzip.open('data2yml.yml.gz', 'wb') as f_out:
        shutil.copyfileobj(f_in, f_out)
```

Thank You