# Data Intake Report

Name: File ingestion and schema validation
Report date: 31/8/2022
Internship Batch: LISUM11:30
Version:<1.0>
Data intake by: Raghavi Gururajan
Data intake reviewer:<intern who reviewed the report>
Data storage location:

**Tabular data details:**

| | |
|---|---|
| **Total number of observations** | 62388 |
| **Total number of files** | 3 |
| **Total number of features** | 4 |
| **Base format of the file** | .csv |
| **Size of the data** | 2 GB |

**Step1:**
Imported the necessary libraries

```
[ ]  import dask
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import os
     import logging
     import subprocess
     import time
     import datetime
     import gc
     import gzip
     import time
     import warnings
     import csv
     import re
     import yaml
     import subprocess
     from dask import dataframe as dd
```

**Step2:**
Read the data with Modin

```
[ ]  pip install modin[all]
```

```
[ ]  #reading the data with Modin
     import modin.pandas as pd
     import ray
     ray.shutdown()
     ray.init()
     start=time.time()
     df = pd.read_csv('birds.csv')
     end=time.time()
     print("Read csv file with modin : ", (end-start), "sec")
```

**Step3:**
Read the data with pandas

```
[ ]  #read the csv file with pandas
     import pandas as pd
     import numpy as np
     import random
     import time
     np.random
     start=time.time()
     df1= pd.read_csv('birds.csv', delimiter = ",")
     end=time.time()

     print("Read csv with pandas: ",(start-end),"sec")
```

```
Read csv with pandas:  -0.21799707412719727 sec
```

**Step4:**

Read the data with Dask

```
#Read in the data with Dask
from dask import dataframe as dd
start = time.time()
df2 = dd.read_csv('birds.csv')
end = time.time()

print("Read csv with dask: ",(end-start),"sec")
```

```
Read csv with dask:  0.046927452087402344 sec
```

Considering the dask library as it has the least runtime of 0.03 seconds

**Step5:**
Imported the h5 file.

```
[ ] from dask import dataframe as dd
    df = dd.read_csv('birds.csv', delimiter = ',')
```

```
import h5py
h5py.run_tests()
```

```
1
```

```
pip install h5py
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simp
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (3.1.0)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from h5
```

```
[ ] pip install h5pyViewer
```

```
[ ]  f1 = h5py.File('/content/EfficientNetB4-BIRDS-0.99.h5','r+')
```

```
[ ]  # open the file as 'f'
     with h5py.File('/content/EfficientNetB4-BIRDS-0.99.h5', 'r') as f:
         data = f['default']

         # get the minimum value
         print(min(data))

         # get the maximum value
         print(max(data))

         # get the values ranging from index 0 to 15
         print(data[:15])
```

```
-3.532408633416863
3.2652773951341167
[-0.04235926 -0.94812048 -0.81680085 -0.34099942  1.58634012  0.54481712
 -0.22594584  0.4382418   0.24239861  1.35197092 -1.03434457 -0.25794343
  1.1940769  -0.67989923  2.32339102]
```

**Step6:**
Write YAML file

```
%%writefile utility.py

def read_config_file(filepath):
    with open(filepath, 'r') as stream:
        try:
            return yaml.load(stream, Loader=yaml.Loader)
        except yaml.YAMLError as exc:
            logging.error(exc)

def col_header_val(df,table_config):
    df.columns = df.columns.str.lower()
    df.columns = df.columns.str.replace('[^\w]','_',regex=True)
    df.columns = list(map(lambda x: x.strip('_'), list(df.columns)))
    df.columns = list(map(lambda x: replacer(x,'_'), list(df.columns)))
    expected_col = list(map(lambda x: x.lower(),  table_config['columns']))
    expected_col.sort()
    df.columns =list(map(lambda x: x.lower(), list(df.columns)))
    df = df.reindex(sorted(df.columns), axis=1)
    if len(df.columns) == len(expected_col) and list(expected_col)  == list(df.columns):
        print("column name and column length validation passed")
        return 1
    else:
        print("column name and column length validation failed")
        mismatched_columns_file = list(set(df.columns).difference(expected_col))
        print("Following File columns are not in the YAML file",mismatched_columns_file)
        missing_YAML_file = list(set(expected_col).difference(df.columns))
        print("Following YAML columns are not in the file uploaded",missing_YAML_file)
        logging.info(f'df columns: {df.columns}')
        logging.info(f'expected columns: {expected_col}')
        return 0
```

```
Overwriting utility.py
```

```
%%writefile utility.py
import yaml
import logging
import re

def read_cfg(path):
    with open(path, 'r') as stream:
        try:
            return yaml.safe_load(stream)
        except yaml.YAMLError as exc:
            logging.error(exc)

def replacer(string, char):
    pattern = char + '{2,}'
    string = re.sub(pattern, char, string)
    return string

def col_header_val(df, table_cfg):
    df.columns = df.columns.str.lower()
    df.columns = df.columns.str.replace('[^\w]', '_', regex=True)
    df.columns = list(map(lambda x: x.strip('_'), list(df.columns)))
    df.columns = list(map(lambda x: replacer(x, '_'), list(df.columns)))
    expected_col = list(map(lambda x: x.lower(), table_cfg["columns"]))
    expected_col.sort()
    #df.columns = list(map(lambda x: x.lower(), list(df.columns)))
    df = df.reindex(sorted(df.columns), axis=1)
    if list(expected_col) == list(df.columns):
        print("column name validation passed")
        return 1
    else:
        print("column name validation failed")
        mismatch = list(set(df.columns).difference(expected_col))
        print("Columns not in YAML file: ", mismatch)
        missing =  list(set(expected_col).difference(df.columns))
        print("Columns not in data file: ", missing)
        return 0
```

```
Overwriting utility.py
```

```
%%writefile language.yaml
file_type: csv
dataset_name: file
file_name: birds.csv
inbound_delimiter: ","
outbound_delimiter: "|"
skip_leading_rows: 1
columns:
      - class
      - index
      - filepath
      - dataset
      - labels
```

```
Writing language.yaml
```

**Step7:**
Reading the config file

```
[ ]  # Reading config file
     import utility as util
     config_data = util.read_cfg("language.yaml")
```

```
[ ]  #data of config file
     config_data
```

```
{'file_type': 'csv',
 'dataset_name': 'file',
 'file_name': 'birds.csv',
 'inbound_delimiter': ',',
 'outbound_delimiter': '|',
 'skip_leading_rows': 1,
 'columns': ['class', 'index', 'filepath', 'dataset', 'labels']}
```

```
[ ]  # Reading process of the file using Dask
     df2_sample = dd.read_csv('birds.csv',delimiter=',')
     df2_sample.head(6)
```

| | class index | filepaths | labels | data set |
|---|---|---|---|---|
| **0** | 0 | train/ABBOTTS BABBLER/001.jpg | ABBOTTS BABBLER | train |
| **1** | 0 | train/ABBOTTS BABBLER/002.jpg | ABBOTTS BABBLER | train |
| **2** | 0 | train/ABBOTTS BABBLER/003.jpg | ABBOTTS BABBLER | train |
| **3** | 0 | train/ABBOTTS BABBLER/004.jpg | ABBOTTS BABBLER | train |
| **4** | 0 | train/ABBOTTS BABBLER/005.jpg | ABBOTTS BABBLER | train |
| **5** | 0 | train/ABBOTTS BABBLER/006.jpg | ABBOTTS BABBLER | train |

+ Code + Text

**Step8:**

```
#Reading the file using config file
file_type = config_data['file_type']
source_file = "./" + config_data['file_name'] + f'.{file_type}'
```

```
df2 = pd.read_csv(source_file,config_data ['inbound_delimiter'])
```

```
config_data
```

```
{'file_type': 'csv',
 'dataset_name': 'file',
 'file_name': 'birds.csv',
 'inbound_delimiter': ',',
 'outbound_delimiter': '|',
 'skip_leading_rows': 1,
 'columns': ['class', 'index', 'filepath', 'dataset', 'labels']}
```

```
#Output gz file
df2.to_csv('birds.csv.gz',
           sep='|',
           header=True,
           index=False,
           quoting=csv.QUOTE_ALL,
           compression='gzip',
           quotechar='"',
           doublequote=True,
           line_terminator='\n')
```

```
['/content/birds.csv.gz/0.part']
```

**Step9:**

Validated the data to obtain the following results

```
[ ]  #No. of Rows
     len(df.index)
```

```
     62388
```

```
[ ]  #Size of the file
     os.path.getsize('birds.csv')
```

```
     3442804
```

```
[ ]  #No, of Columns
     len(df2.columns)
```

```
     4
```

```
[ ]  #Size of the file
     os.path.getsize('birds.csv.gz')
```

```
     4096
```