

1_generate_prediction

October 3, 2021

```
[1]: download_metadata = True # will create a list of all csv files in the s3 bucket
      cache_http_calls = True # TTL for 1 hour
```

```
[ ]: # !pip install python-dotenv
```

```
[2]: import pyspark
      import requests
      import os
      import pandas as pd
      import boto3
      import json
      import cachetools

      from botocore import UNSIGNED
      from botocore.config import Config

      from pyspark.sql.session import SparkSession
      from pyspark.sql.types import *
      from pyspark.sql.functions import *
      from pyspark.sql import Row
      from pyspark.conf import SparkConf

      from copy import deepcopy
      from datetime import datetime, timedelta
      from dotenv import load_dotenv

      load_dotenv()

      os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages "org.apache.hadoop:hadoop-aws:3.
      ↪2.0" pyspark-shell'

      from IPython.core.display import HTML
      display(HTML("<style>pre { white-space: pre !important; }</style>"))

      # !pip install boto3
```

```
# !pip install cachetools

sparkConf = SparkConf()
sparkConf.set("spark.hadoop.fs.s3a.aws.credentials.provider", "org.apache.
↳hadoop.fs.s3a.AnonymousAWSCredentialsProvider")
sparkConf.set("spark.hadoop.fs.s3a.threads.max", 10)
sparkConf.set("spark.hadoop.fs.s3a.endpoint", "s3.amazonaws.com")

sc = pyspark.SparkContext("local[*]", conf = sparkConf, appName = "
↳s_p_challenge")
spark = SparkSession(sc)

print(f"spark version = {spark.version}")
print(f"pyspark version = {pyspark.__version__}")
print(f"Hadoop version = {sc._jvm.org.apache.hadoop.util.VersionInfo.
↳getVersion()}")
```

<IPython.core.display.HTML object>

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform
(file:/usr/local/spark-3.1.2-bin-hadoop3.2/jars/spark-unsafe_2.12-3.1.2.jar) to
constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of
org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal
reflective access operations
WARNING: All illegal access operations will be denied in a future release
Ivy Default Cache set to: /home/jovyan/.ivy2/cache
The jars for the packages stored in: /home/jovyan/.ivy2/jars
org.apache.hadoop#hadoop-aws added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-
parent-0f9963ff-5e5c-4bbc-a922-e17bf04848be;1.0
   confs: [default]
```

```
:: loading settings :: url = jar:file:/usr/local/spark-3.1.2-bin-
hadoop3.2/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
```

```
   found org.apache.hadoop#hadoop-aws;3.2.0 in central
   found com.amazonaws#aws-java-sdk-bundle;1.11.375 in central
:: resolution report :: resolve 137ms :: artifacts dl 4ms
   :: modules in use:
   com.amazonaws#aws-java-sdk-bundle;1.11.375 from central in [default]
   org.apache.hadoop#hadoop-aws;3.2.0 from central in [default]
```

		modules					artifacts	
conf	number	search	downlded	evicted			number	downlded
default	2	0	0	0			2	0

```

-----
:: retrieving :: org.apache.spark#spark-submit-
parent-0f9963ff-5e5c-4bbc-a922-e17bf04848be
  confs: [default]
    0 artifacts copied, 2 already retrieved (0kB/4ms)
21/10/03 14:59:28 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).

spark version = 3.1.2
pyspark version = 3.1.2
Hadoop version = 3.2.0

```

0.0.1 Download data and metadata from gdelt-open-data S3 bucket

```

[3]: # v2 event headers: https://github.com/linwoodc3/gdelt2HeaderRows/blob/master/
      ↪ schema_csvs/GDELT_2.0_Events_Column_Labels_Header_Row_Sep2016.csv
headers = pd.read_csv('headers.csv')
headers.head(n=2)

```

```

[3]:      tableId dataType      Empty \
0  GLOBALEVENTID  INTEGER  NULLABLE
1      SQLDATE  INTEGER  NULLABLE

                                     Description
0  Globally unique identifier assigned to each ev...
1  Date the event took place in YYYYMMDD format. ...

```

```

[4]: def download_metadata():
      """
      Download metadata from https://s3.console.aws.amazon.com/s3/buckets/
      ↪ gdelt-open-data?region=us-east-1
      """
      s3 = boto3.client('s3', config=Config(signature_version=UNSIGNED),
      ↪ region_name='us-east-1')
      s3_events = s3.list_objects_v2(Bucket='gdelt-open-data', Prefix='v2/events/
      ↪ ')
      s3_all_events = []

      is_truncated = True
      continuation_token = None

      while is_truncated:
          if continuation_token:

```

```

        s3_events = s3_events = s3.
    ↪list_objects_v2(Bucket='gdelt-open-data', Prefix='v2/events/',
    ↪ContinuationToken=continuation_token)
        else:
            s3_events = s3_events = s3.
    ↪list_objects_v2(Bucket='gdelt-open-data', Prefix='v2/events/')
            s3_all_events.append(s3_events)
            is_truncated = s3_events['IsTruncated']
            if 'NextContinuationToken' in s3_events:
                continuation_token = s3_events['NextContinuationToken']
            print('Total number of iterations to the S3 list objects = {:,}'.
    ↪format(len(s3_all_events)))
            s3_actual_events = []
            for s3_events in s3_all_events:
                s3_actual_events.extend(s3_events['Contents'])
            print('Total number of files in the S3 bucket = {:,}'.
    ↪format(len(s3_actual_events)))
            return s3_actual_events

```

```

[5]: if download_metadata:
    events_metadata = pd.DataFrame(download_metadata())
    events_metadata.sort_values(by='LastModified', inplace=True,
    ↪ascending=False)
    events_metadata

```

Total number of iterations to the S3 list objects = 144
 Total number of files in the S3 bucket = 143,462

```

[6]: # Example uses GDELT dataset found here: https://aws.amazon.com/public-datasets/
    ↪gdelt/
events = spark.read.csv("s3a://gdelt-open-data/v2/events/20190416151500.export.
    ↪csv", header=False, sep='\t', inferSchema=True)
print(f"Total number of events in current file: {events.count()}")

```

21/10/03 15:01:28 WARN MetricsConfig: Cannot locate configuration: tried hadoop-
 metrics2-s3a-file-system.properties,hadoop-metrics2.properties

Total number of events in current file: 1772

```

[7]: assert len(events.columns) == len(headers['tableId'])
    for idx in range(len(events.columns)):
        events = events.withColumnRenamed(f"_c{idx}", list(headers['tableId'])[idx])
    events = events.withColumn("SQLDATE", to_date(col("SQLDATE").cast("string"),
    ↪"yyyyMMdd"))
    events.printSchema()
    events.show(n=2)

```

root

```

|-- GLOBALEVENTID: integer (nullable = true)
|-- SQLDATE: date (nullable = true)
|-- MonthYear: integer (nullable = true)
|-- Year: integer (nullable = true)
|-- FractionDate: double (nullable = true)
|-- Actor1Code: string (nullable = true)
|-- Actor1Name: string (nullable = true)
|-- Actor1CountryCode: string (nullable = true)
|-- Actor1KnownGroupCode: string (nullable = true)
|-- Actor1EthnicCode: string (nullable = true)
|-- Actor1Religion1Code: string (nullable = true)
|-- Actor1Religion2Code: string (nullable = true)
|-- Actor1Type1Code: string (nullable = true)
|-- Actor1Type2Code: string (nullable = true)
|-- Actor1Type3Code: string (nullable = true)
|-- Actor2Code: string (nullable = true)
|-- Actor2Name: string (nullable = true)
|-- Actor2CountryCode: string (nullable = true)
|-- Actor2KnownGroupCode: string (nullable = true)
|-- Actor2EthnicCode: string (nullable = true)
|-- Actor2Religion1Code: string (nullable = true)
|-- Actor2Religion2Code: string (nullable = true)
|-- Actor2Type1Code: string (nullable = true)
|-- Actor2Type2Code: string (nullable = true)
|-- Actor2Type3Code: string (nullable = true)
|-- IsRootEvent: integer (nullable = true)
|-- EventCode: integer (nullable = true)
|-- EventBaseCode: integer (nullable = true)
|-- EventRootCode: integer (nullable = true)
|-- QuadClass: integer (nullable = true)
|-- GoldsteinScale: double (nullable = true)
|-- NumMentions: integer (nullable = true)
|-- NumSources: integer (nullable = true)
|-- NumArticles: integer (nullable = true)
|-- AvgTone: double (nullable = true)
|-- Actor1Geo_Type: integer (nullable = true)
|-- Actor1Geo_FullName: string (nullable = true)
|-- Actor1Geo_CountryCode: string (nullable = true)
|-- Actor1Geo_ADM1Code: string (nullable = true)
|-- Actor1Geo_ADM2Code: string (nullable = true)
|-- Actor1Geo_Lat: double (nullable = true)
|-- Actor1Geo_Long: double (nullable = true)
|-- Actor1Geo_FeatureID: string (nullable = true)
|-- Actor2Geo_Type: integer (nullable = true)
|-- Actor2Geo_FullName: string (nullable = true)
|-- Actor2Geo_CountryCode: string (nullable = true)
|-- Actor2Geo_ADM1Code: string (nullable = true)
|-- Actor2Geo_ADM2Code: string (nullable = true)

```

```

|-- Actor2Geo_Lat: double (nullable = true)
|-- Actor2Geo_Long: double (nullable = true)
|-- Actor2Geo_FeatureID: string (nullable = true)
|-- ActionGeo_Type: integer (nullable = true)
|-- ActionGeo_FullName: string (nullable = true)
|-- ActionGeo_CountryCode: string (nullable = true)
|-- ActionGeo_ADM1Code: string (nullable = true)
|-- ActionGeo_ADM2Code: string (nullable = true)
|-- ActionGeo_Lat: double (nullable = true)
|-- ActionGeo_Long: double (nullable = true)
|-- ActionGeo_FeatureID: string (nullable = true)
|-- DATEADDED: long (nullable = true)
|-- SOURCEURL: string (nullable = true)

```

21/10/03 15:01:41 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

```

+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
|GLOBALEVENTID|    SQLDATE|MonthYear|Year|FractionDate|Actor1Code|Actor1Name|Acto
r1CountryCode|Actor1KnownGroupCode|Actor1EthnicCode|Actor1Religion1Code|Actor1Re
ligion2Code|Actor1Type1Code|Actor1Type2Code|Actor1Type3Code|Actor2Code|Actor2Nam
e|Actor2CountryCode|Actor2KnownGroupCode|Actor2EthnicCode|Actor2Religion1Code|Ac
tor2Religion2Code|Actor2Type1Code|Actor2Type2Code|Actor2Type3Code|IsRootEvent|Ev
entCode|EventBaseCode|EventRootCode|QuadClass|GoldsteinScale|NumMentions|NumSour
ces|NumArticles|          AvgTone|Actor1Geo_Type|Actor1Geo_FullName|Actor1Geo_Cou
ntryCode|Actor1Geo_ADM1Code|Actor1Geo_ADM2Code|Actor1Geo_Lat|Actor1Geo_Long|Acto
r1Geo_FeatureID|Actor2Geo_Type|Actor2Geo_FullName|Actor2Geo_CountryCode|Actor2Ge
o_ADM1Code|Actor2Geo_ADM2Code|Actor2Geo_Lat|Actor2Geo_Long|Actor2Geo_FeatureID|A
ctionGeo_Type|ActionGeo_FullName|ActionGeo_CountryCode|ActionGeo_ADM1Code|Action
Geo_ADM2Code|ActionGeo_Lat|ActionGeo_Long|ActionGeo_FeatureID|    DATEADDED|
SOURCEURL|
+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+-----+-----+-----+

```



```

    call_cache = call_cache = cachetools.TTLCache(10000,
→ttl=timedelta(hours=1), timer=datetime.now)

def get_model_response(payload: dict[str, object], event_id=None):
    '''
    Sample payload:
    payload = {
        "data": {
            "avg_tone": -2,
            "goldstein": 0.5,
            "actor_code": "GOV",
            "lat": 38,
            "lon": -78,
            "date": "2018-10-23 04:30:00"
        }
    }

    TODO: Need to check if we need to implement politeness while calling the API
    '''
    if event_id and event_id in call_cache:
        return call_cache['event_id']

    headers = {'Content-Type': 'application/json'}
    username = password = os.getenv('API_PASSWORD')

    res = requests.post(
        url=os.getenv('API_URL'),
        headers = headers,
        data = json.dumps(payload),
        auth=(username, password)
    )
    response = json.loads(res.content.decode('utf-8'))
    call_cache[event_id] = response
    return response

def flatten_model_response(actor: str, response: dict[str,object],
→event_id=None, debug=False):
    d = {}
    if event_id and debug:
        print(event_id)
    try:
        d[f'{actor}_model_time_in_ms'] = response['model_time_in_ms']
        d[f'{actor}_release_harness_version'] =
→response['release']['harness_version']
        d[f'{actor}_release_model_version'] =
→response['release']['model_version']
        d[f'{actor}_release_model_version_number'] =
→response['release']['model_version_number']

```

```

    d[f'{actor}_request_id'] = response['request_id']
    d[f'{actor}_result_class1'] = response['result']['class1']
    d[f'{actor}_result_class2'] = response['result']['class2']
    d[f'{actor}_timing'] = response['timing']
except Exception as e:
    print(response)
return d

```

```

[10]: def create_payload(avg_tone, goldstein, actor_code, lat, lon, date):
    data = {}
    data['avg_tone'] = avg_tone
    data['goldstein'] = goldstein
    data['actor_code'] = actor_code
    data['lat'] = lat
    data['lon'] = lon
    data['date'] = date.strftime('%Y-%m-%d %H:%M:%S')
    payload = {}
    payload['data'] = data
    return payload

def call_model_output(row):
    """
    payload = {
        "data": {
            "avg_tone": -2,
            "goldstein": 0.5,
            "actor_code": "GOV",
            "lat": 38,
            "lon": -78,
            "date": "2018-10-23 04:30:00"
        }
    }
    """
    # actor 1
    r = row.asDict(True)
    payload = create_payload(row['AvgTone'], row['GoldsteinScale'],
    ↪row['Actor1Code'], row['Actor1Geo_Lat'], row['Actor1Geo_Long'], datetime.
    ↪strftime(str(row['DATEADDED']), '%Y%m%d%H%M%S'))
    response = flatten_model_response('Actor1', get_model_response(payload),
    ↪event_id=row['GLOBALEVENTID'])

    for k, v in response.items():
        r[k] = v

    # actor 2

```

```

    payload = create_payload(row['AvgTone'], row['GoldsteinScale'],
    ↳row['Actor2Code'], row['Actor2Geo_Lat'], row['Actor2Geo_Long'], datetime.
    ↳strptime(str(row['DATEADDED']), '%Y%m%d%H%M%S'))
    response = flatten_model_response('Actor2', get_model_response(payload),
    ↳event_id=row['GLOBALEVENTID'])

    for k, v in response.items():
        r[k] = v

    return Row(**r)

def define_schema(events):
    schema = deepcopy(events.schema)
    print('Number of columns in schema before addition = {:,}'.
    ↳format(len(schema)))
    # https://spark.apache.org/docs/latest/sql-ref-datatypes.html
    for actor in ['Actor1', 'Actor2']:
        schema.add(StructField(f'{actor}__model_time_in_ms', IntegerType(),
    ↳True))
        schema.add(StructField(f'{actor}_release_harness_version',
    ↳StringType(), True))
        schema.add(StructField(f'{actor}_release_model_version', StringType(),
    ↳True))
        schema.add(StructField(f'{actor}_release_model_version_number',
    ↳IntegerType(), True))
        schema.add(StructField(f'{actor}_request_id', StringType(), True))
        schema.add(StructField(f'{actor}_result_class1', BooleanType(), True))
        schema.add(StructField(f'{actor}_result_class2', IntegerType(), True))
        schema.add(StructField(f'{actor}_timing', DoubleType(), True))
    print('Number of columns in schema after addition = {:,}'.
    ↳format(len(schema)))
    return schema

```

```

[11]: df = events_clean.rdd.map(call_model_output)
      schema = define_schema(events_clean)
      df = spark.createDataFrame(df, schema)
      df.show(n=1, vertical=True)
      df.write.parquet('model_output.parquet')

```

Number of columns in schema before addition = 61
 Number of columns in schema after addition = 77

```

-RECORD 0-----
GLOBALEVENTID          | 838788881
SQLDATE                | 2018-04-16
MonthYear              | 201804

```

Year	2018
FractionDate	2018.2904
Actor1Code	EDU
Actor1Name	ECONOMIST
Actor1CountryCode	null
Actor1KnownGroupCode	null
Actor1EthnicCode	null
Actor1Religion1Code	null
Actor1Religion2Code	null
Actor1Type1Code	EDU
Actor1Type2Code	null
Actor1Type3Code	null
Actor2Code	GOV
Actor2Name	REGULATOR
Actor2CountryCode	null
Actor2KnownGroupCode	null
Actor2EthnicCode	null
Actor2Religion1Code	null
Actor2Religion2Code	null
Actor2Type1Code	GOV
Actor2Type2Code	null
Actor2Type3Code	null
IsRootEvent	1
EventCode	20
EventBaseCode	20
EventRootCode	2
QuadClass	1
GoldsteinScale	3.0
NumMentions	10
NumSources	1
NumArticles	10
AvgTone	-3.15315315315315
Actor1Geo_Type	4
Actor1Geo_FullName	Vancouver, Britis...
Actor1Geo_CountryCode	CA
Actor1Geo_ADM1Code	CA02
Actor1Geo_ADM2Code	12552
Actor1Geo_Lat	49.25
Actor1Geo_Long	-123.133
Actor1Geo_FeatureID	-575268
Actor2Geo_Type	4
Actor2Geo_FullName	Vancouver, Britis...
Actor2Geo_CountryCode	CA
Actor2Geo_ADM1Code	CA02
Actor2Geo_ADM2Code	12552
Actor2Geo_Lat	49.25
Actor2Geo_Long	-123.133
Actor2Geo_FeatureID	-575268

ActionGeo_Type	4
ActionGeo_FullName	Vancouver, Britis...
ActionGeo_CountryCode	CA
ActionGeo_ADM1Code	CA02
ActionGeo_ADM2Code	12552
ActionGeo_Lat	49.25
ActionGeo_Long	-123.133
ActionGeo_FeatureID	-575268
DATEADDED	20190416151500
SOURCEURL	https://www.bnnbl...
Actor1__model_time_in_ms	1000
Actor1_release_harness_version	0.1
Actor1_release_model_version	5ec427ae4cedfd000...
Actor1_release_model_version_number	4
Actor1_request_id	KTF1CHOMAY9Q74SR
Actor1_result_class1	true
Actor1_result_class2	3
Actor1_timing	1000.3688335418701
Actor2__model_time_in_ms	0
Actor2_release_harness_version	0.1
Actor2_release_model_version	5ec427ae4cedfd000...
Actor2_release_model_version_number	4
Actor2_request_id	RGW2IH58DNBT4AFS
Actor2_result_class1	true
Actor2_result_class2	3
Actor2_timing	0.07510185241699219

only showing top 1 row

```

-----
AnalysisException                                Traceback (most recent call last)
/tmp/ipykernel_2919/2354079784.py in <module>
      3 df = spark.createDataFrame(df, schema)
      4 df.show(n=1, vertical=True)
----> 5 df.write.parquet('model_output.parquet')

/usr/local/spark/python/pyspark/sql/readwriter.py in parquet(self, path, mode,
↳ partitionBy, compression)
    1248         self.partitionBy(partitionBy)
    1249         self._set_opts(compression=compression)
-> 1250         self._jwrite.parquet(path)
    1251
    1252     def text(self, path, compression=None, lineSep=None):

/usr/local/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in
↳ __call__(self, *args)
    1302

```

```

1303         answer = self.gateway_client.send_command(command)
-> 1304         return_value = get_return_value(
1305             answer, self.gateway_client, self.target_id, self.name)
1306
/usr/local/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
115             # Hide where the exception came from that shows a
↳non-Pythonic
116             # JVM exception message.
--> 117             raise converted from None
118         else:
119             raise

AnalysisException: path file:/home/jovyan/work/model_output.parquet already
↳exists.

```

```

[ ]: df = spark.read.parquet('model_output.parquet')
dfp = df.toPandas()
dfp.to_csv('model_output.csv')

```