Coventry University | FACULTY OF ENGINEERING, ENVIRONMENT & COMPUTING

(7082CEM)

Coursework Demonstration of a Big Data Program

MODULE LEADER: Dr. Marwan Fuad

Student Name: Vamsidhar Reddy Muchurami

SID: 10864627

BIG DATA ANALYSIS THROUGH MACHINE LEARNING USING PYSPARK

I can confirm that all work submitted is my own: Yes

Table of Contents

INTRODUCTION

In all sectors of the globe, information technology is prevalent and pervasive. Machine learning is essential in data science; they are all fundamental subfields of artificial intelligence that will play a significant role in the future. By utilising greater volumes of data, machine learning assists in the learning and development of computer programmes and performance. Machine learning's major objective is to build better algorithms without the aid of humans.

Big data is also an area of artificial intelligence. This project involves the display of massive amounts of data, machine learning modelling, and visualisation. This project's demonstration has the potential to create a better result, and the project covers software installation, configuration, and operation. Big data is a term for a big data set with exceptionally large instances, with a focus on volume, diversity, and velocity. It conducts research to identify trends, correlations, and patterns. The analysis of such vast volumes of data takes time; however, this may be minimised by utilising current software systems like Hadoop. There are three forms of data analytics: descriptive, predictive, and prescriptive.

A Google paper based on Hadoop was issued in 2004, and work began in 2005. In 2008, Hadoop was the most popular and high-level [platform used by Yahoo, The New York Times, and Facebook. It is a free and open-source platform for running and developing distributed applications that manage enormous quantities of data. A cluster of computers is used to implement data splitting, which is subsequently separated into racks. Hadoop utilises schema-on-read and has a fault-tolerant design.

The project's objective is for Pyspark and Jupyter Notebook to construct a machine study model and view data (Anaconda). Instances, objects, information or other presentations are used for data visualisation. Visualization allows a user to discover information patterns, trends, and linkages quickly. Visualization is used to analyse, test and present information. Data collecting, data augmentation, mapping and distribution include four types of viewing processes. Data acquisition makes it possible to generate or gather data, which is then processed during the data growth stage. Primitives are assigned to the visualisation.

This research focuses on forecasting UFO sightings based on certain characteristics. The data set is in CSV format and was obtained via Kaggle. CSV is a semi-structured data format having recognisable patterns that may be processed. Several scenarios will be used to calculate the price. Data visualisation may also be used to show prediction.

Any aerial object or visual phenomena that cannot be identified easily by the viewer, commonly known as a flying saucer (UFO). UFOs became a big topic after the invention of rocketry after World War II and were considered to represent intelligent alien species visiting the planet earth by some experts.

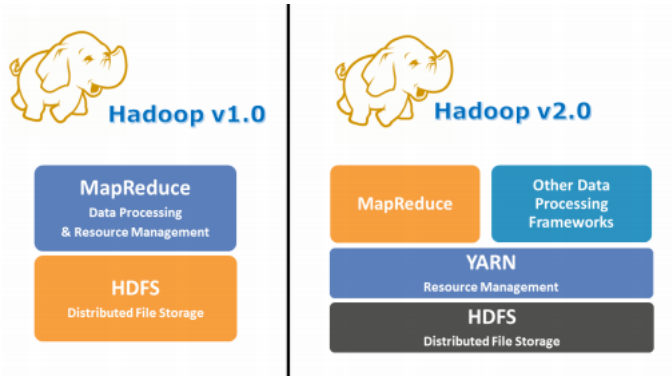
Sightings of unidentified airborne phenomena increased, and in 1948, the United States Air Force began Project Sign, an investigation into these claims. The initial agreement among those involved in the investigation was that the UFOs were most likely advanced Soviet aircraft, while some specialists suggested that they may be spacecraft from other planets, an idea known as the extra-terrestrial hypothesis (ETH). Within a year, Project Sign was followed by Project Grudge, which was followed in 1952 by Project Blue Book, the longest-running official UFO investigation, centred at Wright-Patterson Air Force Base in Dayton, Ohio. From 1952 through 1969, Project Blue Book collected reports of over 12,000 sightings or events, which were then classified as either “identified” with a known astronomical, atmospheric, or artificial (human-caused) phenomenon or “unidentified.” The last category, which accounted for around 6% of the total, included cases in which there was insufficient information to make a match with a known event.

IMPLEMENTATION

BACKGROUND STUDY

1. APACHE HADOOP

Hadoop is big data analytics open-source framework. It incorporates a system of fault tolerances and allows enormous amounts of data to be stored. HDFS, MapReduce and YARN are the three primary components contained in Hadoop.



HDFS: Hadoop Distributed File System was created from the Google File System and is responsible for storing (GFS). Save millions of big files. Two nodes with the name node and data node in the HDFS cluster. HDFS is a master-slave architecture where the master is the node and the slave is the node of the data. HDFS has two major operations, including reading and writing.

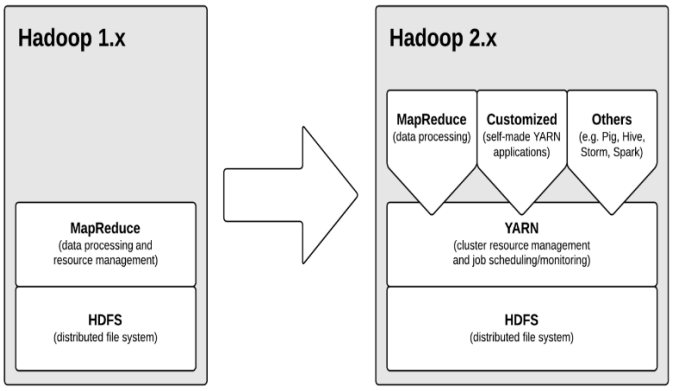


Fig: Hadoop components

MapReduce: MapReduce is a processing feature, employed in distributed programming paradigm, that generates implemented large data in a simultaneous manner across servers in clusters. Mapping and reducing are the key responsibilities. A dataset is converted into other datasets and during the map operation, individual components are divided into key / value pairs. Three stages of MapReduce, map stage, scenery and lastly phase reduction may be passed. The Map Stage processes the data and creates smaller chunks from the HDFS line-by-line input files. Reduce phase is the combination of reduction and shuffling phase, processing the map data and producing new HDFS outputs and stores.

YARN: Yet another resource negotiator is a resource management and job planning procedure. YARN: In Hadoop, YARN supports many sorts of processing. YARN is the intermediate layer which enables thousands of jobs to be divided into work. YARN's components include three major components: resource manager, node manager and app manager. The Resource Manager supports and allocates resources for various applications and has several unique functions: resource management, scheduling,

application management, Hadoop resource containers and containers. Node management supports memory management, assists tracking and memory state, CPU and network on local servers and reports this information Parallel execution of the job is done by application management.

2. APACHE SPARK

Industries use Hadoop to examine their data sets extensively. The rationale is that the Hadoop framework is built on a basic (MapReduce) programming style and provides a scalable, adaptable, tolerant and economical computing solution. The major purpose here is to maintain the speed with which huge datasets are processed in relation to waiting between queries and waiting to run the programme.

Spark was introduced to speed up the Hadoop computer software process by Apache Software Foundation.

3. RDD

RDD is responsible for structured, unstructured data collections, because of its ability to rebuild data in the event of node failures, RDD is strong. The RDD data may be automatically distributed amongst the clusters and can be executed in parallel. If it was built then the RDD cannot be amended as immutable. Each RDD data set may be split into several divisions with different cluster nodes to be computed. Two techniques may be used in RDD- parallel or referencing a dataset. RDD is used by Spark to enhance, speed and efficiency of MapReduce processes.

4. PYSPARK

The Scala language is the name of spark writing and the Python support for Apache Spark. The Py4j package is used for running RDD in python language. The Python API has a Pyspark core connection. In real-time, Pyspark is utilised frequently and especially in quicker processing.

5. PYTHON

The language of programming used by Python is largely for web development, system scripting, software development and mathematics. Python works on Windows, RaspberryPi, MacOS and so on. The syntax is simple and maybe run as written, which tells that prototyping is fast.

We use seaborn and matplotlib for doing visualizations to show different types of graphs.

6. DATA FRAME

Dataframe is a two-dimensional structure or table array with each column containing one variable and each row containing each column value. It is capable of working with many sources of data. Data frame allows you to create a schematic table to hold data.

7. ANACONDA AND JUPYTER NOTEBOOK

Anaconda provides real-world issues, machine learning applications, wide data processing and predictive analytics in python, and R programming languages. Jupyter Notebook is an online tool that offers the opportunity to create and share documents with live scripts, equations, views, etc. Especially used for data cleansing and processing.

SOFTWARE INSTALLATION

Installing an anaconda environment is a very easy task.

STEP 1: Setup Anaconda Environment

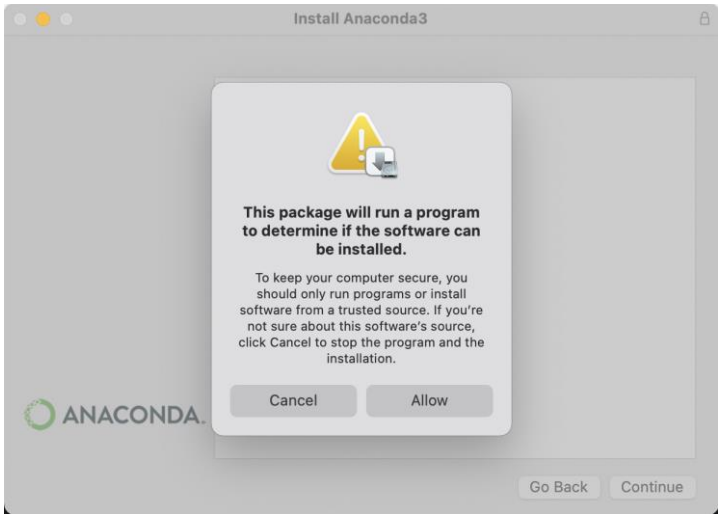
Download Python 3.8 • 64-Bit Graphical Installer • 440 MB from the website given below:

https://www.anaconda.com/products/individual

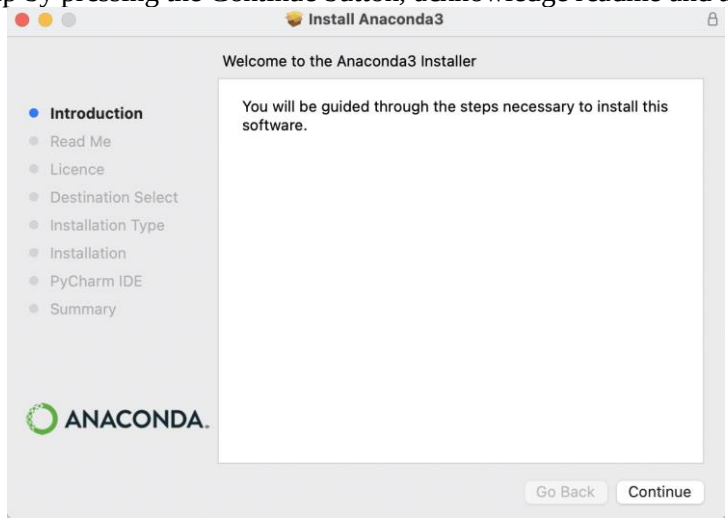
It will be downloaded to the folder in mac.

Browse the macOS desktop Finder and open the Download folder, double-tap the .pkg extension and the new window will start-up.

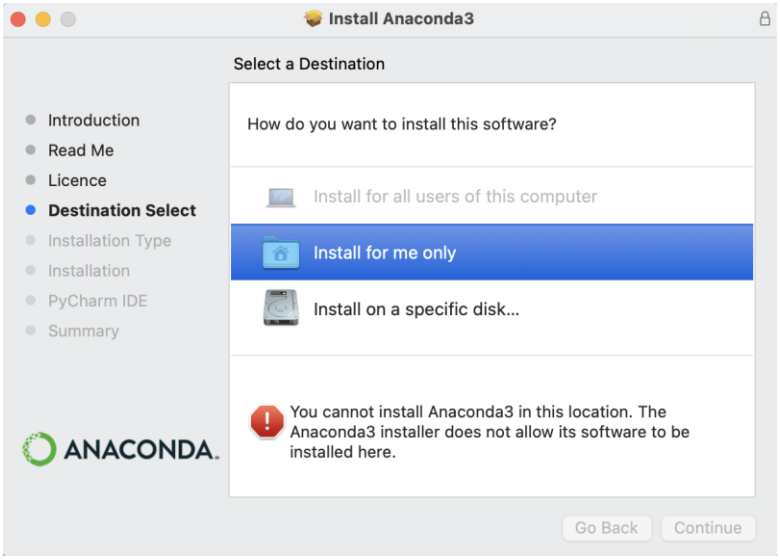
Click on allow.



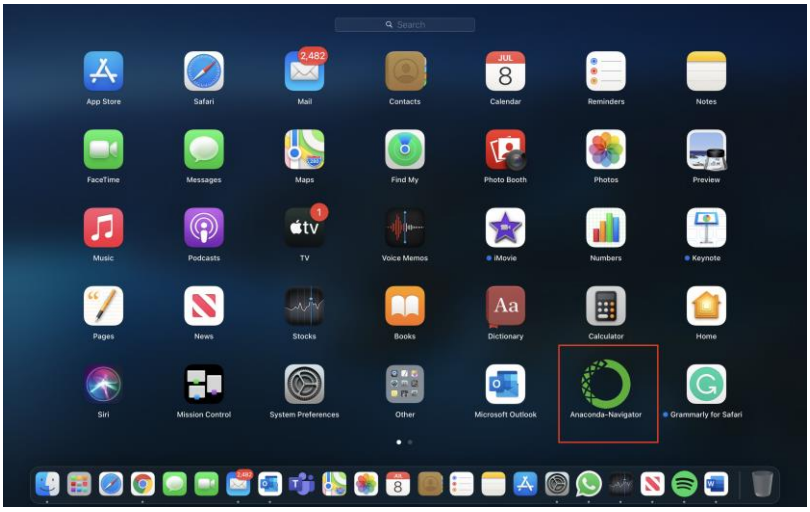
Continue the setup by pressing the Continue button, acknowledge readme and accept the license.



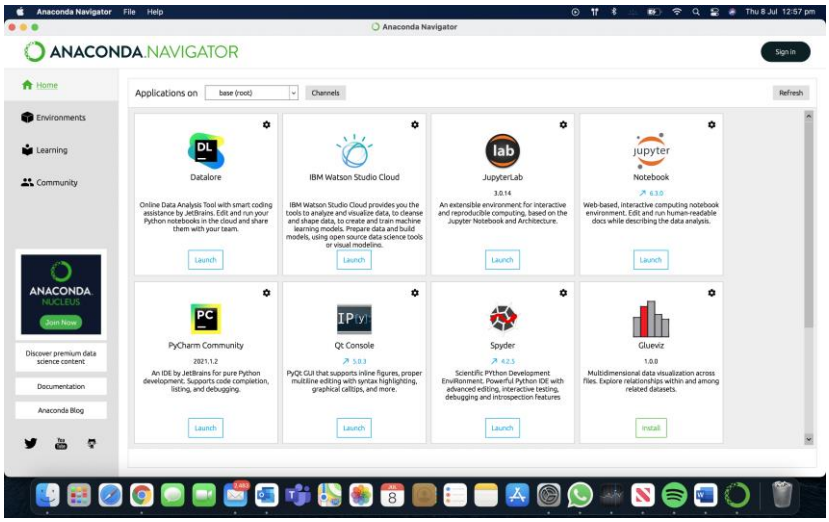
If you want to install in a specific disk change the destination path or continue with the install for me only.



Go to the launch pad after installation and identify and click on the anaconda navigator.



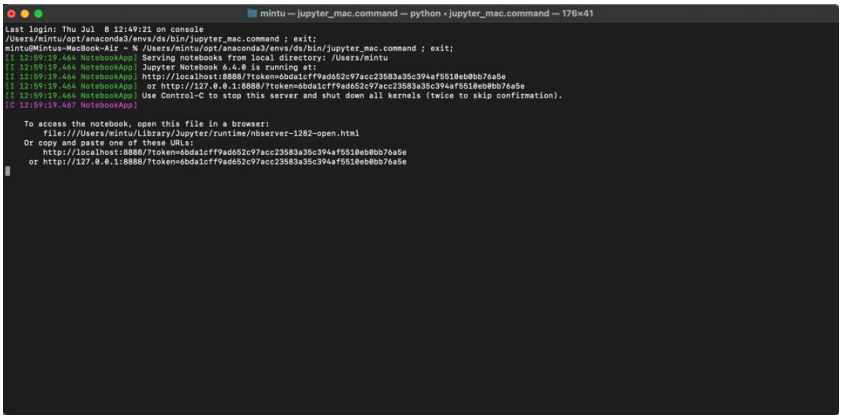
STEP 2: Launch Jupyter Notebook from Anaconda Navigator



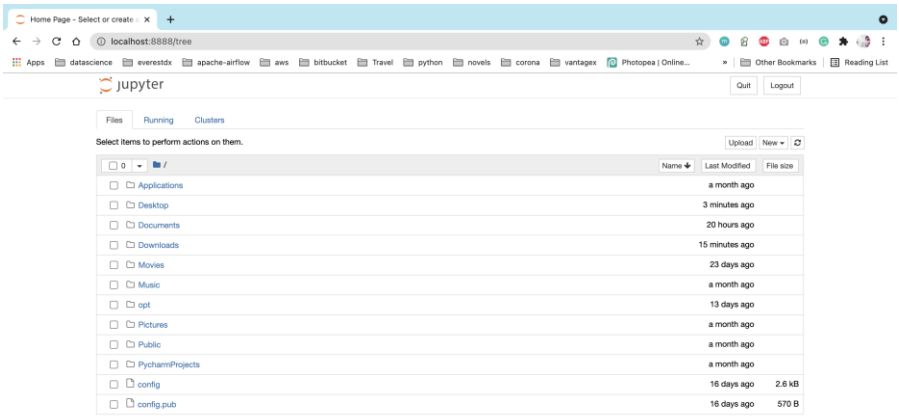
After start-up, a new terminal will open to execute jupyter-notebooks and the user interface will open.

To view ipynb notebooks, navigate to this page

http://localhost:8888/tree



Below is the UI of the Jupyter Notebook after launching it.



STEP 3: Install Pyspark

Open terminal in Jupyter Notebook

Download pyspark using the command: "**pip install pyspark**"

Pyspark will be downloaded.

```
(ds) mintu@Mintus-MacBook-Air ~ % pip install pyspark
Requirement already satisfied: pyspark in ./opt/anaconda3/envs/ds/lib/python3.8/site-packages (3.1.2)
Requirement already satisfied: py4j==0.10.9 in ./opt/anaconda3/envs/ds/lib/python3.8/site-packages (from pyspark) (0.10.9)
(ds) mintu@Mintus-MacBook-Air ~ %
```

Install all the packages you need using the same approach.

DATASET DESCRIPTION

Attributes Description

This article contains UFO encounters with specific characteristics such as date, town, State, nation, form, duration (seconds), duration (hours/min), remark, date uploaded, latitude, longitude. And there are numerous examples and few characteristics in this data set. The attributes indicating UFO sightings in the US are shown in Table 1.

S.No	Attribute	Data Types
1	datetime	String
2	city	String
3	state	String
4	Size	String
5	country	String
6	shape	String
7	duration (seconds)	String
8	duration (hours/min)	String
9	comments	String
10	date posted	String
11	latitude	String
12	longitude	String

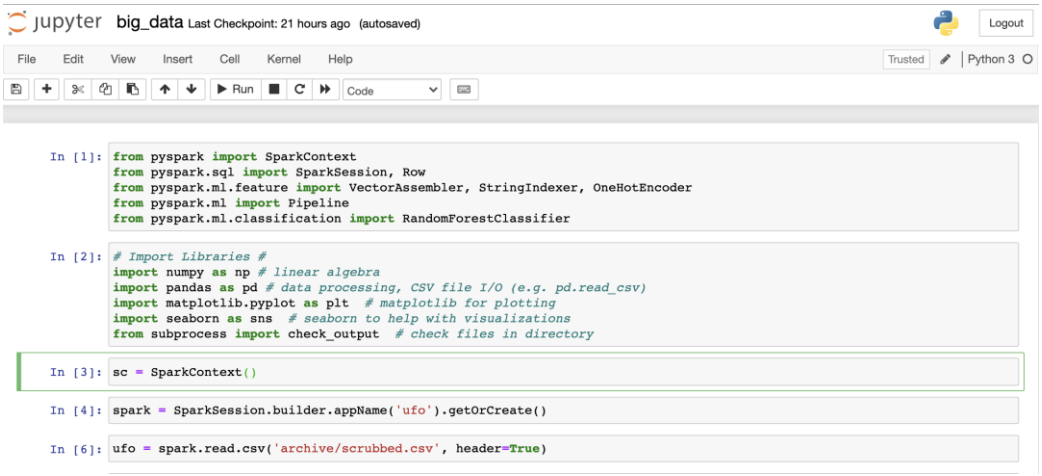
Table(1) : Dataset attributes and types

PREPROCESSING

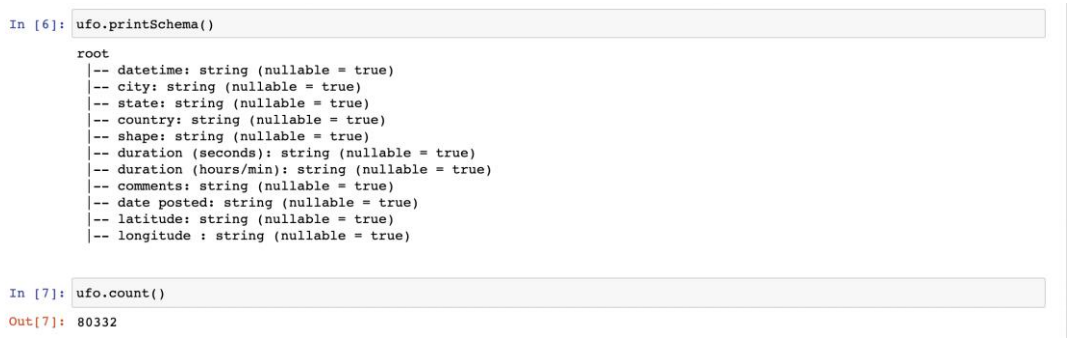
The new Jupyter Notebook for coding is opened. The libraries of Pyspark are first imported. Also called Spark, Dataset is loaded to a certain variable.

Import the functions that are required to read the file and create a new pyspark connection with the app name.

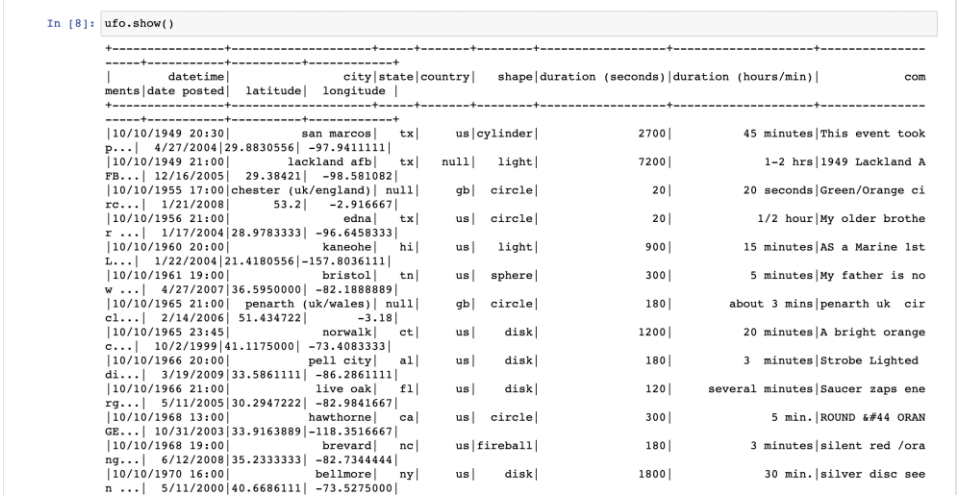
Use spark.read.csv('filename', header=True) to read csv file.



Using PrintSchema the attributes and the types are shown. Total occurrences with count are computed.



The sample data is shown in using ufo.show().



ufo.printschema() will show that all the attributes are in string format even the data inside is having date formats, location details and even integers.

```
In [7]: ufo.printSchema()

root
 |-- datetime: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- country: string (nullable = true)
 |-- shape: string (nullable = true)
 |-- duration (seconds): string (nullable = true)
 |-- duration (hours/min): string (nullable = true)
 |-- comments: string (nullable = true)
 |-- date posted: string (nullable = true)
 |-- latitude: string (nullable = true)
 |-- longitude : string (nullable = true)
```

The source data characteristics are all in string format. The dataframe must be updated with their appropriate data type using withcolumn and cast functions in pyspark.

convert data types

```
In [19]: from pyspark.sql.types import IntegerType
         list_cols = ['duration (seconds)', 'latitude', 'longitude']

In [20]: from pyspark.sql.functions import unix_timestamp, from_unixtime
         from pyspark.sql.types import TimestampType

         ufo = ufo.withColumn("duration (seconds)", ufo['duration (seconds)'].cast('float'))
         ufo = ufo.withColumn("latitude", ufo.latitude.cast('float'))
         ufo = ufo.withColumn("longitude ", ufo['longitude '].cast('float'))
         ufo = ufo.withColumn("'date posted' ", ufo['date posted'].cast('float'))
         ufo = ufo.withColumn("datetime",unix_timestamp("datetime", 'MM/dd/yyyy HH:mm').cast(TimestampType()))

In [ ]:

In [21]: ufo.printSchema()

root
 |-- datetime: datetime (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- country: string (nullable = true)
 |-- shape: string (nullable = true)
 |-- duration (seconds): float (nullable = true)
 |-- duration (hours/min): string (nullable = true)
 |-- comments: string (nullable = true)
 |-- date posted: float (nullable = true)
 |-- latitude: float (nullable = true)
 |-- longitude : float (nullable = true)
```

After completing the transformation, we can verify that the schema of the dataset has been converted to the appropriate data type. Datetime format should be used for the datetime attribute, and float format should be used for the remainder of the data.

Renaming columns

Because longitude is added to the dataset with white space, we are renaming the column name using withColumnRenamed from ‘longitude ’ to ‘longitude’. This will help in the better visualization of data.

replace column name

```
In [27]: from pyspark.sql.functions import when

         ufo = ufo.withColumnRenamed("longitude ","longitude")
```

Groupby Count

```
In [111]: for ufo_cols in ufo.columns:
              print(f"Colname: {ufo_cols}")
              ufo.groupby(ufo_cols).count().sort(desc('count')).show()
```

For calculating and counting all the attribute counts of similar values, groupby functions will be utilised. We grouped all the columns and obtain the following count.

Colname: datetime

datetime	count
7/4/2010 22:00	36
7/4/2012 22:00	31
11/16/1999 19:00	27
9/19/2009 20:00	26
7/4/2011 22:00	24
7/4/2010 21:00	23
10/31/2004 20:00	23
7/4/2013 22:00	22
7/4/2012 22:30	21
11/16/1999 19:05	20
11/16/1999 18:05	20
7/4/2010 23:00	20
7/4/2013 22:30	20
9/19/2009 19:50	19
7/4/2012 21:30	18
9/30/2005 23:00	18
7/4/2010 21:30	18
7/4/2010 22:30	18
9/1/1999 21:30	18
11/16/1999 19:10	16

Colname: city

city	count
seattle	525
phoenix	454
portland	374
las vegas	368
los angeles	353
san diego	338
houston	297
chicago	265
tucson	241
miami	239
orlando	220
austin	218
springfield	214
albuquerque	213
sacramento	201
columbus	200
san jose	189
san francisco	187
denver	185
san antonio	175

Colname: state

state	count
ca	9652
wa	4268
fl	4199
tx	3677
ny	3219
null	2935
az	2689
il	2645
pa	2582
oh	2425
mi	2071
nc	1868
or	1845
mo	1576
on	1560
nj	1512
co	1505
va	1392
in	1385
ma	1358

Colname: shape

shape	count
light	15883
triangle	7551
circle	7188
fireball	5988
other	5375
unknown	5371
sphere	5127
disk	5003
oval	3577
formation	2339
cigar	1953
changing	1883
null	1850
flash	1271
rectangle	1246
cylinder	1230
diamond	1107
chevron	927
egg	714
teardrop	693

Colname: duration (seconds)

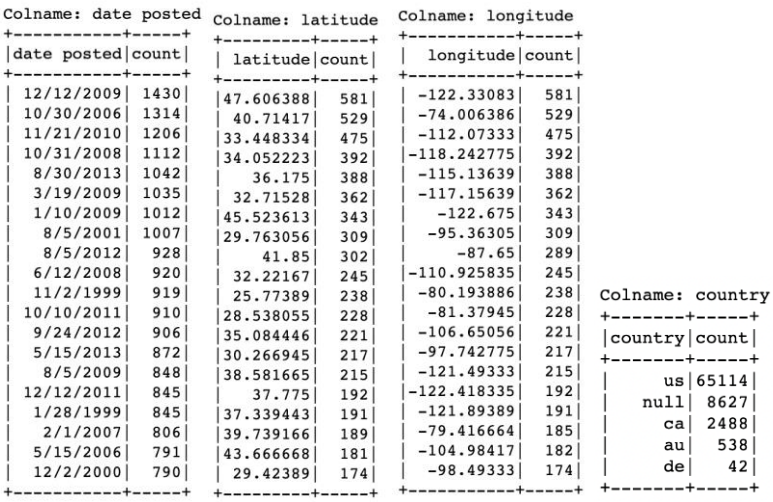
duration (seconds)	count
300.0	8296
120.0	6789
600.0	6043
60.0	5565
180.0	4843
900.0	3901
30.0	3508
10.0	3213
1200.0	2857
5.0	2808
1800.0	2657
3600.0	2113
2.0	2077
15.0	2048
240.0	1989
20.0	1842
3.0	1541
7200.0	1135
2700.0	1018
45.0	980

Colname: duration (hours/min)

duration (hours/min)	count
5 minutes	4574
2 minutes	3352
10 minutes	3222
1 minute	2927
3 minutes	2427
30 seconds	2205
15 minutes	2021
10 seconds	1906
5 seconds	1735
20 minutes	1403
30 minutes	1301
1 hour	1273
15 seconds	1140
20 seconds	1070
3 seconds	895
4 minutes	874
5 min	819
2 seconds	667
2 hours	619
10 min	617

Colname: comments

comments	count
null	15
Fireball	11
((NUFORC Note: N...	9
UFO	7
Lights in the sky	7
Bright Light	6
Lights in the sky.	6
lights in the sky	6
((NUFORC Note: W...	6
Orange ball of li...	6
Fireball.	5
Bright light	5
Green Fireball	4
bright light	4
3 lights in the sky	4
Bright light in t...	4
Fast moving light	3
Orange lights.	3
Fire in the sky.	3
I saw a UFO	3



Colname: date posted

date posted	count
12/12/2009	1430
10/30/2006	1314
11/21/2010	1206
10/31/2008	1112
8/30/2013	1042
3/19/2009	1035
1/10/2009	1012
8/5/2001	1007
8/5/2012	928
6/12/2008	920
11/2/1999	919
10/10/2011	910
9/24/2012	906
5/15/2013	872
8/5/2009	848
12/12/2011	845
1/28/1999	845
2/1/2007	806
5/15/2006	791
12/2/2000	790

Colname: latitude

latitude	count
47.606388	581
40.71417	529
33.448334	475
34.052223	392
36.175	388
32.71528	362
45.523613	343
29.763056	309
41.85	302
32.22167	245
25.77389	238
28.538055	228
35.084446	221
30.266945	217
38.581665	215
37.775	192
37.339443	191
39.739166	189
43.666668	181
29.42389	174

Colname: longitude

longitude	count
-122.33083	581
-74.006386	529
-112.07333	475
-118.242775	392
-115.13639	388
-117.15639	362
-122.675	343
-95.36305	309
-87.65	289
-110.925835	245
-80.193886	238
-81.37945	228
-106.65056	221
-97.742775	217
-121.49333	215
-122.418335	192
-121.89389	191
-79.416664	185
-104.98417	182
-98.49333	174

Colname: country

country	count
us	65114
null	8627
ca	2488
au	538
de	42

The figures above illustrate the set of columns in which sightings were given with the count, whereas the table below shows a table with a high count. This will aid us in better comprehending the data, such as which shape of ufo is observed in the sky and where it is most likely to occur.

S.no	Column name	Units	Count
1	datetime	7/4/2010	36
2	city	Seattle	525
3	state	Ca	9652
4	shape	Light	15883
5	duration (seconds)	300.0	8296
6	duration (hours/min)	5 minutes	4574
7	comments	null	15
8	date posted	12/12/2009	1430
9	latitude	47.6063889	581
10	longitude	-122.3308333	581
11	country	us	65114

Table(2) : Groupby columns

Filtering

Data filtering is the process of selecting a subset of your data collection for display or analysis. Filtering is usually transitory - the entire data set is maintained, but only a subset of it is utilised for computation. Using the filtering method we can analyze the data better from a different perspective.

The data will be filtered depending on latitude 50 and -125 longitude. This gives us the data with filter addition that shows above latitude 50 and -125 longitude.

```
In [29]: ufo = ufo[(ufo.latitude > 50) & (ufo.longitude > -125) == False]

In [30]: ufo.show()
```

```
+----------------+----------------+-----+-------+--------+------------------+--------------------+--------------------
-+-----------+---------+-----------+
|        datetime|            city|state|country|   shape|duration (seconds)|duration (hours/min)|             comment
s|date posted| latitude| longitude|
+----------------+----------------+-----+-------+--------+------------------+--------------------+--------------------
-+-----------+---------+-----------+
|10/10/1949 20:30|      san marcos|   tx|     us|cylinder|            2700.0|          45 minutes|This event took
p...|  4/27/2004|29.883055| -97.94111|
|10/10/1949 21:00|    lackland afb|   tx|   null|   light|            7200.0|            1-2 hrs|1949 Lackland AF
B...| 12/16/2005| 29.38421|-98.581085|
|10/10/1956 21:00|            edna|   tx|     us|  circle|              20.0|           1/2 hour|My older brother
...|  1/17/2004|28.978333|-96.645836|
|10/10/1960 20:00|         kaneohe|   hi|     us|   light|             900.0|          15 minutes|AS a Marine 1st
L...|  1/22/2004|21.418056| -157.8036|
|10/10/1961 19:00|         bristol|   tn|     us|  sphere|             300.0|           5 minutes|My father is now
...|  4/27/2007|   36.595| -82.18889|
|10/10/1965 23:45|         norwalk|   ct|     us|    disk|            1200.0|          20 minutes|A bright orange
c...|  10/2/1999|  41.1175| -73.40833|
|10/10/1966 20:00|       pell city|   al|     us|    disk|             180.0|          3  minutes|Strobe Lighted d
i...|  3/19/2009|33.586113| -86.28611|
|10/10/1966 21:00|        live oak|   fl|     us|    disk|             120.0|     several minutes|Saucer zaps ener
g...|  5/11/2005|30.294722| -82.98417|
|10/10/1968 13:00|       hawthorne|   ca|     us|  circle|             300.0|              5 min.|ROUND &#44 ORANG
E...| 10/31/2003| 33.91639|-118.35167|
|10/10/1968 19:00|         brevard|   nc|     us|fireball|             180.0|           3 minutes|silent red /oran
g...|  6/12/2008|35.233334| -82.73444|
|10/10/1970 16:00|        bellmore|   ny|     us|    disk|            1800.0|             30 min.|silver disc seen
...|  5/11/2000| 40.66861|  -73.5275|
|10/10/1970 19:00|      manchester|   ky|     us| unknown|             180.0|           3 minutes|Slow moving&#44
s...|  2/14/2008| 37.15361| -83.76195|
|10/10/1971 21:00|       lexington|   nc|     us|    oval|              30.0|          30 seconds|green oval shape
d...|  2/14/2010| 35.82389| -80.25361|
```

NAN values

Null data sets are like blocks and create skewness, thus the NaN values in columns must be displayed to check the amount of data.

```
In [39]: for cols in ufo.columns:
             try:
                 print({cols: ufo.filter(ufo[cols].isNull()).count()})
             except:
                 pass
```

```
{'datetime': 0}
{'city': 0}
{'state': 2935}
{'country': 8627}
{'shape': 1850}
{'duration (seconds)': 3}
{'duration (hours/min)': 0}
{'comments': 15}
{'date posted': 0}
{'latitude': 0}
{'longitude': 0}
```

Dropping NaN values may help us to do the analysis better.

```
ufo.dropna(how='any').show()
```

```
+----------------+----------------+-----+-------+--------+------------------+--------------------+--------------------
-+-----------+---------+-----------+
|        datetime|            city|state|country|   shape|duration (seconds)|duration (hours/min)|             comment
s|date posted| latitude| longitude|
+----------------+----------------+-----+-------+--------+------------------+--------------------+--------------------
-+-----------+---------+-----------+
|10/10/1949 20:30|      san marcos|   tx|     us|cylinder|            2700.0|          45 minutes|This event took
p...|  4/27/2004|29.883055| -97.94111|
|10/10/1956 21:00|            edna|   tx|     us|  circle|              20.0|           1/2 hour|My older brother
...|  1/17/2004|28.978333|-96.645836|
|10/10/1960 20:00|         kaneohe|   hi|     us|   light|             900.0|          15 minutes|AS a Marine 1st
L...|  1/22/2004|21.418056| -157.8036|
|10/10/1961 19:00|         bristol|   tn|     us|  sphere|             300.0|           5 minutes|My father is now
...|  4/27/2007|   36.595| -82.18889|
|10/10/1965 23:45|         norwalk|   ct|     us|    disk|            1200.0|          20 minutes|A bright orange
c...|  10/2/1999|  41.1175| -73.40833|
|10/10/1966 20:00|       pell city|   al|     us|    disk|             180.0|          3  minutes|Strobe Lighted d
i...|  3/19/2009|33.586113| -86.28611|
|10/10/1966 21:00|        live oak|   fl|     us|    disk|             120.0|     several minutes|Saucer zaps ener
g...|  5/11/2005|30.294722| -82.98417|
|10/10/1968 13:00|       hawthorne|   ca|     us|  circle|             300.0|              5 min.|ROUND &#44 ORANG
E...| 10/31/2003| 33.91639|-118.35167|
|10/10/1968 19:00|         brevard|   nc|     us|fireball|             180.0|           3 minutes|silent red /oran
g...|  6/12/2008|35.233334| -82.73444|
|10/10/1970 16:00|        bellmore|   ny|     us|    disk|            1800.0|             30 min.|silver disc seen
...|  5/11/2000| 40.66861|  -73.5275|
|10/10/1970 19:00|      manchester|   ky|     us| unknown|             180.0|           3 minutes|Slow moving&#44
s...|  2/14/2008| 37.15361| -83.76195|
|10/10/1971 21:00|       lexington|   nc|     us|    oval|              30.0|          30 seconds|green oval shape
d...|  2/14/2010| 35.82389| -80.25361|
|10/10/1972 19:00|   harlan county|   ky|     us|  circle|            1200.0|          20minutes|On october 10&#4
4...|  9/15/2005|36.843056|-83.321945|
|10/10/1972 22:30| west bloomfield|   mi|     us|    disk|             120.0|           2 minutes|The UFO was so c
l...|  8/14/2007|42.537777|-83.233055|
```

To reduce clutter in the data analysis we either need to drop the NaN rows or update them with 0.

```
ufo.select(ufo.columns[0:]).fillna(0)
```

```
DataFrame[datetime: string, city: string, state: string, country: string, shape: string, duration (seconds): float, duration (hours/min): string, comments: string, date posted: string, latitude: float, longitude: float]
```

Data Analysis and Visualisation

In order to facilitate the understanding and insight for the human brain, data visualisation is a method to translate information into a visual environment such as a map or a graph. Identifying patterns, patterns, trends and outliers in huge data sets is the fundamental objective of data visualisation. The word is commonly used with others, including graphical information, information viewing and statistical graphics.

Data visualization is one of the phases of the data science process, whereby it must be displayed for conclusions to be reached following collection, processing and modelling of data. Data visualisation is also part of the wider DPA discipline, aimed at identifying, locating, manipulating, formatting and providing data as efficient as feasible.

For visualisation, we utilise python libraries such as Seaborn, Matplotlib, etc. We need to convert the code to Pandas through ufo.toPandas() for better visualisation.

```
ufo_data = ufo.toPandas()
```

After conversion from the pyspark dataframe to pandas dataframe, we have performed ETA (Explore, Transform and Analysis) process.

ETA

In the context of statistics, data transformation involves the application to each data point of a mathematical expression. In contrast, transformation in a data engineering environment can also indicate data transformation from one format to another throughout the ETL process.

Why should I transform my data?

- Enhanced performance.
- Graphs for decompressing.
- To understand how variables are related.
- To meet assumptions for statistical inference**.**

```
# Choose columns to work with based on our initial questions
ufo_subcols = ['datetime', 'city', 'state', 'country', 'shape', 'duration (seconds)',
        'comments', 'date posted', 'latitude',
       'longitude ']

# After tidying data and choosing what to work with, create dataframe to work with
ufo_data = pd.DataFrame(data=ufo_data, columns=ufo_subcols)

# ...drop null values
ufo_data = ufo_data.dropna(thresh=8)

#...reset the index
ufo_data = ufo_data.reset_index(drop=True)

# EXTRACT LATITUDES #
ufo_data['latitude'] = pd.to_numeric(ufo_data['latitude'],errors = 'coerce')  # latitudes as numerics
ufo_data['longitude '] = pd.to_numeric(ufo_data['longitude '], errors='coerce')

# CHANGE VARIABLES UFO_DATE IN SCRIPT 1-1-18
ufo_date = ufo_data.datetime.str.replace('24:00', '00:00')  # clean illegal values
ufo_date = pd.to_datetime(ufo_date, format='%m/%d/%Y %H:%M')  # now in datetime

ufo_data['datetime'] = ufo_data.datetime.str.replace('24:00', '00:00')
ufo_data['datetime'] = pd.to_datetime(ufo_data['datetime'], format='%m/%d/%Y %H:%M')
```

I followed the below steps for transforming the data and performing visualization techniques.

- Drop the columns with NaN values as this will increase skewness.
- Reset the index.
- Transform latitude and longitude from string to numeric values.
- Transform datetime string values to datetime format.

After doing all these transformations, the data is ready for visualization with 'ufo_data.dtypes' we can observe that the attributes are changed from string to other formats and below is the dataframe.

```
ufo_data.dtypes
```

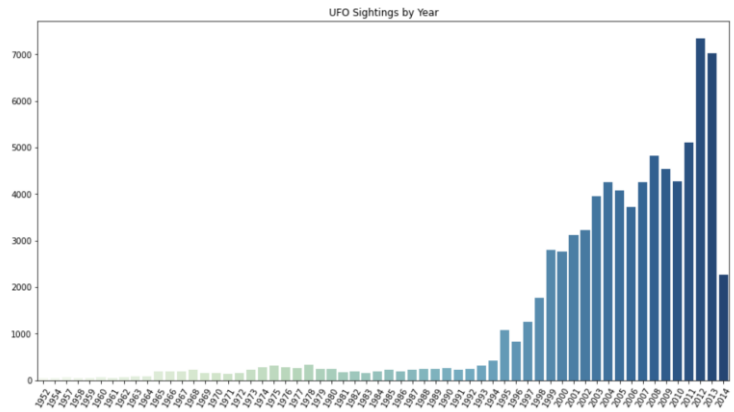
```
datetime              datetime64[ns]
city                          object
state                         object
country                       object
shape                         object
duration (seconds)            object
comments                      object
date posted                   object
latitude                     float64
longitude                    float64
season                        object
hemisphere                  category
dtype: object
```

```
ufo_data
```

	datetime	city	state	country	shape	duration (seconds)	comments	date posted	latitude	longitude
0	1949-10-10 20:30:00	san marcos	tx	us	cylinder	2700	This event took place in early fall around 194...	4/27/2004	29.883056	-97.941111
1	1949-10-10 21:00:00	lackland afb	tx	NaN	light	7200	1949 Lackland AFB&#44 TX. Lights racing acros...	12/16/2005	29.384210	-98.581082
2	1955-10-10 17:00:00	chester (uk/england)	NaN	gb	circle	20	Green/Orange circular disc over Chester&#44 En...	1/21/2008	53.200000	-2.916667
3	1956-10-10 21:00:00	edna	tx	us	circle	20	My older brother and twin sister were leaving ...	1/17/2004	28.978333	-96.645833
4	1960-10-10 20:00:00	kaneohe	hi	us	light	900	AS a Marine 1st Lt. flying an FJ4B fighter/att...	1/22/2004	21.418056	-157.803611
...	...	...	...	...	...	...	...	...	...	...
80252	2013-09-09 21:15:00	nashville	tn	us	light	600	Round from the distance/slowly changing colors...	9/30/2013	36.165833	-86.784444
80253	2013-09-09 22:00:00	boise	id	us	circle	1200	Boise&#44 ID&#44 spherical&#44 20 min&#44 10 r...	9/30/2013	43.613611	-116.202500
80254	2013-09-09 22:00:00	napa	ca	us	other	1200	Napa UFO&#44	9/30/2013	38.297222	-122.284444
80255	2013-09-09 22:20:00	vienna	va	us	circle	5	Saw a five gold lit cicular craft moving fastl...	9/30/2013	38.901111	-77.265556
80256	2013-09-09 23:00:00	edmond	ok	us	cigar	1020	2 witnesses 2 miles apart&#44 Red & White...	9/30/2013	35.652778	-97.477778

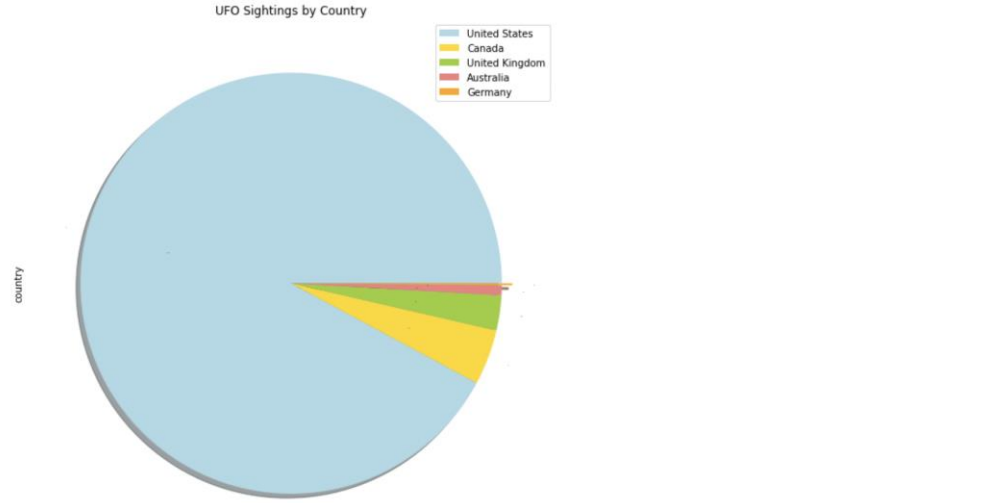
we can see a large number of sightings in the year 2012 from the barplot below.

```
ufo_yr = ufo_date.dt.year  # series with the year exclusively

## Set axes ##
years_data = ufo_yr.value_counts()
years_index = years_data.index  # x ticks
years_values = years_data.values

## Create Bar Plot ##
plt.figure(figsize=(15,8))
plt.xticks(rotation = 60)
plt.title('UFO Sightings by Year')

years_plot = sns.barplot(x=years_index[:60],y=years_values[:60], palette = "GnBu")
```



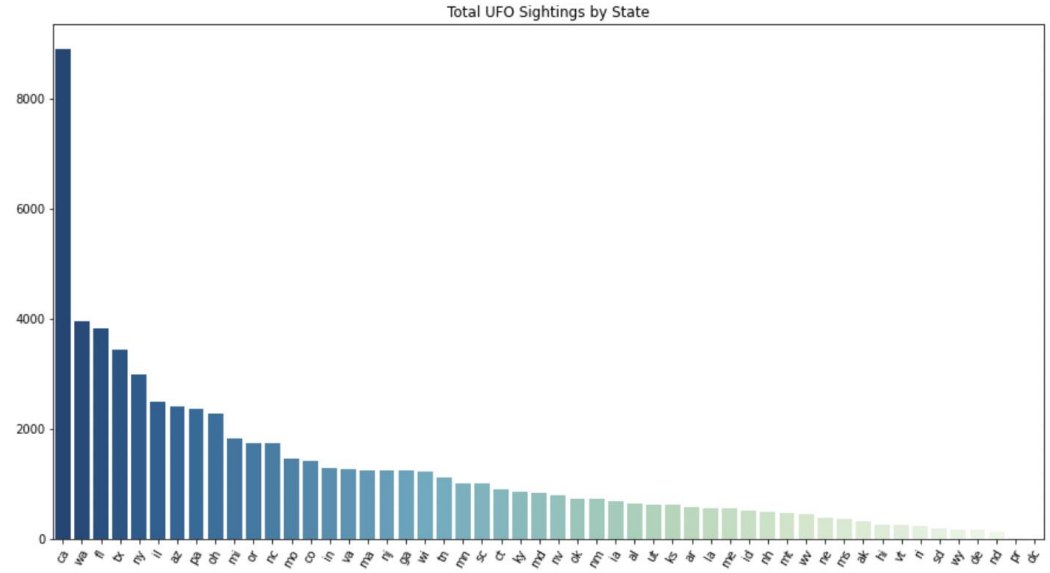
The data represented at the bottom is for the sightings in the country.

```
country_sightings = ufo_data.country.value_counts()  # num ufo sightings per country

explode = (0, 0, 0, 0., 0.05)
colors = ['lightblue','gold','yellowgreen','lightcoral','orange']
country_sightings.plot(kind = 'pie', fontsize = 0, title='UFO Sightings by Country', colors=colors,
                       autopct='%1.1f%%',shadow=True, explode=explode,figsize=(8,8))
plt.legend(labels=['United States','Canada','United Kingdom','Australia','Germany'], loc="best")
plt.tight_layout()
```

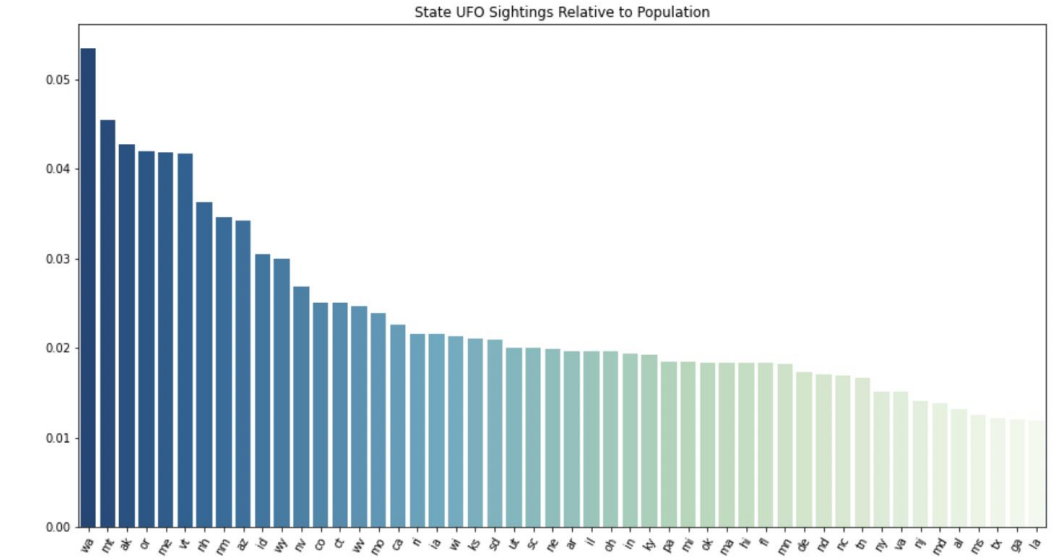


This barplot depicts the sighting that occurred in the state of California ‘ca’.

```
# Filter US Values to analyze US state sightings #
usa_filter = ufo_data['country']=='us'  # filter non-usa country
us_data = ufo_data[usa_filter]  # DF ufo_data of only US sightings - includes puerto rico and dc

# Get x and y axes for states bar viz #
states_sights = us_data.state.value_counts()  # State Data
state_names = states_sights.index  # x axis ticks
state_freq = states_sights.values  # y axis values

# States Frequency Pareto Chart #
plt.figure(figsize=(15,8))
plt.xticks(rotation = 60)
plt.title('Total UFO Sightings by State')
states_plot = sns.barplot(x=state_names,y=state_freq, palette="GnBu_r")
plt.show()
print('Top 10 States for Total UFO Sightings:')
print(states_sights[:10].sort_values(ascending=False))
```

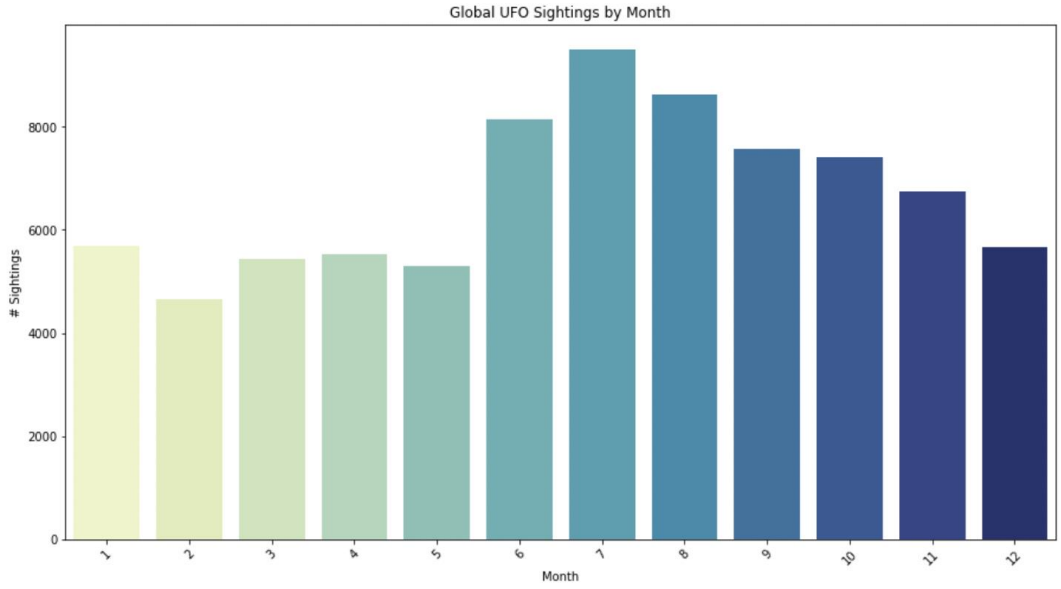


This barplot indicates ‘wa’ state has more occurrences.



The UFO's globally appear in the 7th and 8th months of a year.

```
m_cts = (ufo_data['datetime'].dt.month.value_counts()).sort_index()
m_ctsx = m_cts.index
m_ctsy = m_cts.values
f, ax = plt.subplots(figsize=(15,8))

sns.barplot(x=m_ctsx, y=m_ctsy, palette="YlGnBu")
ax.set_title('Global UFO Sightings by Month')
ax.set_xlabel('Month')
ax.set_ylabel('# Sightings')
plt.xticks(rotation=45)
plt.show()
```



We will append the season of a year to the dataframe to view any weather differences occur when it appears.

```
# Add Season Column to ufo_date #
# Given a datetime, return the season that it's in #
ufo_datem = ufo_date.dt.month
spring = range(5,7)
summer = range(7,10)
fall = range(10,12)
seasons = []

for st_date in ufo_datem:
    # Conversion Process #
    if st_date in spring:
        seasons.append('Spring')
    elif st_date in summer:
        seasons.append('Summer')
    elif st_date in fall:
        seasons.append('Fall')
    else:
        seasons.append('Winter')
ufo_data['season'] = pd.Series(seasons, index=ufo_data.index)
```

```
# Add Hemisphere Column to ufo_date #
hemis = []
for st_loc in ufo_data['latitude']:
    if st_loc >= 0 :
        hemis.append('Northern Hemisphere')
    else:
        hemis.append('Southern Hemisphere')
```

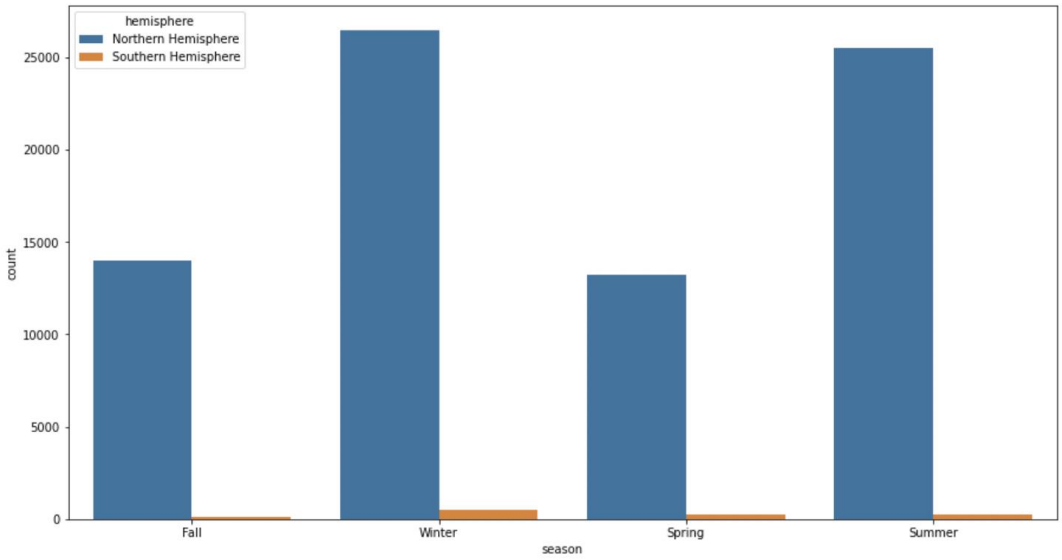
```
ufo_data['hemisphere'] = (pd.Series(hemis, index=ufo_data.index)).astype('category')
ufo_data['season'].value_counts()
```

```
Winter    26971
Summer    25719
Fall      14142
Spring    13425
Name: season, dtype: int64
```

The plot indicates what part of the globe will occur in the Northern Hemisphere, as more UFO encounters tend to occur.

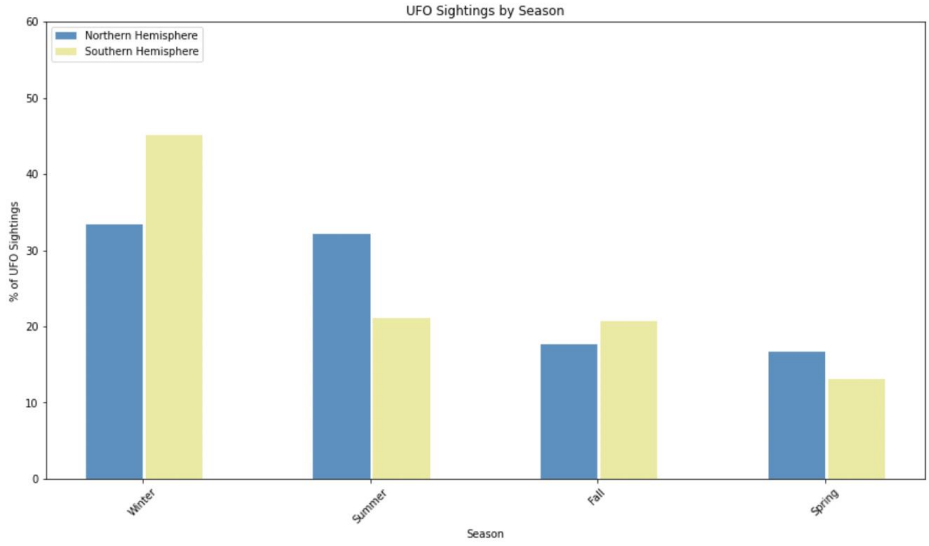
```
plt.figure(figsize=(15,8))
sns.countplot(x='season', hue='hemisphere', data=ufo_data)
```

```
<AxesSubplot:xlabel='season', ylabel='count'>
```

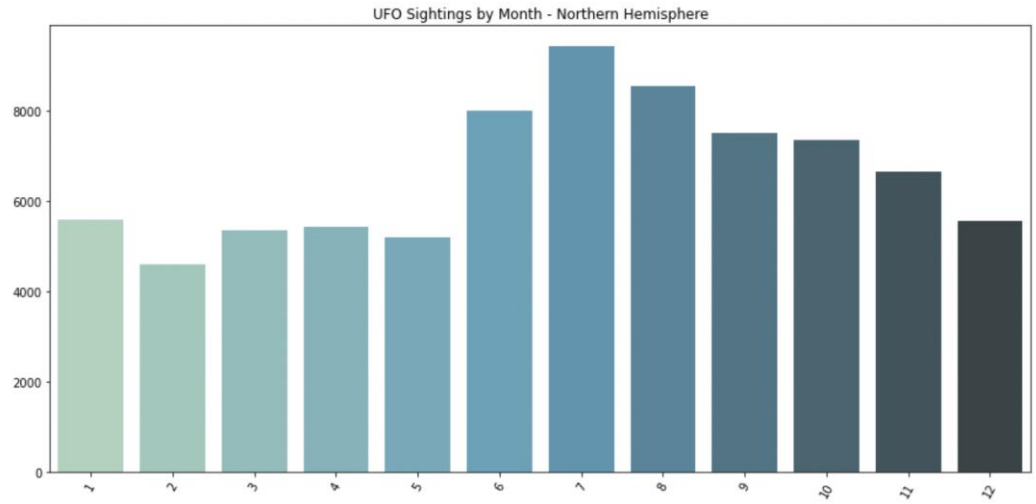


The probability of UFOs are likely to appear in the winter season is high.

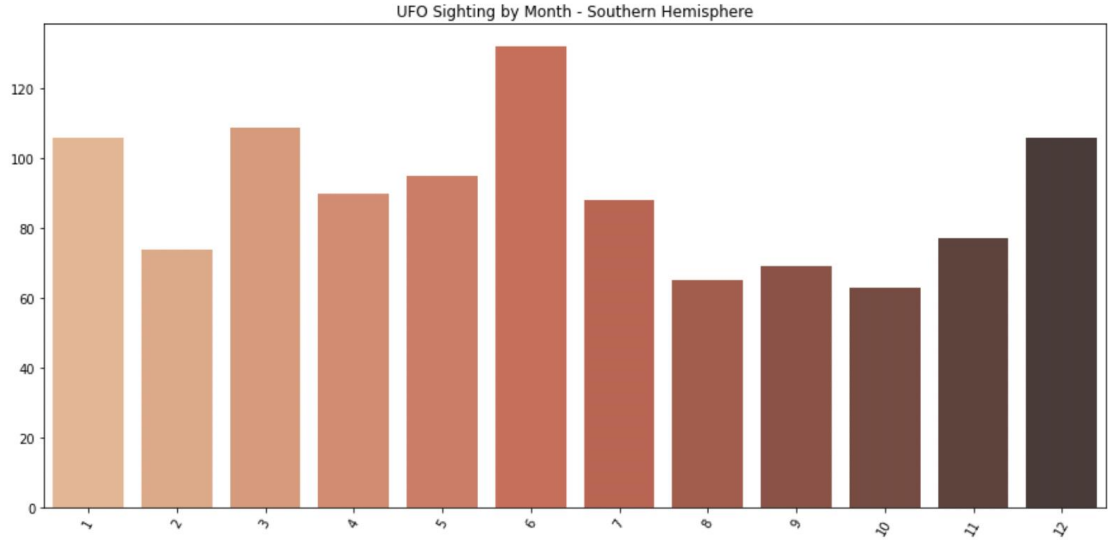
```
nsperc = resp_n['season'].value_counts() / len(resp_n) * 100
ssperc = resp_s['season'].value_counts() / len(resp_s) * 100

pos = list(range(len(nsperc)))
width = 0.25
fig, ax = plt.subplots(figsize = (15,8))

plt.bar(pos, nsperc, width, alpha = .7, color='#0064A9')
plt.bar([p + width*1.05 for p in pos], ssperc, width, alpha = .65, color='#E1E066')

ax.set_title('UFO Sightings by Season')
ax.set_xlabel('Season')
ax.set_ylabel('% of UFO Sightings')
ax.set_xticks([p + .5 * width for p in pos])
ax.set_xticklabels(nsperc.index)
plt.xticks(rotation=45)

plt.ylim([0, 60])
plt.legend(['Northern Hemisphere','Southern Hemisphere'], loc='upper left')
plt.show()
print('Northern Hemisphere:\n', nsperc)
print('Southern Hemisphere:\n',ssperc)
```



The probability of UFO's appearing is in the 7th month of a year in the northern hemisphere.



The probability of UFO's appearing is in the 6th month of a year in the southern hemisphere.



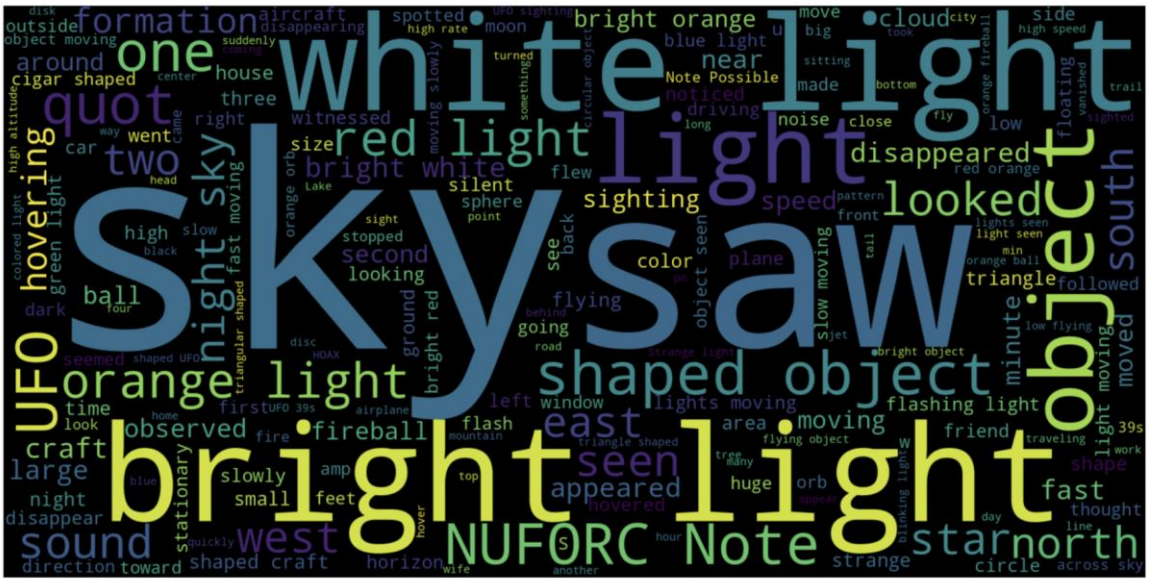
The count of months from the index can be observed below.

```
Top Months for UFO Sightings in N. Hemishpere:
7     9424
8     8562
6     8004
9     7511
10    7343
11    6659
Name: datetime, dtype: int64
Top Months for UFO Sightings in S. Hemisphere:
6     132
3     109
1     106
12    106
5      95
4      90
Name: datetime, dtype: int64
```

We can build wordcloud from the population comments.

```
from wordcloud import WordCloud
words = [i for i in df.comments.dropna()]
words = " ".join(words)
plt.subplots(figsize=(28,12))
wordcloud = WordCloud(background_color='black',
                          width=2048,
                          height=1024
                          ).generate(words)
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('graph.png')

plt.show()
```

Wordcloud is in various sizes, representing frequency or the value of each word, which may have often been found in you. This is known as Tag Cloud. You learn how to construct your own WordCloud in Python and to tweak it according to your wishes for this lesson. This tool will make your report more vibrant and helpful to explore text data.



MACHINE LEARNING

Linear Regression

We shall update and assign a dataframe with the datetime module of python. Group the 'report' column to change where the credibility reports are either high or low.

```
import datetime

data = df.dropna()
data = data.reset_index(drop=True)
data = data[['shape','latitude','longitude','datetime','comments']]
data['datetime'] = pd.to_datetime(data.datetime, format='%Y/%m/%d HH:MM', errors='ignore')

# data['datetime'].apply(lambda x: pd.Timestamp(x).date())
data = data[(data['datetime'] <= '1/1/2017 20:30')]
# data[(data['datetime'] <= datetime.date(2017, 1, 1))]



cols = ['shape', 'latitude', 'longitude', 'datetime']
data['reports'] = data.groupby(cols).transform('count')
data['credibility'] = np.where(data['reports']>=2, 'High', 'Low')
```

The vectorizing procedure is used to transform the algorithm from one value to the value set at a time. Modern CPUs let vectors work immediately when there are several data with a single command and calibrate the support vector machines.

```
#Define features (summaries) and labels (credibilities)
summaries = data['comments']
credibilities = data['credibility']

#Vectorize features, aka text summaries.
vectorizer = TfidfVectorizer(ngram_range=(1,3), min_df=5)
vectors = vectorizer.fit_transform(summaries)

f_train, f_test, l_train, l_test = train_test_split(vectors, credibilities, test_size = 0.3)
svm = LinearSVC(C=0.7, class_weight='balanced')
clf = CalibratedClassifierCV(svm)
clf.fit(f_train, l_train)
```

Classification report tells the f-1 score for the model that put for the trained model and testing model.

```
"""Training Data Predictions."""
pred_train = clf.predict(f_train)
print(classification_report(l_train, pred_train))
```

```
              precision    recall  f1-score   support

        High       0.00      0.00      0.00         2
         Low       0.99      1.00      1.00       361

    accuracy                           0.99       363
   macro avg       0.50      0.50      0.50       363
weighted avg       0.99      0.99      0.99       363
```

```
"""Testing Data Predictions."""
pred_test = clf.predict(f_test)
print(classification_report(l_test, pred_test))
```

```
              precision    recall  f1-score   support

        High       0.00      0.00      0.00         2
         Low       0.99      1.00      0.99       154

    accuracy                           0.99       156
   macro avg       0.49      0.50      0.50       156
weighted avg       0.97      0.99      0.98       156
```

To check if the model is working we will pass on some arguments to check the credibility percentage of the model.

```
"""Using the model: Percent chance credibilities."""
fake = ["I saw a white flying saucer heading southeast while I was out with my dog.\
         It hovered for a bit before disappearing. I haven't told anyone else about it."] # I wrote this, haha...

real = ["My wife and I were walking our dogs on the south side of South Mountain\
        in the mountain park when we saw the ufo come over the mountain at low\
        altitude (500'appx) headed southbound.  It was s i l e n t and had 5 white\
        globes along the leading edge of a black triangle shaped object which blacked\
        out the stars and was about a mile wide."] # description from a sighting with multiple reports...

def credibility(summary):
    """String needs to be in a list, e.g. [Description]"""
    s = vectorizer.transform(summary)
    y_proba = clf.predict_proba(s)
    prob = y_proba[0][0]*100
    if prob > 30.0: #50% too large since even mass sightings will be around 55-60%
        return print("High credibility: " + "%.2f" % (y_proba[0][0]*100) + "%")
    else:
        return print("Low credibility: " + "%.2f" % (y_proba[0][0]*100) + "%")
```

```
credibility(fake) #credibility percentage of a fake sighting
credibility(real) #credibility percentage of a real, mass sighting
```

```
Low credibility: 1.73%
Low credibility: 1.91%
```

DISCUSSION

Our report deeply gives insights about UFO sightings based on Seasons, Months, Dates, Countries & Years. In this report, we have discussed the ufo sightings from the year 1952 to 2014 and we also discuss the countries where possible sightings occurred in the past. While cleansing the data there were a number of missing data which might create skewness, hence not to create any discrepancies in the data, we drop the null data. We also discuss in our report that the number of UFO's sightings occur mostly in the USA than in any other country.

CONCLUSION

Although it is possible to have seen a UFO in the world anytime soon there is no credible information or any government responses that these sightings are clear.

Possible explanations for UFO sightings

The precision of UFO sightings has broad variation, as assessed by the number of witnesses, the independence of the witnesses and the monitoring circumstances (e.g., fog, haze, lighting type) and the direction of the sighting. Typically, witnesses who take the time to record a sighting believe the object is of extra-terrestrial origin or a military ship under intelligent control. This assumption is typically based on what seems to be formation flying by groups of objects, unusual often sudden motions, a lack of sound, changes in brightness or colour, and bizarre forms.

Tricks are well-known to unaided eyes. A luminous light frequently seems to move, as does Planet Venues. The drivers also find astronomical objects unsettling since they "follow" the automobile. Visual impressions of the distance and velocity of UFOs are often very inaccurate, given that they are based on an imagined size and are typically produced against a plain sky without any background material (clouds, mountains...). Windows and eyeglass reflections provide superposed vistas, while sophisticated optical systems, such as camera lenses, may make point light sources seem subdued.

Such optical illusions and the psychological need for pictures are famous for a large number of visual UFO claims and some observations are acknowledged to be falsifications. Radar views are not distinguished between manmade objects and meteor trails, ionised gas, rain, or thermal discontinuities in the atmosphere, but are more trustworthy in certain ways. Contacts like kidnappings are often associated with UFOs as they are attributed to extraneous visitors. But the validity of ETH as an explanation of abductions is doubtful by most psychologists who investigated this phenomenon. A

regular experience is known as "sleep paralysis" is to blame, because it makes sleepers feel momentarily immobile.

Let's take a look back and see what we would want to anticipate after having a look at UFO patterns. First, there is no way to say whether or not visitation is a real UFO, but we may utilise mass visits as a measure of how reliable a visit is. This implies that if many individuals claim a sighting (similar shape, place, and date), then it's far more credible than a single person reported it. This is not mean single testimony sightings are not real, but they are not as trustworthy as multi-testing sightings.

Appendix

dataset

https://www.kaggle.com/NUFRC/ufo-sightings

REFERENCES

Bharill, N., Tiwari, A., & Malviya, A. (2016). Fuzzy Based Scalable Clustering Algorithms for Handling Big Data Using Apache Spark. *IEEE Transactions on Big Data*, *2*(4), 339-352. https://doi.org/10.1109/tbdata.2016.2622288

Unidentified flying object - Wikipedia. En.wikipedia.org. (2021). https://en.wikipedia.org/wiki/Unidentified_flying_object

unidentified flying object | History, Sightings, & Facts. Encyclopaedia Britannica. (2021). https://www.britannica.com/topic/unidentified-flying-object

Vectorization, Part 2: Why and What? Quantifisolutions.com. https://www.quantifisolutions.com/vectorization-part-2-why-and-what

What is Data Filtering? Displayr. (2021). https://www.displayr.com/what-is-data-filtering/

What is data visualization and why is it important? SearchBusinessAnalytics. https://searchbusinessanalytics.techtarget.com/definition/data-visualization

Configuration - Spark 2.1.0 Documentation. Spark.apache.org. (2021). https://spark.apache.org/docs/2.1.0/configuration.html

Apache Spark™ - Unified Analytics Engine for Big Data. Spark.apache.org. https://spark.apache.org/.