The code was written in Python in part due to the strengths of the development team but also due to the sponsor mentioning Python as one of the supported languages by their organization. Python offers many tools in the form of libraries that can be useful in data analysis projects such as this one. The following table contains a list of libraries used in this project and their purpose:

Purpose	Libraries Used
Importing large raw data files	Dask
Storing smaller data files	Pandas
Working with time functions	datetime
Computations	Pandas Numpy Math PandasSQL
Working with geographic data	Geopy Shapely
AWS Specific libraries	Sagemaker Boto3 OS

Table 1: Python libraries used throughout the solution

Figure 1 depicts the high-level overview of the flow of the algorithm. The specifics of the functions will be explained further in this section.

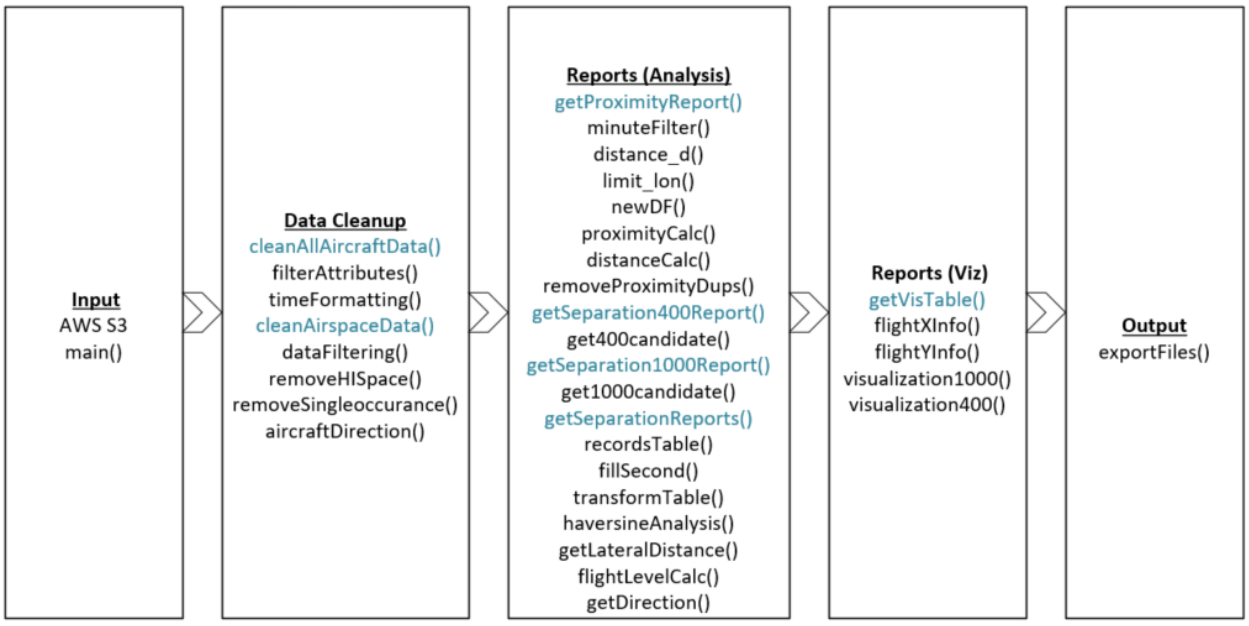

Figure 1: Algorithm Flow

There are data frames that are stored as global since they are accessed by many of the functions. Passing the full data frames from function to function at times caused timeouts and general slowness, therefore it was decided to have a few key data frames be global so that they could be accessed from inside functions without needing to pass them from function to function each time. The following data frames were created as global:

Data Frame Name	Description
rawData_df	Dask data frame containing the full raw dataset as it was imported from the AWS (Amazon Web Services) S3 bucket.
allAircraftData	Pandas data frame containing every record from the original dataset with only the attributes that are used in the code. New attributes were added to hold the parsed our date and time.
airspaceData	Pandas data frame containing the fully formatted data that is needed to create the "proximityReport" output.

Table 2: Global data frames

The solution code is broken into functions which are then called by a main function. This was done to break out code into a format that would be easy to maintain and update if needed. Also, having defined functions helps with all the loops and iterations that need to occur to get to the final reports. The "proximityReport", "separation400Report", and "separation1000Report" all have records where there are two flights per record since these were within a proximity of interest. The "visualization400" and "visualization1000" reports break up the "separation400Report" and the "separation1000" reports into one flight per record since that is the input required for visualizations. The following table includes a summary of the functions in the algorithm:

Name	Input	Output
main	None	None
Description		
1. Set up the global "airspaceData" and "allAircraftData" tables 2. Call on the functions to clean data 3. Call on the functions to create reports 4. Call on the function to export reports back to AWS S3 bucket		
Name	**Input**	**Output**
filterAttributes	None	airspaceData
Description		
1. Creates a table named "airspaceData" from the full raw data which only has the attributes needed for this project 2. Changes the flight levels from the shorthand notation to the full flight level in feet in the newly created table 3. Returns the newly created and updated table		
Name	**Input**	**Output**

timeFormatting	allAircraftData	allAircraftData
Description		
1. Receives the table as returned from the “filterAttributes” function 2. Changes date/time attribute from string data type to dateTime data type for future computations 3. Separates the date/time attribute into separate date and time attributes for future computations 4. Returns the with the edits made in this block		
Name	**Input**	**Output**
dataFiltering	None	None
Description		
1. Adds to the global “airspaceData” table only the records from the “allAircraftData” table that have a flight level at or above FL240 2. Removes records that do not occur within the first 5 seconds of each minute		
Name	**Input**	**Output**
removeHISpace	None	None
Description		
1. Creates a polygon of Hawaii airspace as defined by data provided to from project sponsors 2. Checks for each record from the “airspaceData” table whether it is within the Hawaii polygon or not and flags it 3. Removes the records from the “airspaceData” table flagged as being within the Hawaii polygon (airspace)		
Name	**Input**	**Output**
removeSingleoccurrence	None	None
Description		
1. Checks the “airspaceData” for any Target IDs that only occur once and removes those records (causes errors when only one record exists, and these target IDs were typically not found to be real flight IDs)		
Name	**Input**	**Output**
aircraftDirection	None	None
Description		
1. Assigns the direction of East or West based on the selected heading 2. Limitation: if there is no selected heading or if it is recorded as less than 0 degrees then it will mark the direction as “not applicable.” This is no longer a limitation later in the reports when looking at the points that were flagged of interest at the 400 and 1000 levels		
Name	**Input**	**Output**
minuteFilter	HourCounter, MinuteCounter	recordsInMinute
Description		
1. For each unique Target ID, the first flight that occurred in that minute and hour (HourCounter, MinuteCounter) that is being analyzed is filtered and stored in a “recordsInMinute” table [done to account for the “time” variable and make it negligible from the other functions for the rest of this portion of the analysis since this project deals with four dimensions: time, latitude, longitude, height]		
Name	**Input**	**Output**
distance_d	point0, pointX	distance
Description		

1. Input the coordinates for the two flights that need the lateral distance between them
2. Compute the distance between two points using the standard Haversine Formula
3. Convert the output from the Haversine Formula into Nautical Miles
4. Return the distance

Name	Input	Output
limit_lon	point0	Boundaries around point

Description

1. Input is the aircraft being analyzed by its latitude and longitude
2. Computer the boundaries around the aircraft created when looking at a 25 nautical mile radius around it
3. Returns the boundary around the aircraft to later see which aircraft comes near it

Name	Input	Output
newDF	OrderDF,x,y,d	OrderResult

Description

1. Input is the table with the information for the aircraft, latitude and longitude of the two aircraft being analyzed, and the lateral distance between the aircraft
2. Creates and formats the “OrderResult” table to have the two aircraft being analyzed and their information all in one record side by side
3. Returns the newly created “OrderResult” table

Name	Input	Output
proximityCalc	LongitudeOrderDF	Resultsdf

Description

1. Input is the table returned from the “minuteFilter” function
2. With the use of other functions and its own logic, calculates the lateral distance of each aircraft in the “minuteFilter” table one aircraft a time from every other aircraft that was in the airspace during that minute
3. When a distance between two aircraft is found to be at or below 25 nautical miles, the pair of aircraft and their information as well as distance is recorded in the Resultsdf table with the correct format
4. The newly created results table is returned

Name	Input	Output
distanceCalc	resultsDF	resultsDF

Description

1. Input is the table returned from the “proximityCalc” function
2. Calculates the height difference in feet between the two aircraft listed in each record and stores the height difference
3. Flags the height differences as “True/False” for being at or below 400 feet and “True/False” for being below 1000 feet
4. Returns the table with the new fields: height difference, flag for the difference being at or below 400 feet, flag for the height difference being below 1000

Name	Input	Output
removeProximityDups	proximityReport	proximityReport

Description

1. Gets the table created in “getProximityReport”
2. Removes records that did not have flight IDs due to incomplete data in the raw data file

3. Removes records that are copies of each other due to how they were compared (ex: the point "Flight A, Flight B" is considered different than "Flight B, Flight A" although they are in fact duplicate records)
4. Returns the table with the cleaned-up records

Name	Input	Output
getProximityReport	None	proximityReport

Description

1. Sequence of steps to get the proximity report by looping through every minute of the day (00:00 to 23:59) and getting the details to store in a "proximityReport" table:
    1.1. Records at the given minute from the "minuteFilter" function
    1.2. Pairs of flights in that minute that were within 25 nautical miles of each other from the "proximityCalc" function
    1.3. Height difference between aircraft within 25 nautical miles at the start of the minute and whether they are within the 400- and 1000-feet difference
    1.4. Adding all the results for this minute to the running "proximityReport" for the day
    1.5. Removing duplicate entries
2. Returns the "proximityReport" table

Name	Input	Output
get400candidate	proximityReport	LossCandidates400

Description

1. Input is the table from the "getProximityReport" function
2. Checks the "proximityReport" table for any values marked as "True" for having a flight level difference of 400 or less
3. Returns the table with the results on an empty table as "LossCandidates400"

Name	Input	Output
get1000candidate	proximityReport	LossCandidates1000

Description

1. Input is the table from the "getProximityReport" function
2. Checks the "proximityReport" table for any values marked as "True" for having a flight level difference of less than 1000
3. Returns the table with the results on an empty table as "LossCandidates1000"

Name	Input	Output
recordsTable	InstancesAtLevel, X	flightInformation

Description

1. Input is the table from either "get400candidate" function or "get1000candidate" function and the location of the record being analyzed
2. Gets the records from the global "allAircraftData" table for 5 minutes before and after the recorded potential separation. Saves the records to the "flightInformation" table
3. Returns the newly created "flightInformation" table

Name	Input	Output
fillSecond	data_x, data_y	y_transformed

Description

1. Input is the records for the two flights causing the potential separation, both with the records 5 minutes before and after the potential separation
2. Puts the two flights on the same time scale for comparison
3. Linear interpolation is completed to fill in the times for the seconds on flight "y" that do not match those in flight "x" and saves it to the "y_transformed" table - (this is needed because

the readings don't happen at the same time but for comparison it is best to have them on the same scale)
4. Returns the "y_transformed" table

Name	Input	Output
transformTable	flightData	analyzedTable

Description

1. Input is tables returned from either the "getSeparation400" table or the "getSeparation1000" table
2. Sends the records to the "fillSeconds" function
3. Removes records that were returned with no latitude or longitude - (if the first record of a flight is at 9:00:00 and the function was looking to do an interpolation for 8:57:00 for example, those times where the flight were not present are returned with an empty latitude and longitude)

Name	Input	Output
haversineAnalysis	lat1, lon1, lat2, lon2, to_radians=True, earth_radius=6371	lateral distance

Description

1. Inputs are the two points for which the lateral distance is being calculated
2. The Haversine formula is used to get the distance between the two points
3. The distance is returned in nautical miles

Name	Input	Output
getLateralDist	analyzedTable	analyzedTable

Description

1. Input is the table created in the "getSeparationReports" table which has the information for +/- five minutes from when the potential deviation was flagged
2. Calls on the "haversineAnalysis" function to get the lateral distance and records the result
3. Returns the "analyzedTable" with the lateral distances appended

Name	Input	Output
flightlevelCalc	analyzedTable	analyzedTable

Description

1. Input is the table created in the "getSeparationReports" table which has the information for +/- five minutes from when the potential deviation was flagged
2. Calculates the flight level distance (in feet) between the pair of flights for all the interpolated intervals of the flight
3. Returns the "analyzedTable" with the flight level distances appended

Name	Input	Output
getDirection	analyzedTable	analyzedTable

Description

1. Input is the table created in the "getSeparationReports" table which has the information for +/- five minutes from when the potential deviation was flagged
2. Determines the direction of the aircraft (E/W) based on the latitude and longitude position of the first entry for the flight and the second to last
3. Returns the "analyzedTable" with the flight level distances appended

Name	Input	Output
getSeparationReports	instancesAtLevel	`separationReport`

Description

1. Input is the table resulting from the “getCandidate400” function or the “getCandidate1000” functions
2. Sets the order for the steps needed to generate the report and stores the report as the “separationReport” table
    2.1. Get the raw data for 5 minutes before and after the potential loss of separation by calling “recordsTable” function
    2.2. Put the interpolated flight information side by side using the “transformTable” function
    2.3. Get the lateral distance (“getLateralDist” function), flight level distance (“flightLevelCalc” function), and the direction of the flight (“getDirection” function)
3. Returns the “separationReport” table

Name	Input	Output
flightXInfo	separationData	flightX

Description

1. Input is the table returned from the “getSeparationReports” table
2. Pulls the information for just the first flight from the combined “separationData” table and stores it in a table for just the first flight (“flightX”)
3. Returns the table with the information for one flight as the “flightX” table

Name	Input	Output
flightYInfo	separationData	flightY

Description

1. Input is the table returned from the “getSeparationReports” table
2. Pulls the information for just the first flight from the combined “separationData” table and stores it in a table for just the first flight (“flightY”)
3. Returns the table with the information for one flight as the “flightY” table

Name	Input	Output
getVisTable	Resulttable	tableToVisualize

Description

1. Input is the table from the “getSeparation400Report” table or the “getSeparation1000Report” table
2. Get the separate values for each flight and combine them one after the other rather than side by side and save it to the “tableToVisualize” table – (each record will only have one flight)
3. Return the "tableToVisualize” table

Name	Input	Output
getSeparation1000Report	proximityReport	flSeparation1000Report

Description

1. Input is the report from the “getProximityReport” function
2. Call the “get1000candidate” function to get the list of potential separations below 1000 feet and save to the “flSeparation1000report”
3. (If there are results from step 2) call the “getSeparationReports” function to get the information for the potential loss of separations as a side-by-side report
4. Remove any records returned blank (information not available due to there not being records of the flight at a given time)
5. Return the “flSeparation1000report”

Name	Input	Output
visualization1000	report1000	viz1000Data

Description

1. Input is the file from the “getSeparation1000Report” function
2. Sends the input report into the “getVizTable” function to split up the records then saves it to the “viz1000Data” table
3. Adds a flag to each record in the “viz1000Data” table as to whether they are below the 1000 ft separation level so that this does not have to be computed by the visualization tool
4. Returns the report as the “viz1000Data” table

Name	Input	Output
getSeparation400Report	proximityReport	getSeparation400Report

Description

1. Input is the report from the “getProximityReport” function
2. Call the “get400candidate” function to get the list of potential separations at or below 400 feet and save to the “flSeparation400report”
3. (If there are results from step 2) call the “getSeparationReports” function to get the information for the potential loss of separations as a side-by-side report
4. Remove any records returned blank (information not available due to there not being records of the flight at a given time)
5. Return the “flSeparation400report”

Name	Input	Output
visualization400	report400	viz400Data

Description

1. Input is the file from the “getSeparation1000Report” function
2. Sends the input report into the “getVizTable” function to split up the records then saves it to the “viz400Data” table
3. Adds a flag to each record in the “viz400Data” table as to whether they are below the 1000 ft separation level so that this does not have to be computed by the visualization tool
4. Returns the report as the “viz400Data” table

Name	Input	Output
exportFiles	proximityReport, separation400Report, visualization400Report, separation1000Report, visualization1000Report	None

Description

1. Input is all the reports that have been saved as pandas data frames
2. Converts all the Pandas data frames to CSV files
3. Create the path for the location in AWS where files will be stored
4. Save the CSV files to the instructed path in the AWS S3 bucket

Name	Input	Output
cleanAllAircraftData	None	None

Description

1. Calls on functions to make edits to the global “allAircraftData” table
    1.1. Call on the “filterAttributes” function
    1.2. Call on the “timeFormatting” function

Name	Input	Output
cleanAirspaceData	None	None

Description

1. Calls on functions to make edits to the global “allAircraftData” table

1.1. Call on the “dataFiltering” function
1.2. Call on the “removeHISpace” function
1.3. Call on the “removeSingleoccurence” function
1.4. Call on the “aircraftDirection” function

Table 3: Function Definitions