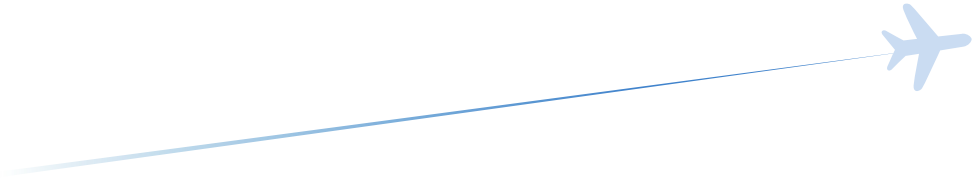

Air traffic complexity estimation module

Deliverable ID:	**D3.3**
Dissemination Level:	**PU**
Project Acronym:	**AISA**
Grant:	**892618**
Call:	**H2020-SESAR-2019-2**
Topic:	**SESAR-ER4-01-2019**
Consortium Coordinator:	**FTTS**
Edition date:	**31 March 2021**
Edition:	**00.01.00**
Template Edition:	**02.00.02**

Founding Members

EUROPEAN UNION EUROCONTROL

SESAR JOINT UNDERTAKING

Authoring & Approval

Authors of the document

Name/Beneficiary	Position/Title	Date
Ivan Tukarić/FTTS	FTTS contributor	23/02/2021

Reviewers internal to the project

Name/Beneficiary	Position/Title	Date
Bruno Antulov-Fantulin/FTTS	Project Associate	26/02/2021
Javier Alberto Perez Castan/UPM	Task 3.2 Leader	29/03/2021
Lars Ludger Schmidt/TUBS	Task 3.1 Leader	29/03/2021
Tomislav Radišić/FTTS	Project Coordinator	30/03/2021

Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date
Tomislav Radišić/FTTS	Project Coordinator	30/03/2021

Rejected By - Representatives of beneficiaries involved in the project

Name/Beneficiary	Position/Title	Date

Document History

Edition	Date	Status	Author	Justification
00.00.01	15/01/2021	Revision	Ivan Tukarić/FTTS	New Document
00.00.02	08/03/2021	Revision	Ivan Tukarić/FTTS	First Draft
00.00.03	16/03/2021	Revision	Ivan Tukarić/FTTS	Second Draft
00.00.04	31/03/2021	Revision	Ivan Tukarić/FTTS	Final

Copyright Statement

AISA

AI SITUATIONAL AWARENESS FOUNDATION FOR ADVANCING AUTOMATION

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 892618 under European Union's Horizon 2020 research and innovation programme.

Abstract

Air traffic complexity models use different bases for determining complexity. The goal is to build an air traffic complexity model which will be able to determine complexity regardless of the traffic situation, observed airspace, or air traffic controller. This is done by implementing a novel solution that uses air traffic controller tasks, defined depending on the traffic situation, to determine air traffic complexity. This approach offers a solution to certain problems recognized in previous complexity models.

As opposed to air traffic controllers grading one traffic situation by its complexity, a comparison method is used. Changing the grading method from grading one traffic situation to comparing two traffic situations solves the problem of assessment inconsistency – grading the traffic situations one by one may result in equal complexity grades for traffic situations of different levels of air traffic complexity. The problem of airspace bounding is solved by creating generic airspace for air traffic situations. This reduces the impact of controller familiarity with specific airspace.

The model uses air traffic situation characteristics to determine tasks for each aircraft in a given traffic situation. Then the comparison method output and calculated tasks are used to develop a new model for determining air traffic complexity by using machine learning techniques. The model is validated by using the same comparison method on model-unseen, actual airspace.[1]

[1] The opinions expressed herein reflect the author's view only. Under no circumstances shall the SESAR Joint Undertaking be responsible for any use that may be made of the information contained herein.

Purpose

In WP3 three individual ML modules were developed to be further studied in the field of shared situational awareness in the AISA project. Deliverable D3.1 presents a trajectory prediction module and deliverable D3.2 presents a conflict detection module.

Deliverable D3.3 describes the creation of a model for determining air traffic complexity regardless of airspace, traffic or air traffic controller based on background provided by FTTS [1]. Complexity calculation is based on air traffic controller tasks, defined depending on the traffic situation.

Intended Audience

There are two main groups of the intended audience:

- Experts from the related fields,
- The AISA consortium.

The development of the air traffic complexity module via AI SA deliverable (AISA D.3.3) is important for the consortium as:

- In the framework of WP3, it develops one of the ML modules for the AISA project.
- The document will provide direct input to the other technical work packages (WP3, WP4, WP5) and the related deliverables, by providing the conflict detection module developed based on ML techniques.

The document is also useful for external stakeholders, especially the following ones:

- Air Traffic Management (ATM) system developers who would like to understand how AI, and particularly ML techniques, can be integrated into ATM,
- ATM experts conducting related research.

General automation and AI experts would like to see the possible use of AI in a new domain.

Associated documentation

SESAR JU, “AISA Grant Agreement No 892618.”

AISA, “AISA D2.1 Concept of Operations for AI Situational Awareness System”

AISA, “AISA D2.2 Requirements for automation of monitoring tasks via AI SA”

Terminology

Following table lists the abbreviations used in this document.

Abbreviation	Description
AB	Advisory Board
ADS-B	Automatic Dependent Surveillance-Broadcast
AISA	AI Situational Awareness

ATC	Air Traffic Control
ATCO	Air Traffic Controller
ATM	Air Traffic Management
CA	Consortium Agreement
CLFL	Cleared Flight Level
CUFL	Current Flight Level
EFL	Exit Flight Level
GA	Grant Agreement
ML	Machine Learning
SESAR JU	Single European Sky ATM Research Joint Undertaking
WP	Work Package

Table of Contents

List of Figures

1 Introduction

The air traffic sector has experienced significant growth in the period from 2013 to 2018 [2]. Even though the growth is a significant driver of air traffic development, it can also carry negative aspects such as airspace congestion, high traffic density and flight delays with further consequences such as increased flight costs and environmental impact.

Further growth was forecast for 2019 and subsequent years [3], but the COVID-19 pandemic has somewhat slowed it. Nevertheless, those problems are set to reappear with the return to pre-pandemic numbers and could create complex air traffic scenarios that present safety hazards if air traffic controllers (ATCOs) are unable to resolve them.

Meeting the traffic demand means ensuring adequate airspace capacity so safe and efficient air traffic can be conducted. Airspace capacity and ATCO workload are linked, so one of the main focuses for solving both problems should be the air traffic complexity.

1.1 Air traffic complexity

The topic of air traffic complexity has been widely researched since the very beginnings of modern air traffic control, with first references to complexity research dating back to the 1960s [4]. A definition of complexity must first be established, which is often a difficult task in and of itself.

Mogford, et al. [5] offered a literature review of ATC complexity and ATCO workload research. ATC complexity is referred to as the effect that the airspace and the air traffic inside the airspace have on the ATCO. Airspace effects are stated to be separate from traffic effects, but it is deemed not useful to separate them when discussing ATC complexity.

The combined effect on the ATCO offers a further conclusion that ATC complexity generates controller workload. However, complexity is not the only relevant factor so the connection between the two is not straightforward – mediating factors such as cognitive strategies, equipment quality and individual ATCO characteristics must also be taken into account.

Hilburn, et al. [6] agreed that no complexity indicator can be examined in isolation and that complexity indicator interactions might not be linear. Furthermore, the connection between complexity and workload is identified to be complex with many mediating factors. The importance of individual and general human cognitive characteristics is highlighted as an important but not sufficiently explored issue.

Regarding specific complexity indicators, even traffic density, which was accepted as the most important traffic characteristic in air traffic complexity, is not accepted as the best complexity factor. This may be due to different density definitions, traffic characteristic interactions etc. Ideally, a complexity indicator would be independent of airspace and controller-specific factors.

Air traffic complexity can be defined as the difficulty of controlling a certain air traffic situation [7]. This definition recognizes two important elements highlighted in previous research – the human element and the traffic situation. All ATCOs are not the same and do not work in identical circumstances; similarly, traffic situations differ from each other based on many different characteristics.

1.2 Air traffic complexity assessment

Air traffic complexity assessment aims to award a score on a predetermined scale to any air traffic situation. The motivation for this complexity assessment model was solving several problems recognized in existing complexity assessment models, such as inadequate complexity indicators, airspace binding, complexity assessment inconsistency and others [8].

Inadequate complexity indicators encompass models where the indicators used to determine complexity were limited to the airspace used, difficult to implement or lacking connection to air traffic complexity.

Models with airspace bounding problems are ones that cannot be easily applied to airspaces other than the one used during model development. This might be due to the complexity indicators, model development methods or model characteristics.

Complexity assessment inconsistency appears when air traffic controllers tend to give the same complexity grade to air traffic situations of different complexity levels [9].

2 Air traffic complexity model

Each air traffic situation is made up of two major components – the airspace and the air traffic. The following sub-chapters describe the necessary model inputs and calculations, followed by Python implementation descriptions. All coding was done in the Python programming language. Anaconda software was used to create a programming environment with all packages required for certain code components to operate correctly.

2.1 Python packages and modules

Modules are code segments that are separated from the main code to simplify it and enable reusability. Several modules may be connected into a package, where the relationship is similar (but not identical) to that of a folder and files. Modules are usually connected into packages based on the type of problem they solve.

Several tasks in the model were solved by use of premade modules and packages, added to the code when there was a need for certain functions. To use a module, it must be either partially or entirely imported. All modules were imported at the beginning of the code, followed by specific function imports, for easier review.

Examples of premade packages include:

- Mathematical packages (*Math*, *NumPy*, *Shapely*)
- Geographical packages (*GeoPandas*, *GeographicLib*)
- Other packages (*Pandas*, *Cartopy*)

Mathematical packages were used to enable the use of common mathematical constants (e.g., pi), functions (e.g., square root, sinus and cosine) and data structures (such as arrays), as well as geometric objects and related object manipulation functions. Geographical packages were used to enable the use of geospatial data and for solving geodesic problems, while “other” includes packages with various functionalities ranging from data import to plotting.

Some parts of the code were split off from the main code, again for the purposes of simplification and reusability. These custom modules were imported back into the main code after the premade modules, followed by specific functions or objects defined within them. Examples of custom modules are the conflict point distance calculator, trajectory and potential trajectory creation modules and others. Some custom modules also make use of premade modules, so each required premade module is imported at the beginning of the custom module.

2.2 Airspace

The airspace is a defined volume of air above a territory, with a border that can change depending on the height. Sectors are parts of a larger, national airspace that divide the larger airspace into smaller ones for the purposes of complexity reduction and capacity increase.

2.2.1 Base map

The construction of the base map included plotting the territory and then the airspace. *Matplotlib* package enables data visualization in Python. It contains *pyplot*, a collection of functions that make *Matplotlib* work like MATLAB software. First, a figure element was created, acting as a container for all plots. Then an axes element is added for actual data visualization. During the axes creation, a map projection must be selected and applied.

Map projection definitions can be found in the *Cartopy* package. Some available map projections are: Plate Carree, Mercator, Transverse Mercator, EuroPP. The chosen projection, Plate Carree, is associated with data defined in WGS84 and that data can be shown without any transformations. If another map projection is needed, the change should be defined in the axes call. All plots in the code must then be accompanied by the transformation from Plate Carree (or rather WGS84).

Cartopy also contains the "borders" feature, which is used to add country borderlines to the axes. The axes extent (border location) is then set to show only the area around the Swiss border.

All other plots generated by this code were set on this base map. Their purpose was to check the output of the various operations in the code.

2.2.2 Observed airspace

Most, but not all ATCO tasks are confined to aircraft inside the airspace border. For the purposes of this model, another border was constructed using a homothetic transformation. The second border enables the inclusion of flights which are located outside of the original border, but which may add to an ATCO's planer tasks. Aircraft outside of the outer border have no influence on the current traffic situation's complexity.

For the *development* of the airspace complexity model, a generic airspace was constructed. The airspace is hexagonal in shape, measures 100 NM diagonally and 10 000 ft vertically. This was done to prevent ATCO's familiarity with the previously learned experience on familiar airspace and to enable the use of the model on new airspace without the need for adaptation each time the airspace changes.

For the *application* of the model, Swiss airspace was used. Sector LSAZM567 is composed of three sectors – LSAZM5, LSAZM6 and LSAZM7. The first one ranges from FL 355 to FL 375, the second one ranges from FL 375 to FL 385, and the last sector includes all flight levels above FL 385, respectively.

For the creation of the airspace and expanded airspace, several inputs are required. The first input are the coordinates of border points, acquired by exporting relevant data from NEST software. The data is in the form of airblock boundary point coordinates and so must be converted into the airspace border by choosing appropriate points and sorting them.

Another required input is the homothetic transformation function, applied to all boundary points from the airspace centroid. The function parameter is the expansion factor, which has been set to 1.5 for the needs of this model.

Forming of the airspace was done in a custom module called “sector”. The first part of the module contains all airblock point coordinates. These are separated into lists containing point longitudes and latitudes. References to these lists are made in the second part of the module, which contains the call to a *Shapely* Polygon object.

Calling the Polygon object requires point coordinates, but it is not enough to import all airblock points. Of those, only the outer points make up the airspace boundary. Furthermore, the relevant points must be ordered for the correct geometric object to be constructed. Both actions were done by hand and the result is a *Shapely* Polygon object called “polygon”.

For application on other airspaces, the same process must be repeated. To shorten the process, another way to procure the airspace boundary points should be found or a sorting application should be developed.

This module is imported into the main code, followed by the Polygon object within. Because a *Shapely* object cannot be directly plotted onto the base map, it must be transformed into an appropriate type such as a *GeoPandas* GeoSeries. The transformation into a *GeoPandas* object is accompanied by the creation of a new variable called “airspace”.

This transformation only defines how the airspace coordinates are stored, but not how they can be plotted. To enable plotting, the “set_crs” command is used – it specifies that all coordinate data within “airspace” is defined in a specific coordinate reference system. The standard CRS in ATC is WGS84 and its EPSG code – a standard way of shortening full CRS names – is 4326.

Any “child” object created from the airspace object will share its CRS, unless specified otherwise. This simplifies the creation of the extended airspace to creating a new variable called “airspace_extended”, using the “scale” function and passing appropriate arguments. As previously stated, the magnification factor is 1.5, so that value is passed for all three dimensions.

Both airspace boundaries can now be plotted onto the base map, which is done with arguments detailing the correct axes and desired face- and edge-colors.

2.3 Traffic

Air traffic consists of all airborne aircraft in a certain airspace. In each moment, a flight has multiple characteristics which are being recorded; for use in this model, necessary information consists of: flight callsign, latitude, longitude, barometric altitude, velocity, heading, cleared and exit flight level.

The technology behind the data is called “Automatic Dependent Surveillance-Broadcast” or ADS-B, for short. It is “automatic” because there is no need for external input in its operation. Through this, flight information is periodically broadcast by the aircraft and can be recorded by using dedicated sensors.

One of the organizations that collects ADS-B data is the OpenSky Network [10], a non-profit located in Switzerland, which uses sensors set up by volunteers and other supporters. The raw and organized data is stored on their servers and access is provided for research purposes.

All necessary traffic data was available in the form of “state-vectors”, which collect the following flight information: timestamp, ICAO24 address, latitude, longitude, velocity, heading, vertical rate, callsign, on-ground status, alert/spi, squawk code, baroaltitude, geoaltitude, last position update, last contact timestamp and hour batch timestamp.

All information is presented using the International System of Units. The timestamps are UNIX timestamps, so they represent the number of seconds since 01/01/1970.

The data was further processed by project partners at TUBS. Since ADS-B data relies on aircraft broadcast quality and frequency and number of sensors, it was determined that there exist gaps and anomalies in the data. Flight trajectories (and all other flight characteristics) were extracted from raw data and exported to a similar format to be used as model inputs.

Air traffic data is imported from a Windows Excel file by using the *Pandas* package functionality. Excel file location, sheet name and target columns must all be specified. As the file formatting will be identical, target columns need not be changed when changing inputs for different air traffic situations. This is not true for file locations and sheet names, so these have been left for the users to define.

In the case of file locations, an example was provided to avoid confusion. File location means the full system file path, formatted as “D:\User\Folder\Datafolder\traffic_data.xlsx”. The name of the sheet should be identical to the sheet name in Excel, including capitalization.

Imported data is stored to a variable called “data” and its column names are specified. It is then immediately converted to a *GeoPandas* GeoDataFrame. This transformation requires the latitude and longitude columns to be specified – *Shapely* Points are automatically created in the ‘geometry’ column of the GeoDataFrame, next to the imported data. Additionally, a CRS is added to all values in the GeoDataFrame.

2.3.1 Secondary data calculation and transformation

Secondary data is the data created from Excel imports, calculated so it can be used later in the model. This includes direct horizontal trajectories, sector entry/exit points and flight trajectory intersections in the horizontal plane.

Direct horizontal trajectories are created from a flight’s current position and heading. First, a new point is created by choosing a distance and calculating the coordinates of a point which is at that distance from the starting position in a given heading. The distance was set to 400 km, as it was shown that that value fulfills certain requirements.

The requirements stem from calculations which need to be executed on these trajectories. The trajectories will first be used to determine if a flight intersects the observed airspace, so they must be long enough to be useful in those extreme cases of flights just entering the extended airspace and traveling the longest possible distance to the exit point. The trajectories will also be used to determine flight trajectory intersections, which may occur anywhere inside the extended airspace.

For these reasons, a sufficient distance was needed, and the value of 400 km was shown to be acceptable as it worked for all flights and did not introduce any downsides. Further experiments might be made to determine the lowest possible value of the offset distance.

Using a *GeoPandas* command, intersections of flight trajectories and the sector can be determined. The output of the function is not (as might be expected) in the form of points, but in the form of *Shapely* LineStrings. Those lines represent all parts of a trajectory that are inside the airspace.

All in all, four scenarios are possible when it comes to those LineStrings:

1) A flight is already inside the airspace at the observed moment and its trajectory crosses the airspace boundary only once. For these flights, the starting point of the LineString is the current flight position and the ending point is the airspace exit point.

2) A flight is already inside the airspace at the observed moment, but its trajectory crosses the airspace boundary multiple times. For these flights, LineString becomes a MultiLineString which contains several LineStrings – the first LineString ranges from the current flight position to the first exit point, the second LineString ranges from the re-entry point to the second exit point etc., until the final exit point is reached.

3) A flight is outside of the airspace at the observed moment and its trajectory crosses the airspace boundary only twice. For these flights, the starting point of the LineString is the airspace entry point and the LineString ending point is the airspace exit point.

4) A flight is outside of the airspace at the observed moment, but its trajectory crosses the airspace boundary more than 2 times. Again, the intersection LineString becomes a MultiLineString with each LineString representing a flight section inside the airspace.

The 1st and 3rd scenario have simple solutions – flights inside the airspace border have only an exit point, while flights outside the airspace have only an entry and exit point. For flights whose intersections are MultiLineStrings, a further examination on a case-to-case basis is needed to determine the correct points.

If a flight is inside the airspace, the end of the last LineString will always be chosen. If a flight is outside of the airspace, the length of the first LineString dictates the entry point. If the first LineString length is under a previously set limit, it might be ignored and the first point of the next LineString taken as the entry point. This decision stems from actual ATC procedures – a short excursion outside of the starting airspace (not the one being observed) will not prompt a transfer to the ATCO in the new sector.

All flights outside of the extended airspace boundary are removed during this step, together with those flights whose intersection output is empty – in other words, if there is no intersection between a flight's trajectory and the airspace.

Flights that are not removed have their closest intersection and exit point added to the GeoDataFrame for further use. Naturally, for flights inside the sector, these points are one and the same. They must still be stored as different values because certain tasks will specifically require the use of one of the aspects, either the exit point or closest intersection.

2.4 Task calculations

ATCO tasks are defined depending on the characteristics of an air traffic situation – they are not dependent on the individual controlling the traffic. Basing the complexity score on ATCO tasks makes the score more objective as opposed to using standard complexity indicators. Because they were

developed to be unaffected by any airspace and person controlling the traffic, the tasks are also applicable to other sectors and sector configurations.

2.4.1 Conflict definition

Since most tasks are related to the concept of "conflict", a comprehensive explanation is given here.

In air traffic, for the en-route flight phase, aircraft are required to maintain a minimum horizontal and vertical separation. Minimal horizontal separation value (S_{min}) is defined as a 5 NM radius around the aircraft, while the minimal vertical separation value (H_{min}) is 1000 ft above and below the aircraft. These requirements form a cylinder around the aircraft – the radius of the cylinder is 5 NM and the height is 2000 ft, with the aircraft in the center.

If another aircraft flight path crosses the cylinder boundary, those aircraft are in a state of conflict. Similarly, if another aircraft *potential* flight path could violate the cylinder boundary, those flights are in a state of potential conflict.

For the purposes of this model, minimum vertical separation value was left unchanged, but horizontal values were changed to better suit the methodology. Since ATCOs were given static images of traffic situations, the higher values enabled easier conflict identification. The use of the extended airspace also plays a role, so special values were awarded for flights in that section.

New horizontal separation values are 10 NM for flights inside the airspace (denoted by S_{min}) and 15 NM for flights outside of the airspace (denoted by S'_{min}).

2.4.2 Flight trajectory intersections

Two aircraft with intersecting trajectories share an intersection point *C*, towards which their trajectories are converging. Depending on the flights' speed and initial positions, there are two possible scenarios.

1) One of the flights reaches and passes the intersection points well before the other. The trajectories diverge after the intersection point, so horizontal separation loss does not occur.

2) The flights converge towards the intersection point and, at some moment, their distance falls under the minimum prescribed norm. The flights continue along their trajectories which start diverging after the conflict point, so the distance again rises above the minimum prescribed norm at another point in time. These times, beginning and end of horizontal separation loss, are called critical times *tt1* and *tt2,* respectively.

As with the intersections of flight trajectories and the airspace, intersections of flight trajectories can be determined by using the created direct horizontal trajectories.

The process involves choosing a flight trajectory ("main trajectory") and using a loop to check for intersections with all other trajectories, one by one. The check is *not* performed when the loop tries to compare a flight's trajectory with itself. The output of this process is a single intersection point or an empty geometry (which means that there is no intersection between two trajectories). The trajectory intersection point is associated with a pair name – callsigns of both aircraft separated by a hyphen.

The callsign of the aircraft whose trajectory is the current "main trajectory" is placed before the hyphen, followed by the callsign of the aircraft whose trajectory is currently being checked.

Checking for horizontal plane intersections between a flight and all other flights means there are duplicates that are created in the process. In other words, the intersection between Flight 1 and Flight 2 is seen as being different from the intersection between Flight 2 and Flight 1. It is also important to note that there are intersections outside of the extended airspace border. Both the duplicates and non-relevant intersections are deleted.

2.4.3 Horizontal separation violations

For those flight pairs whose trajectories were found to intersect in the horizontal plane, the *critical times* must be determined. Critical time *tt1* represents the first and critical time *tt2* represents the last moment when the flights are in the state of loss of horizontal separation.

At critical time *tt1*, the aircraft are usually separated by $2*S_{min}$ or $2*S'_{min}$. At the middle of that distance is the conflict point T_c. The distances from initial aircraft positions to the conflict point are calculated and labeled d_1 and d_2, where d_1 is always the shorter of the pair. The aircraft which is at distance d_1 from the conflict point is denoted as A_i, while the other is A_j. Determining *tt1* and *tt2* also limits further calculations to that time interval.

Only intersecting flight pairs are checked for horizontal separation loss. This is done by projecting their points along their trajectories. A loop was formed that considers the next hour of flight, in seconds. For each second, the code uses aircraft speed to calculate the distance flown from the initial moment/position. The initial position, flown distance and heading are used as inputs for solving the direct geodesic problem – the output is the new position of the aircraft at the observed time.

This is done for both aircraft in a pair simultaneously, so their new positions can be compared and the distance between them calculated by solving the inverse geodesic problem. Conditional loops then check the distance value and store the first occurrence of horizontal separation loss as well as the onset of horizontal separation restoration. Pairs in which horizontal separation loss occurs are stored to a list so they can later be checked for vertical separation loss, while the rest are simply skipped.

At the moment of horizontal separation loss, an additional value is calculated and stored – the position of conflict point T_c. By definition, conflict point T_c is at half length of the distance between the aircraft when the conflict occurs for the first time.

Distances from the initial aircraft positions to the conflict point are also calculated because they determine which of the aircraft is designated A_i (with the other one being A_j). The implementation of A_i/A_j designation is solved by assigning the A_i aircraft's callsign to the first place in the pair's name.

Some aircraft pairs are in the state of conflict at the initial moment, t=0. Previous aircraft position data is not known, so that moment is taken as the beginning of the conflict and *tt1* is equal to 0. The conflict point T_c is at half length of the distance between the aircraft at tt1, which sets it exactly between the initial aircraft positions. Attempting to calculate d_1 and d_2, distances from aircraft initial positions to the conflict point, yields that d_1 and d_2 are equal. In this case, it is not important which aircraft is A_i and which is A_j, so the aircraft pair name is not changed – the aircraft whose callsign is first in the pair name will be treated as A_i, and the other aircraft will be A_j.

If a flight pair has no critical times *tt1* and *tt2*, the flight pair name is not stored for further calculations.

The next step is determining the vertical movements of the aircraft, first on their actual trajectories and then on their potential trajectories.

2.4.4 Vertical separation violations

The trajectory of an aircraft has the following vertical profile – the aircraft starts at the current flight level (CUFL), climbs immediately to its cleared flight level (CLFL), then flies at the cleared flight level until the moment it must start climbing/descending towards the exit flight level (EFL) to reach it at the airspace exit point. Only if it is proven that a flight's cylindrical boundary is not violated on that trajectory is it necessary to check if the boundary is violated on one of the potential trajectories.

The potential trajectories of an aircraft are a collection of similar trajectories, all originating at the CUFL and ending at the EFL. They are necessary because an ATCO may instruct the pilot to begin ascending/descending from the CLFL to the EFL at any time, as long as the aircraft reaches the EFL before exiting the sector.

Each trajectory starts the ascent/descent from CUFL towards the CLFL immediately, but after reaching it they begin the ascent/descent towards the EFL at different moments. The last potential trajectory starts the ascent/descent at the last possible moment where it can reach the EFL at the airspace exit point.

Flight pairs that experience horizontal separation loss have been, as was previously mentioned, stored to a list. That list acts as a basis for vertical separation loss checks, so they need not be performed on all flight pairs with intersecting trajectories. The vertical separation loss check functions similarly to the horizontal separation loss check, except for the trajectory creation that must be done beforehand.

First, a module named "conflict_trajectory" was made for the creation of conflict trajectories. It takes an aircraft's initial position, speed, altitude, EFL and CLFL, airspace exit point coordinates and aircraft pair critical times as inputs and produces a collection of altitudes that make up the aircraft trajectory. CLFL is marked as an optional input – if no value is provided, the code assumes the aircraft current altitude is the CLFL altitude.

To calculate aircraft altitude at different points in time, aircraft critical times must first be determined. This is done by calculating differences between aircraft flight levels and dividing them by the rate of climb/descent. For the purpose of brevity, "climb" will be used for all terms where "climb/descend" would be used, such as "rate of climb/descent" or "climbing/descending time". The correct term differs for all flights, depending on the relationship between a flight's flight levels.

Since the rate of climb varies depending on aircraft type and current altitude, a simplification was introduced by choosing a single value of 1000 ft/min for all calculations. That value was determined to be attainable by all observed aircraft. Dividing the altitude differences by the rate of climb results in times required to climb from CUFL to CLFL and CLFL to EFL.

The first critical time of an aircraft is the moment is reaches CLFL which, because the climb from CUFL to CLFL starts in t=0, is equal to the climbing time between those levels. If the aircraft immediately continues to EFL, the moment it reaches EFL is the second critical time for that aircraft. It is calculated by summing the two calculated climbing times.

If the aircraft does not continue to EFL but flies on CLFL until the last moment when it must begin its climb to EFL to reach it at the sector exit point, that last moment on CLFL is the third critical time of an aircraft. The fourth critical time is the moment when the aircraft reaches EFL at the sector exit point, so the third critical time is calculated by deducting the climbing time from CLFL to EFL from the fourth critical time.

The vertical profile of an aircraft is calculated by assigning the aircraft its current altitude in t=0, adding the rate of climb for every moment between t=0 and the first critical time, then assigning all points between the first and third critical time the CLFL altitude. When the third critical points is reached, the altitude is once again increased until the last critical time, after which the aircraft continues horizontal flight at the EFL altitude. This profile and all critical times which are between *tt1* and *tt2* are returned to the main code.

A list of combined critical times is formed – initially it includes only tt1 and tt2, and critical times of each aircraft which are between tt1 and tt2 are added to it. The times are then sorted in the ascending order. The code will check for vertical separation loss only at those critical times which are in the list.

At *tt1*, the code determines which aircraft is higher and which is lower. The altitude of the initially higher aircraft is then subtracted from the altitude of the initially lower aircraft, so the calculated altitude difference is initially negative. It is immediately checked if the value of the altitude difference is below the previously set limit – if it is, the aircraft are in conflict in *tt1*. If not, the next critical time in the list is checked – until all have been checked or the onset of vertical separation loss has been discovered.

The altitude difference value can reveal two things:

1. If it is determined that the value has fallen below the previously set limit, that would mean the vertical separation loss occurred at some point between the previous critical time and the one being checked currently.

2. If at some point it is determined that the value has become positive, that means the aircraft which was initially higher is now lower (and vice versa). A logical conclusion is that the aircraft had (at some point during the change) been at the exact same altitude and thus there was vertical separation loss.

In both cases, the period between the previous and current critical times is checked so the exact moment when the vertical separation loss starts can be determined. The flight pair name is stored to a previously initialized list for further processing. Flight pairs with no vertical separation loss are not skipped, as was the case when checking for horizontal separation loss, but are stored to another list so their potential trajectories can be checked for potential conflict.

A second module, “potential_conflict_trajectory”, contains the code for the creation of potential trajectories. The potential trajectories represent possible positions of an aircraft with two edge cases:

- The aircraft climbs directly from CUFL to EFL and maintains level flight at EFL,
- The aircraft follows the conflict trajectory – going from CUFL to CLFL, then maintaining level flight on CLFL until the last possible moment when it must start climbing towards EFL to reach it in the airspace exit point.

Those two trajectories diverge when the aircraft reaches CLFL and converge in the airspace exit point. The aircraft could potentially be located at any point in the space bounded by those trajectories, provided that the ATCO gives the aircraft the command to start climbing from CLFL to EFL at a specific moment. The check for potential conflicts functions similarly to the conflict check – aircraft altitudes were compared for critical times and, if vertical separation loss was detected, the period between the previous and current critical time was checked for the first instance of vertical separation loss.

In this case, aircraft altitudes could not be compared using just one calculation – the concept of potential trajectories means that the aircraft could be at any altitude within a *range*, for the same moment in time. That range was represented by its maximal and minimal values, which were then compared to the maximal and minimal values of the other aircraft. If any separation loss is detected, it signifies that there exist potential trajectories for the observed aircraft pair which, if flown, will mean those aircraft are in the state of potential conflict. All pairs found to have such potential trajectories are stored to a list for further classification.

2.4.5 Task definitions

The list of ATCO tasks (with related task codes) is as follows [1]:

1. Conflict resolution (Code: C)
2. Potential conflict resolution (Code: P)
3. Coordination of conflict resolution (Code: CC)
4. Coordination of potential conflict resolution (Code: CP)
5. Interactive conflict screening (Code: SI)
6. Potential interactive conflict screening (Code: SP)
7. Non-interactive conflict screening (Code: SN)
8. Initial call (Code: IC)
9. Frequency transfer (Code: FT)
10. Execution of requests (Code: ER)

The first two tasks deal with conflict situations described earlier. The next two tasks function the same way, but the horizontal separation values are changed to those for flights outside of the airspace (S'_{min} as opposed to S_{min}). Tasks 5-7 are related to tasks 1-4 and are only performed based on certain task occurrence.

In other words, task 5 is performed if there are tasks 1 and 3; task 6 is performed if there are tasks 2 and 4; task 7 is performed if none of the first four tasks are present. The reason for this is that tasks 5-7 provide information on which task the ATCO is doing. Even though the tasks may seem similar and ordered, they are not – e.g., the ATCO first notices a conflict as task 5, then resolves it through either task 1 or task 3.

Task 8 is performed for aircraft in the extended airspace, nearing the airspace border. If the aircraft is 20 NM or less from the observed airspace boundary, the task takes place. Similarly, task 9 is performed for aircraft inside the observed airspace but nearing its border – any aircraft 15 NM or less from the border triggers this task.

The last task, execution of request, is performed only if a flight's CUFL and EFL are not the same and there is a need for an altitude change.

2.4.6 Conflict classification

First four tasks are further categorized according to their parameters. The categories, which are evaluated in order, are as follows:

1) Track (3 values)
2) Distance between aircraft (6 values)
3) Speed comparison (3 values)
4) Convergence angle value (6 values)
5) First aircraft distance to conflict point (6 values)
6) Second aircraft distance to conflict point (6 values)
7) First aircraft freedom of movement – left side (2 values)
8) First aircraft freedom of movement – right side (2 values)
9) Second aircraft freedom of movement – left side (2 values)
10) Second aircraft freedom of movement – right side (2 values)
11) First aircraft freedom of movement – above (2 values)
12) First aircraft freedom of movement – below (2 values)
13) Second aircraft freedom of movement – above (2 values)
14) Second aircraft freedom of movement – below (2 values)
15) First aircraft exit distance (4 values)
16) Second aircraft exit distance (4 values)

Each classification adds its one-digit code to the final conflict task designation. All in all, there are $1.927544120276803 * 10^{27}$ task code possibilities.

A separate module was created for each classification, calculating a letter or digit code for further use. A value calculated inside a module can be returned to the main code via the *return* function. The module is called inside the main code and a free variable must be designated to accept the calculated value.

Afterwards, aircraft must be checked for tasks related to individual aircraft – initial call, frequency transfer and execution of requests. Those checks were divided into three separate modules.

2.4.6.1 Track

The first subclassification is the division depending on the flight angle between two aircraft in conflict. The categories, their codes, and corresponding angular difference values are:

- Same track – Code: S – angular difference of less than 45° and more than 315°
- Crossing track – Code: C – angular difference between 45° and 135° or between 225° and 315°
- Opposite track – Code: O – angular difference of more than 135° and less than 225°

Aircraft heading values range between 0° (designating geographic north) and 359°, expressed as decimal numbers with up to 13 decimal places. This is the convention adopted by the OpenSky Network and it is not equal to the ATM convention.

For calculation purposes, limits based on aircraft A_i heading were used. All in all, four limits were defined – one "lower" and one "higher" limit for "Same track" and "Opposite track", respectively. Any track not belonging to those categories belongs, by default, to the "Crossing track" category.

The approach of calculating category boundaries based on the A_i heading value requires five different conditional statements to be set. The first case deals with low values of A_i headings, ranging from 0° to 45° – all calculated boundaries are then under 360°, and there is no need for additional calculations.

As soon as the A_i heading value reaches 45° degrees, the value of one of the "Same track" borders reaches 360° and must be reverted to 0°. Thus, every time a value could reach 360°, a new conditional statement is set and all limit values over 360° are converted to be in (0°, 359°) range.

This module has 3 possible return values – S, C or O.

2.4.6.2 Distance between aircraft

The second classification divides conflicts depending on the distance between initial aircraft positions. All in all, there are six different distance categories (listed here with their respective codes):

- Between 0 and 10 NM (Code: 1)
- Between 10 and 20 NM (Code: 2)
- Between 20 and 30 NM (Code: 3)
- Between 30 and 50 NM (Code: 4)
- Between 50 and 80 NM (Code: 5)
- More than 80 NM (Code: 6)

Each category is checked by a separate conditional statement and the "code" variable is set to an appropriate value.

This module uses the *GeographicLib* package for solving geodesic problems.

2.4.6.3 Aircraft speed difference

Conflict classification based on aircraft speed difference checks the speeds of both aircraft in a conflict pair. The result belongs to one of three categories:

- Speed of aircraft A_i is higher than the speed of aircraft A_j (Code: 1)
- Speed of aircraft A_i is the same as the speed of aircraft A_j (Code: 2)
- Speed of aircraft A_i is less than the speed of aircraft A_j (Code: 3)

2.4.6.4 Converging angle

This module classifies the conflict based on the heading difference of aircraft trajectories, sorting it into one of six categories:

- Angle difference between 0° and 20° (Code: 1)
- Angle difference between 20° and 44° (Code: 2)
- Angle difference between 44° and 90° (Code: 3)
- Angle difference between 90° and 135° (Code: 4)
- Angle difference between 135° and 159° (Code: 5)
- Angle difference between 159° and 180° (Code: 6)

The heading difference is calculated and its value is checked. For difference values over 180°, the difference is subtracted from 360° to obtain a value between 0° and 180°. The result is then evaluated in conditional loops, in the same order as they are listed above.

2.4.6.5 Distance from aircraft to conflict point

The fifth and sixth classifications divide conflicts depending on the distance between initial aircraft positions and the conflict point. This is done first for aircraft A_i, then for aircraft A_j. The checks are identical, and they sort the conflict into one of six different distance categories (listed here with their respective codes):

- Distance between 0 and 10 NM (Code: 1)
- Distance between 10 and 20 NM (Code: 2)
- Distance between 20 and 30 NM (Code: 3)
- Distance between 30 and 50 NM (Code: 4)
- Distance between 50 and 80 NM (Code: 5)
- Distance of more than 80 NM (Code: 6)

Each category is checked by a separate conditional statement and the “code” variable is set to an appropriate value. The value is then returned to the main code.

This module uses the *GeographicLib* package for solving geodesic problems.

Even though the distances to the conflict point were already calculated during the horizontal separation loss calculation, they are calculated again at this point. The reason is that the existence of horizontal separation loss does not guarantee that two aircraft are in conflict – storing values for each of those aircraft pairs to recall them during this step is more complicated than using a separate module that also performs the calculation.

2.4.6.6 Freedom of movement calculations

Freedom of movement calculations are done on both aircraft in a pair to check which trajectory change can be performed by the aircraft. These include checks for freedom of movement in the horizontal and vertical plane. Order of operations is:

- Horizontal freedom of movement check to the left of aircraft A_i
- Horizontal freedom of movement check to the right of aircraft A_i
- Horizontal freedom of movement check to the left of aircraft A_j
- Horizontal freedom of movement check to the right of aircraft A_j

- Vertical freedom of movement check above aircraft A_i
- Vertical freedom of movement check below aircraft A_i
- Vertical freedom of movement check above aircraft A_j
- Vertical freedom of movement check below aircraft A_j

In practice, aircraft heading is never changed by less than 5°. Likewise, changes above 45° are extremely rare in en-route operations. Thus, values of 5° and 45° were taken as range limits, with a step of 5°. For horizontal freedom of movement checks, aircraft heading is changed by each value in the range, first by subtraction (to calculate new headings to the left) and then by addition (to calculate headings to the right).

The vertical profile of the trajectory – climbing from CUFL to CLFL, flight at CLFL and climbing to EFL to reach it at the airspace exit point – is not changed for horizontal freedom of movement checks. Potential trajectories are not created and potential conflict is not checked at this time; only new *conflict* is relevant.

Each new trajectory creates a new traffic situation in which the observed aircraft is checked for conflicts. If the aircraft, which is flying along the new trajectory, has no conflicts with other aircraft, it is deemed to be free of traffic. If the current conflict persists or a new conflict emerges, the aircraft is not free of traffic.

Vertical freedom of movement checks represents new trajectories which have had their vertical profiles modified. The observed aircraft climbs to a newly designated EFL and maintains level flight. This new trajectory is then checked for conflicts in the same manner as for the horizontal freedom of movement checks.

New flight levels are chosen depending on the observed sectors – a flight in a sector ranging from FL 320 to FL 355 would be checked for flight levels between those, with a step of 10. In the case of LSAZM567, those are flight levels ranging from FL 355 to FL 999. Checking all those flight levels is obviously unnecessary, as not many aircraft are able to utilize flight levels above 430.

The output of this module is binary, returning 1 if an aircraft is free of traffic on all available trajectories in a given category and 0 if it is not.

2.4.6.7 Distance to exit

This module calculates the distance from initial aircraft positions to their respective airspace exit points. The *GeographicLib* package is used for calculating the distance.

It is performed for aircraft A_i and A_j separately, and sorts the conflict into one of 4 distance categories (listed here with their codes):

- 0 – 15 NM (Code: 1)
- 16 – 30 NM (Code: 2)
- 30 – 45 (Code: 3)
- 45+ NM (Code: 4)

2.4.6.8 Initial call

The initial call task exists for aircraft that are entering the airspace border and are within a prescribed distance from the border. The distance value is set to 20 NM. This task is binary – for a given aircraft,

the task either does or does not exist. For that reason, the module returns *None* (in the case the task does not exist) or '*IC*' (if the task does exist).

The *GeographicLib* is used for calculating the distance from the initial position to the aircraft entry point.

2.4.6.9 Frequency transfer

This module is a counterpart of the previous one. The module is used only on flights which are inside the airspace, and it checks if they are within a prescribed distance from their exit point. The module returns *None* (in case the aircraft is more than 15 NM from the exit point) or '*FT*' (if the aircraft is 15 or less NM from the exit point).

Same as the previous module, the *GeographicLib* is used for the distance calculation.

2.4.6.10 Execution of request

Every flight level change is an ATCO task – thus the existence of a difference between CFL and EFL points to the existence of an execution of request task.

This module is used on all aircraft and checks if there is a difference between those flight levels. If a cleared flight level is not specified, CUFL is calculated from the altitude and used for this calculation. This is done by converting the barometric altitude from meters to feet, transforming it into a flight level by dividing by 100 and then rounding the number to an integer. The conversion to meters and subsequent rounding might change the current flight level slightly, so the calculation treats flight levels directly above and below EFL as being the same as EFL.

The output of the module is binary – *None* if the flight level difference does not exist, '*ER*' if it does exist.

2.4.7 Task code assembly

The task code calculation is automated to determine tasks for each aircraft (in case of tasks 8-10) and between each aircraft (for all other tasks). For a given traffic situation with N aircraft, a matrix of size N x N is calculated which contains all calculated task codes, and it is written in a form Δ_{ij}.

The appearance of each task can be written as a binary state – either 0 or 1. Each of the first seven tasks as well as subsequent conflict classifications were assigned a variable B_i and all other tasks were assigned a variable C_i. B_i and C_i variables are collected as a set Ω and describe the conflict between aircraft A_i and A_j in a manner similar to the conflict task code.

For the calculation of actual task codes, four separate loops were made in the main code.

The first loop creates a list of all possible aircraft pair combinations, without duplicates. All pairs will have at least one task assigned to them, regardless of the existence of conflict/potential conflict. That task is the traffic screening task, and may attain one of three values:

- Interactive conflict screening (Code: SI)
- Potentially interactive conflict screening (Code: SP)
- Non-interactive conflict screening (Code: SN)

For aircraft pairs in a state of conflict, task SI and a conflict task code are assigned within the second loop. Each pair is removed from the list of all pairs, so it does not appear during later task SN assignment. For aircraft pairs in a state of potential conflict, the same is done with task SI being replaced by task SP.

A conflict task code consists of all previously described category codes, connected into a single string. An example of a full conflict task code, with spaces added for clarity, is:

"C C 4 3 3 3 3 2 2 2 2 2 2 2 1 4 4"

where the first two conflict code parts designate the conflict/potential conflict type and track category, the third designates distance between aircraft category and so on. Once all aircraft pairs in a state of conflict/potential conflict have had their conflict task code calculated and they have been removed from the list of all pairs, the remaining pairs can be assigned the SN task.

After all pairs have been processed, each aircraft is checked for individual tasks – the initial call, frequency transfer and execution of requests. The initial call is done only on aircraft outside of the airspace and the frequency transfer task is only done on aircraft inside the airspace. For both tasks, task codes are set to *None* for aircraft which are not checked.

All aircraft in a traffic situation are checked for the execution of request task. All three "individual" task codes are then added to a string, separated by commas.

3 Machine learning

3.1 Data gathering

In the data gathering stage, ATCOs were given two paper static images representing traffic situations in a generic airspace. Their task was to compare the two situations and assess which situation is more complex. Afterwards, they were instructed to give a complexity score for each traffic situations, with 1 being the lowest and 5 being the highest complexity score. Additionally, ATCOs were asked to determine a point at which they believed the sector should be divided in order to reduce the overall air traffic complexity.

120 unique air traffic situations were used, divided into 6 groups. Each group consisted of 30 traffic situations, with 18 situations unique to that group and 12 situations which were repeated in all groups. This was done in order to better determine ATCO complexity assessments between themselves. Each group of air traffic situations was evaluated by 3 ATCOs, so a total of 18 ATCOs was tested [1].

The Merge sort algorithm was used for ranking the traffic situations. This way, two traffic situations cannot have the same complexity score, so inconsistency in ATCO scoring is eliminated. The output was a clear rank from the lowest complexity situation (w^1) to the highest (w^M). Controllers' complexity scoring and grading was used to assign linearly interpolated grades η^l to each air traffic situation w^l for each ATCO [1].

3.2 Statistical models

Next, a statistical model was trained from chosen exploratory variables (ATCO tasks) and target variables η^{l}, which gives a relationship between the exploratory variables and the complexity for each air traffic situation. If a model were given the variables for another, unseen traffic situation, it could determine the complexity of that situation [1].

A clear overview of the process is depicted in Figure 1.

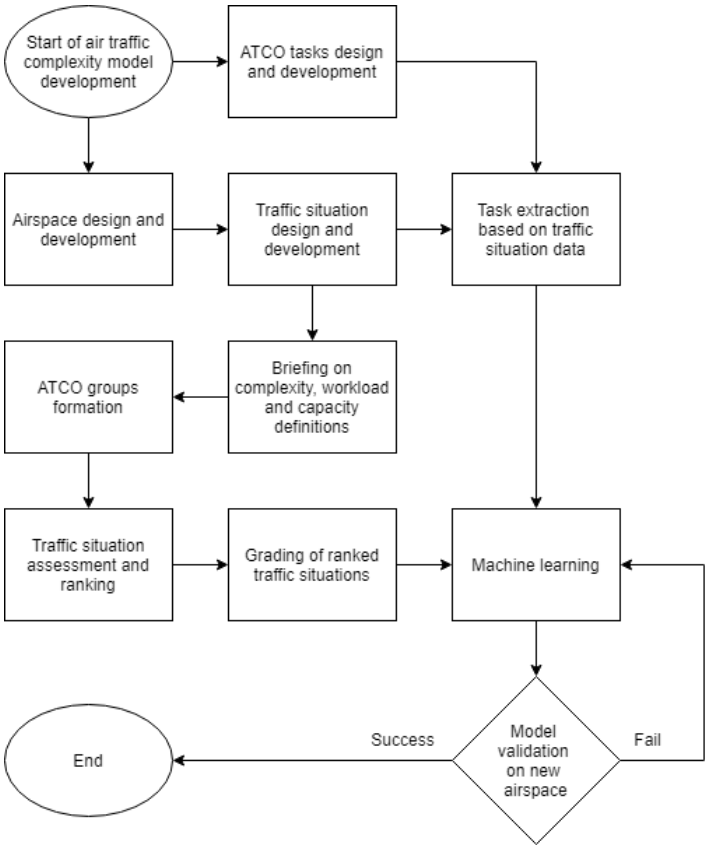

Figure 1 Methodology and plan of research (Adapted from [1])

A linear statistical model, Bayesian Ridge Regression, was used. While a nonlinear statistical model could have been chosen, the advantage of choosing a linear model is that there is a direct relationship between the model's coefficients and the contribution that each exploratory variable (in this case, each ATCO task) contributes to the overall air traffic complexity – which means that coefficients can be used as a measure of ATCO task complexity.

All statistical models can also be used to determine the contribution of each individual aircraft to the complexity of a given air traffic situation. A complexity model is trained by using "training data", but then a copy of the situation is created from which the selected aircraft is removed. The developed model is then used on this new traffic situation – the complexity difference between the two air traffic situations is the measure of that aircraft's contribution to the complexity. The same procedure can be applied to all aircraft in a given air traffic situation.

3.3 Exploratory feature analysis

For determining key features of traffic situations, an exploratory feature analysis was performed. Histograms of feature value distributions showed that almost all ATCO tasks were represented to the same extent, with only one task not appearing in any of the 120 situations and several appearing in more than 60 situations [1].

A correlation matrix was created to determine if there are correlations between the features themselves. Some features were found to correlate highly (such as the number of aircraft and aircraft screening) and some were found to have a weak negative correlation [1].

3.3.1 Feature construction

From all features, a feature selection for the model must be determined. Each air traffic situation consists of several aircraft and their mutual positions – ATCO tasks correspond to the tasks that need to be performed in order to assess the situation and resolve any conflicts. The tasks are independent of the controller – every controller is aware of the set of tasks and may choose their own way of resolving the air traffic situation. Each feature is effectively the number of times that a particular task appears in a given situation.

The machine learning section overview is shown on Figure 2.

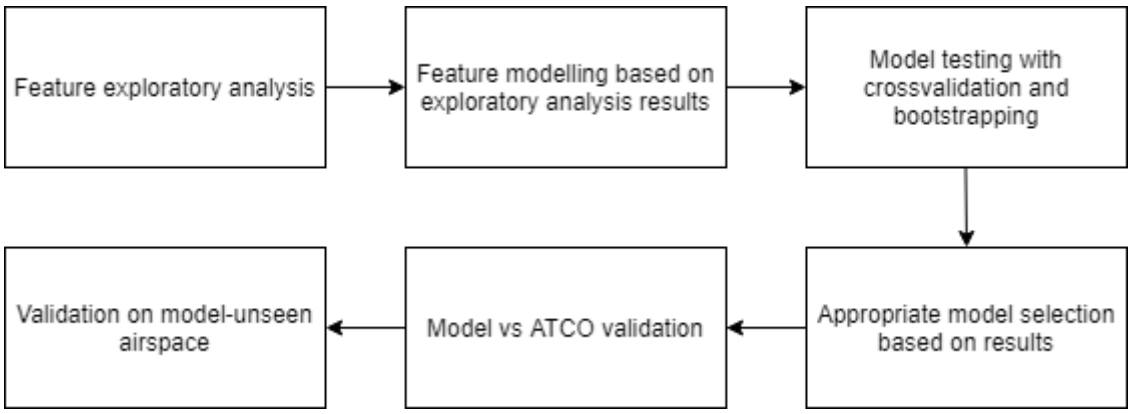

Figure 2 Machine learning section execution plan [1]

The following set Ω of 74 features was chosen:

$B_1 - B_4$ (C, P, CC, CP) are combined with $B_5 - B_7$ (C, O, S) to get 12 task types (CCC … PS), to which another 6 are added – screening tasks $B_{51} - B_{53}$ and individual aircraft tasks $C_1 - C_3$ (ER, FT, IC). When adding conflict subclassifications, it should be noted that variables $B_{35} - B_{42}$ are counted twice because the values 0 and 1 represent different tasks. That makes for a total of 69 individual tasks [1].

Additional features include the aircraft count (with N^{in} being the number of aircraft about to enter the airspace and N^{out} being the number of aircraft about to exit the airspace) and the number of possible aircraft pairs for screening [1].

As for target variables, two different sets were used. The first set are the results of air traffic situation pair comparisons. The second set are ATCO complexity grades, which come in two versions – original

discrete grades ranging from 1 to 5 and linearly interpolated grades (also on a scale from 1 to 5) which were calculated based on how the controllers ranked a certain traffic situation within each grade.

There are a total of 540 situation grades – those situations which were identical in all groups were graded by all ATCOs, while the traffic situations unique to a certain group were graded by only those controllers assessing that group. Regardless of the number of controllers that graded a certain situation, the mean value of all grades was taken to obtain a single grade for each situation.

The difference between using mean values of discrete controller grades and mean values interpolated grades is not significant, but it is clear from Figure 3 that interpolated grades contain more information than the other option. That makes them a better choice for use as target variables in the regression model [1].

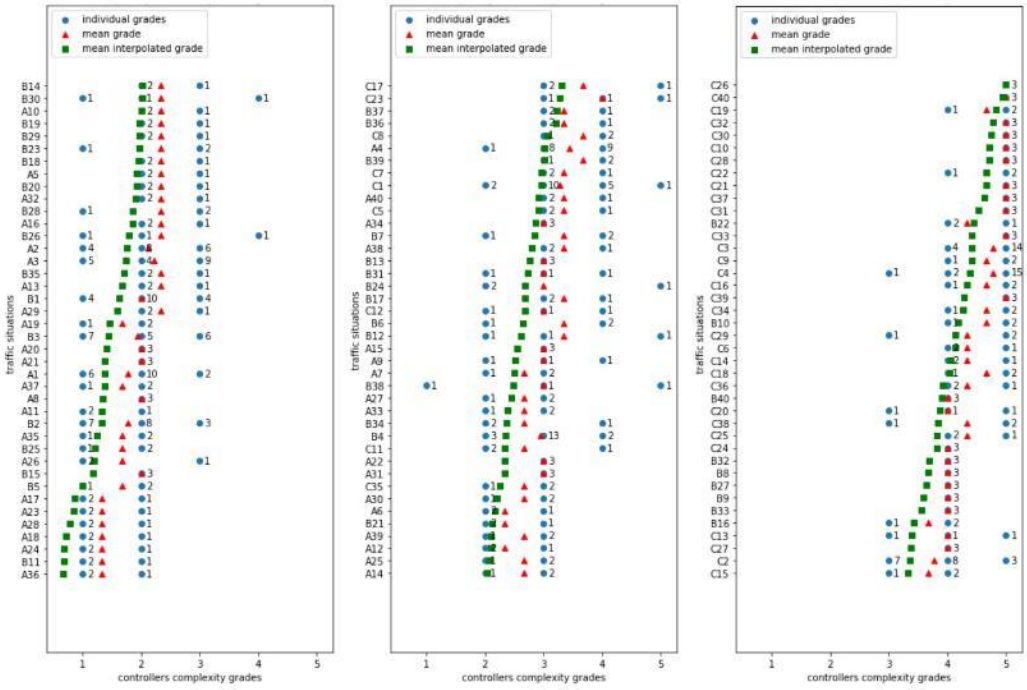

Figure 3 ATCO complexity scores for the original 120 traffic situations [1]

3.4 Regression models

Two regression model types were tested, one for each target variable set [1]:

- Logistic regression models which work on binary comparison target variable data
- Linear regression models which learn from ATCO complexity scores.

The logistic regression model was soon abandoned because the linear regression models showed significantly better results. Of the linear regression models, Bayesian Ridge Regression was chosen. The goal was to connect sets of binary states Ω (variables B_i and variables C_i) from Δ_{ij}, N^{in} and N^{out} to the interpolated score estimates η^{l}. This would enable the creation of a linear model η_{real} with weight coefficients β_z, $\gamma_{z'}$, $\alpha_{z''}$, where $\partial_{i,j} = 1$ for $i = j$ for Δ_{ij} and $\partial_{i,j} = 0$ for $i \neq j$ for Δ_{ij} [1]:

$$\eta_{real} = \sum_{j \geq i \geq 1}^{N^l} [\Delta_{ij}]_l + F_l(N^{in}, N^{out}) \quad (1)$$

Where:

$$[\Delta_{ij}]_l = \sum_{z \in \Omega} \beta_z (B_z)_{ij} (1 - \partial_{i,j}) + \sum_{z'=1}^{3} \gamma_{z'} (C_{z'})_{ii} \partial_{i,j} \quad (2)$$

$$F_l(N^{in}, N^{out}) = \alpha_1 N_l^{in} + \alpha_2 N_l^{out} + \alpha_3 \left(\frac{N_l^{in}(N_l^{in} - 1)}{2} \right) + \alpha_4 \left(\frac{N_l^{out}(N_l^{out} - 1)}{2} \right) + \alpha_5 \left(\frac{(N_l^{in} + N_l^{in})\left((N_l^{in} + N_l^{in}) - 1\right)}{2} \right) \quad (3)$$

Calculated coefficients were standardized prior to applying to the formula (1). Regression coefficients will not be published because the model can use any set of Ω features required for air traffic complexity calculations.

Defining the model enables the use of a recursive forward feature selection. This is the process of determining which feature combination gives the best correlation score. Since some features have a high correlation rate, it is possible to eliminate certain features while keeping good complexity estimates. Recursive forward feature selection also showed that the best features were the ones somehow connected to aircraft counts.

The correlation between the mean interpolated grade and some feature sets reached high values with just six features, most of which were related to either the aircraft count, aircraft freedom of movement or opposite track conflicts. 20 different feature sets were constructed for use in logistic regression and linear regression models – 3 sets for the former and 17 for the latter models. Sets used different combinations of features from the original set of 69 task types and the additional information (aircraft counts, pairings).

The bootstrap method was applied to all 17 feature sets – a data set (features and target variables) was repeated 300 times and randomly divided into a training set (80%) and a testing set (Figure 4). After training on the 80% set, it was tested on the remaining 20% and the Pearson correlation coefficient was calculated.

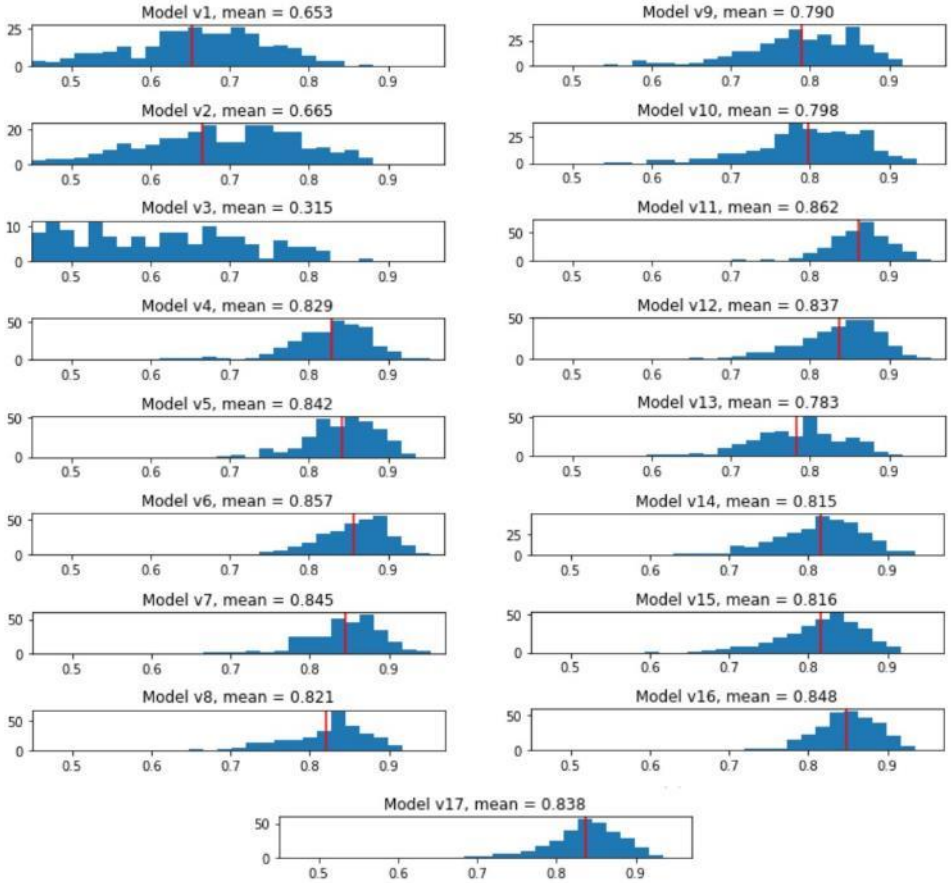

Figure 4 Bootstrap aggregating results for the selected feature sets [1]

Out of 17 feature sets, 2 sets showed the best (and mutually similar) correlation results. The features in the sets were analyzed and a decision was made to use the set whose correlation score was slightly lower. This decision was shown as justified in the validation phase when another comparison between those sets and models was conducted.

3.5 Model validation

Two model validations were performed. Model versus controller validation looks at how the model complexity score and controller grades differ from the mean grade calculated across all controllers for a specific traffic situation. New airspace validation includes applying the model to a new airspace, where traffic situation in that airspace were not used in the training of the statistical model.

New airspace validation results are shown here. Random subsets were used for building separate statistical models, with each subset consisting of 6 traffic situations (which is the same as the number of new airspace traffic situations that ATCOs had to grade) (Figure 5). Two approaches to data results were taken:

- "Average" approach – takes into account the *average* difference from the mean grade across all 30 situations a particular ATCO graded

- "Extreme" approach – takes into account the extreme difference (maximum absolute difference) from the mean grade across all 30 traffic situations a particular ATCO graded.

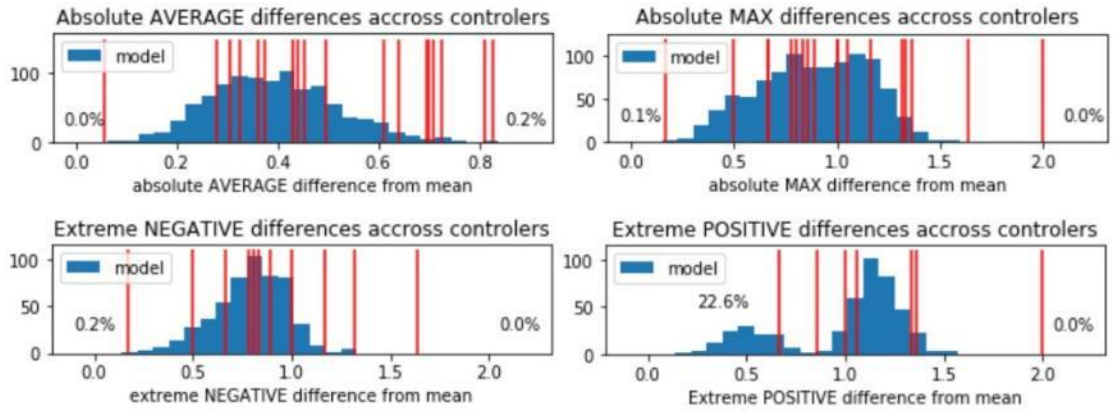

Figure 5 Chosen model validation on a new airspace compared to the ATCO deviation to mean interpolated scores [1]

Figure 5 shows both error from the model and the air traffic controllers for a specific air traffic situation deviating from the mean interpolated complexity grade. It can be seen from the model distribution on the new airspace, when tested separately 1000 times, that the overall model distribution makes less error than the outmost extreme cases from the air traffic controllers that are depicted with the red vertical lines.

Pearson and Spearman correlations of model complexity versus ATCO complexity estimations are shown in Figure 6. The model was trained on 100% original data set and tested on a random 20% of the new, validation data set – this was repeated 600 times [1].

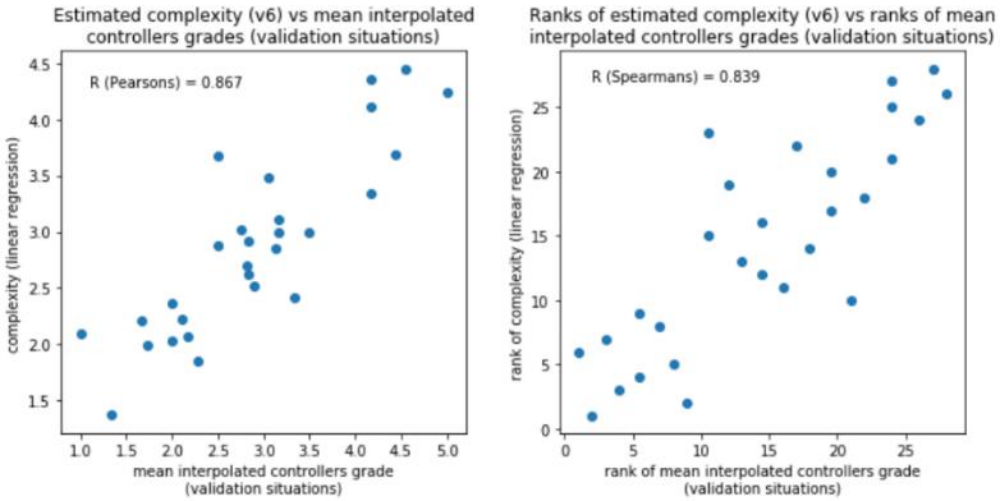

Figure 6 Pearson and Spearman correlation of model complexity compared to the ATCO complexity estimation for the new, validation traffic situations [1]

4 Conclusion

The creation of this model was motivated by the previous solutions to the problem of adequate complexity score determination in ATC [8]. Previous research offered various approaches to the problem of air traffic complexity by both including and excluding the ATCOs perspective of the air traffic complexity. This model is proof that air traffic complexity can be determined based on ATCO tasks, which makes it person and airspace non-dependable.

One of the advantages of this model is the fact that it is an LOAA (Learn Once Apply Anywhere) model [1]. This claim was confirmed through validation on model-unseen, realistic airspace where the model calculated air traffic complexity with errors below ATCO assessment levels.

As was previously stated, the contribution of an individual ATCO task to the overall complexity can be determined from the coefficients of the linear regression model. With the increase in automation in ATC, it will be possible to determine how the automation of specific tasks lowers the ATCO workload and air traffic situation complexity. The model can also identify the most complex aircraft (whose presence causes the highest complexity rise) in a given traffic situation and check how parameter changes for that aircraft affect the overall complexity.

Further automation will see this traffic complexity model applied to real-time air traffic data or predicted trajectory data, as opposed to historical data. Having the model read from and, in turn, populate a knowledge graph would improve individual and shared situational awareness of ATC team members.

5 References

[1] B. Antulov-Fantulin, *Air traffic complexity model based on air traffic controller tasks,* Zagreb, 2020.

[2] Performance Review Comission, "Performance Review Report 2017," EUROCONTROL, 2017.

[3] Performance Review Comission, "Performance Review Report 2019," EUROCONTROL, 2019.

[4] C. Davis, J. Danaher and M. Fischl, "The influence of selected sector characteristics upon ARTCC controller activities," The Matrix Corporation, Arlington, 1963.

[5] R. Mogford, J. A. Guttman, S. Morrow and P. Kopardekar, "The Complexity Construct in Air Traffic Control: A Review and Synthesis of the Literature," CTA Incorporated, McKee City, 1995.

[6] B. Hilburn, "Cognitive Complexity in air traffic control: a literature review," Center for Human Performance Research, 2004.

[7] T. Radišić, P. Andraši, D. Novak, B. Juričić and B. Antulov-Fantulin, "Air Traffic Complexity as a Source of Risk in ATM," in *Risk Assessment in Air Traffic Management*, vol. 31, London, IntechOpen, 2020, pp. 56-83.

[8] B. Antulov-Fantulin, B. Juričić, T. Radišić and C. Çetek, "Determining Air Traffic Complexity - Challenges and Future Development," *Traffic & Transportation,* vol. 32, no. 4, pp. 475-8, 2020.

[9] P. Andraši, T. Radišić, D. Novak and B. Juričić, "Subjective Air Traffic Complexity Estimation Using Artificial Neural Networks," *Traffic & Transportation,* vol. 31, no. 4, pp. 377-86, 2019.

[10] OpenSky Network, ""The OpenSky Network - Free ADS-B and Mode S data for Research"," [Online]. Available: https://opensky-network.org. [Accessed March 2021].

[11] C. Meckiff, R. Chone and J.-P. Nicolaon, "The Tactical Load Smoother for Multi-Sector Planning," in *2nd USA/EUROPE AIR TRAFFIC MANAGEMENT R&D SEMINAR*, Orlando, 1998.


University of Zagreb
Technische Universität Braunschweig
POLITÉCNICA
UNIVERSIDAD POLITÉCNICA DE MADRID
skyguide
zh aw
School of Engineering
ZAV Center for Aviation
JKU
JOHANNES KEPLER UNIVERSITÄT LINZ
onsulting
lot