



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**USE OF MACHINE-LEARNING TECHNIQUES BASED
ON PYTHON LANGUAGE CODE TO CLASSIFY FAILURE
DATA FROM THE BRAZILIAN AIR FORCE DATABASE**

by

Ygor H. de Almeida

June 2023

Thesis Advisor:
Co-Advisor:

Susan K. Aros
Daniel Cherobini,
Brazilian Air Force

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 2023	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE USE OF MACHINE-LEARNING TECHNIQUES BASED ON PYTHON LANGUAGE CODE TO CLASSIFY FAILURE DATA FROM THE BRAZILIAN AIR FORCE DATABASE			5. FUNDING NUMBERS	
6. AUTHOR(S) Ygor H. de Almeida				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Like other major flight operators, the Brazilian Air Force (BrAF) must effectively manage multiple aircraft fleets, which are costly assets. The failure data collected from these assets is essential for decision-makers to assess the systems' reliability, availability, and maintainability. Obtaining accurate and reliable information depends on the quality of failure data collected. BrAF engineers typically preprocess the data by classifying it as failure or non-failure for analysis, but this task is repetitive and time-consuming. Therefore, this study aims to develop and evaluate a machine-learning model capable of automatically performing this classification task. Of the six machine-learning techniques assessed, the Support Vector Classifier (SVC) model performed best in the F1-score metric. The results suggest that the SVC model has the potential to classify failure data from the BrAF database accurately, saving a significant amount of time. Additionally, the model could aid maintainers during the failure recording process, preventing them from inserting non-useful data in the database, and for inventory management of specific workshop repairs, thus providing more accurate information about the number of failures.				
14. SUBJECT TERMS machine learning, natural language processing, failure data			15. NUMBER OF PAGES 151	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**USE OF MACHINE-LEARNING TECHNIQUES BASED ON PYTHON
LANGUAGE CODE TO CLASSIFY FAILURE DATA FROM THE BRAZILIAN
AIR FORCE DATABASE**

Ygor H. de Almeida
Captain, Brazilian Air Force
BME, Universidade Federal de Minas Gerais, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF BUSINESS ADMINISTRATION

from the

**NAVAL POSTGRADUATE SCHOOL
June 2023**

Approved by: Susan K. Aros
Advisor

Daniel Cherobini
Co-Advisor

Matthew T. Geiser
Academic Associate, Department of Dean of Students

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Like other major flight operators, the Brazilian Air Force (BrAF) must effectively manage multiple aircraft fleets, which are costly assets. The failure data collected from these assets is essential for decision-makers to assess the systems' reliability, availability, and maintainability. Obtaining accurate and reliable information depends on the quality of failure data collected. BrAF engineers typically preprocess the data by classifying it as failure or non-failure for analysis, but this task is repetitive and time-consuming. Therefore, this study aims to develop and evaluate a machine-learning model capable of automatically performing this classification task. Of the six machine-learning techniques assessed, the Support Vector Classifier (SVC) model performed best in the F1-score metric. The results suggest that the SVC model has the potential to classify failure data from the BrAF database accurately, saving a significant amount of time. Additionally, the model could aid maintainers during the failure recording process, preventing them from inserting non-useful data in the database, and for inventory management of specific workshop repairs, thus providing more accurate information about the number of failures.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND	2
B.	RELIABILITY-CENTERED MAINTENANCE.....	5
C.	RCM IN THE BRAZILIAN AIR FORCE	7
D.	RESEARCH QUESTIONS.....	8
E.	ORGANIZATION OF THE STUDY	9
II.	LITERATURE REVIEW	11
A.	THE BRAZILIAN AIR FORCE RELIABILITY SYSTEM.....	11
B.	FAILURE DATA PREPROCESSING PROCEDURE IN THE BRAZILIAN AIR FORCE.....	16
C.	PYTHON PROGRAMMING LANGUAGE.....	20
D.	MACHINE LEARNING	23
1.	Logistic Regression	26
2.	Support Vector Classifier	29
3.	Multinomial Naïve Bayes Classifier	32
4.	Decision Trees.....	35
5.	K-Nearest Neighbors Classifiers.....	40
6.	Machine-Learning Model’s Performance Measures	42
E.	NATURAL LANGUAGE PROCESSING.....	44
1.	Tokenization	46
2.	Stemming	47
3.	Stop Words Removal	48
4.	TF-IDF	49
III.	METHODOLOGY	53
A.	DATA PREPROCESSING PROCEDURES.....	53
1.	Data Collection Procedures.....	53
2.	Data Cleaning Procedures.....	54
3.	Data Classifying Procedures	55
B.	NATURAL LANGUAGE PROCESSING PROCEDURES	57
C.	MODEL CONSTRUCTION AND TRAINING.....	58
IV.	DIFFERENCES IN THE CODE UNDER ASSESSMENT	63
A.	DATA COMPARISON	63
B.	ADDITIONAL CLASSIFIER	65

C.	PERFORMANCE MEASURES USED	66
V.	RESULTS AND ANALYSIS	69
A.	FAILURE DATA MANAGEMENT PROCESS.....	69
B.	MODEL PERFORMANCE AND MODEL SELECTION.....	71
C.	PERFORMANCE OF THE SELECTED SVC MODEL	75
D.	COMPARATIVE ANALYSIS OF MODELS IN THE PRESENT AND PREVIOUS RESEARCH.....	77
E.	OBSERVATIONS ABOUT AN INITIAL ASSUMPTION	79
F.	COMPARISONS OF TIME SPENT ON DATA CLASSIFICATION	82
VI.	CONCLUSION	85
	APPENDIX.....	91
A.	LOAD_AND_TRANSFORM_DATA.PY	91
B.	TECHNIQUES_BUILDING.PY	94
C.	APP_TECHNIQUES.PY	98
D.	MODEL_BUILDER.PY	117
E.	REQUIREMENTS.....	119
	LIST OF REFERENCES	123
	INITIAL DISTRIBUTION LIST	133

LIST OF FIGURES

Figure 1.	BrAF squadrons. Adapted from Forca Aerea Brasileira (n.d.).	3
Figure 2.	RCM decision diagram. Source: Jones (2006).	6
Figure 3.	SISCONF's links. Adapted from BrAF (2006b).	12
Figure 4.	FCDD form front. Adapted from BrAF (2017).	13
Figure 5.	FCDD form back. Adapted from BrAF (2017).	14
Figure 6.	Illustrative representation of an FCDD spreadsheet.	16
Figure 7.	Representation of complete and incomplete failure data for hypothetical pumps. Adapted from BrAF (2006a).	18
Figure 8.	Question views in Stack Overflow. Source: Robinson (2017).	21
Figure 9.	Data-driven decision-making process. Source: Provost and Fawcett (2013).	22
Figure 10.	Machine-learning process. Source: Centric Consulting (2019).	24
Figure 11.	Machine-learning categories. Source: Akbari and Do (2021).	25
Figure 12.	A cheat sheet for choosing the best machine-learning approach. Source: Pedregosa et al. (2011).	25
Figure 13.	Comparison between linear and logistics regressions. Source: James et al. (2021, p. 133).	28
Figure 14.	A hyperplane in a two-dimensional space. Adapted from James et al. (2021, p. 370).	30
Figure 15.	Illustration of the support vectors. Source: James et al. (2021, p. 372).	30
Figure 16.	Support Vector Classifier and its margins. Source: James et al. (2021, p. 376).	31
Figure 17.	Support Vector Machine examples. Source: James et al. (2021, p. 383).	32
Figure 18.	Decision tree example. Adapted from James et al. (2021, p. 332).	35

Figure 19.	Decision trees versus linear regression classifiers. Source: James et al. (2021).	36
Figure 20.	Representative illustration of the Random Forest process. Source: Padovese and Padovese (2019).	38
Figure 21.	Boosting technique representation. Source: Deng et al. (2021).	39
Figure 22.	KNN classifier boundary with different values of K. Adapted from James et al. (2021).	41
Figure 23.	Consequences of different choices of K in a KNN classifier. Source: Steinbach and Tan (2009).	41
Figure 24.	Details about the datasets analyzed, by aircraft type and year.	64
Figure 25.	Locations in Brazil where the studied aircraft types operate.	65
Figure 26.	Proportion of FCDDs manually classified as failures and non-failures, by aircraft type and year.	66
Figure 27.	Diagram representing the failure data management process in the BrAF.	69
Figure 28.	Models' performance.	73
Figure 29.	Ensembles' performance.	74
Figure 30.	SVC model confusion matrix using the validation dataset.	75
Figure 31.	Performance of the SVC model using the remaining data.	76
Figure 32.	Confusion matrix for the models after classifying the data from the year 2021.	77
Figure 33.	Comparing the performance of the models classifying data from the year 2021.	78
Figure 34.	Percentage increase in performance of SVC model compared to Silva et al. (2021) model.	79
Figure 35.	Time spent on manually classifying the datasets.	83

LIST OF TABLES

Table 1.	Confusion matrix. Adapted from Bekkar et al. (2013).	43
Table 2.	FCDDs discrepancy descriptions samples. Adapted from Silva et al. (2021).	50
Table 3.	TF from FCDDs in Table 2. Adapted from Silva et al. (2021).	50
Table 4.	IDF and TF-IDF calculations from FCDDs in Table 2. Adapted from Silva et al. (2021).	51
Table 5.	FCDDs in each data file.	55
Table 6.	Common descriptions of discrepancies and their classifications.	56
Table 7.	Criteria utilized to filter failure data.	71
Table 8.	Hyperparameters selected.	72
Table 9.	Top 10 FCDDs that have the highest probabilities of being failures.	80
Table 10.	Top 10 FCDDs that have the lowest probabilities of being failures.	81

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
BrAF	Brazilian Air Force
COMAR	Regional Air Command
DIRMAB	Directorate of Aeronautical and War Material
FCDD	Defect Data Collection Form
FN	False Negative
FP	False Positive
GUI	Graphical User Interface
HMC	How Malfunction Code
ICA	Air Force Command Instruction
ILA	Air Force Institute of Logistics
KNN	K-Nearest Neighbors
LightGBM	Light Gradient Boosting Machines
LR	Logistic Regression
MCA	Air Force Command Manual
ML	Machine Learning
MSG-3	Maintenance Steering Group 3
MultinomialNBC	Multinomial Naïve Bayes Classifier
NBC	Naïve Bayes Classifier
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PN	Part Number
PN CJM	Part Number High Assembly
PN LHA	Part Number Last High Assembly
RCM	Reliability Centered Maintenance
RDT&E	Research, Development, Test & Evaluation
RFC	Random Forest Classifier

SILOMS	Integrated Logistics System for Material and Services
SISCONF	Reliability System Organizational Framework
SN	Serial Number
SVC	Support Vector Classifier
SVM	Support Vector Machine
TBO	Time between Overhauls
TF-IDF	Term Frequency – Inverse Document Frequency
TN	True Negative
TP	True Positive

ACKNOWLEDGMENTS

Before anything else, I would like to honor God. He has provided me with fortitude and motivation throughout the difficult phases of my time in the U.S. I profoundly appreciate His unending love, mercy, and compassion.

This endeavor would not have been possible without my family. My parents were always proud of my achievements and supportive of my success. My brother, my sister-in-law, and my nephews were pillars to make me not give up during the bad times. My sister and nieces adorably encouraged me to continue pursuing my goals.

I'm extremely grateful to my advisors, Dr. Susan Aros and Major Daniel Cherobini. Their patience and helpful advice were essential in the evolution of the present study. I want to express my deepest gratitude to all NPS faculty, especially the professors who were my instructors. Their high quality and unique knowledge make the entire institution stand out among the best in the world.

I am deeply indebted to the Brazilian Air Force and former superiors in that institution, especially Lieutenant Colonel Nelson Hotta, Lieutenant Colonel Jardel Silva, and Lieutenant Colonel Cesar Imaniche. Without their efforts, I would not be able to be here completing this master's degree. I also want to acknowledge First Lieutenant Lucas Silva and his partners in his research for providing their code and all material used to develop their study.

My friends Jefferson Sousa, who helped me in the data acquisition, and Pedro Lopes, who kindly provided me with his time and knowledge when I was stuck with the Python programming language. I must remember the foreign and the U.S. NPS friends who were always good fellows during this journey.

I could not forget those who have already left us and are close to Jesus Christ now. They are on high but still are and always will be in my heart. Without them, I would not have made it this far.

Last but not least, my beloved girlfriend, Thais, was my safe harbor on the most stressful days and the best company on the days that I could explore the country.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Failure data is important to the life-cycle management of any complex system, such as aircraft, ships, combat vehicles, or any other defense system. It provides decision makers with useful information about system reliability, availability, and maintainability. This kind of information has become of great importance as a mission's success is highly dependent on the correct functioning of the defense systems. Moreover, a cost component is involved in system support. Adequate management of these system characteristics—reliability, availability, and maintainability—can reduce maintenance costs and provide information to managers to make the labor force and resources available when needed and in the required quantity.

The utilization of failure data by managers can provide significant insights, but it is imperative to exercise caution in its application to avoid adverse consequences. Engineers entrusted with the responsibility of conducting analyses based on such data must exercise diligence in evaluating the reliability of the source on which they rely. Moreover, they must exercise critical thinking skills to ensure that the data analyzed accurately reflects the intended meaning to facilitate informed decision making. For instance, there is a common misperception that preventive maintenance is a system failure because the system must be shut down during the procedure. The improper use of data could create the false idea that a system is more dependable than it actually is, jeopardizing the mission or even the safety of the individuals who rely on it. Alternatively, it could create the misperception that a system is unreliable, running up unnecessary maintenance costs and time spent preventing failure.

Selecting the correct data to examine a system's features begins with the failure log. The system's maintainers must have the discipline to record each system failure and activity accurately. This takes time and involves strengthening the institutional culture. Occasionally, the requirement for a ready airplane or ship can outweigh the significance of appropriately documenting a failure, resulting in forms being filled out in minimal detail to comply with the established protocols. Consequently, engineers are compelled to

implement a data-filtering technique due to a lack of confidence in the data caused by insufficient or incorrect information in a failure entry.

In the Brazilian Air Force (BrAF), it has been noted that failure records are not flawless. Hence, the time consuming process of data filtering accounts for most of any analysis using failure data. In the presentation of their work, Sousa et al. (2022) estimated that 50% of their effort was spent preprocessing the data necessary to conduct a reliability analysis on a model of a BrAF attack aircraft. The primary objective of this study is to provide an overview of the knowledge and skills required to apply machine learning to enhance the present time-consuming techniques for processing failure data. This study demonstrates that machine learning may boost the speed and quality of extracting meaningful information from the BrAF database.

A. BACKGROUND

The BrAF is the most significant Air Force in South America (Janes, 2022). The primary responsibilities of this branch of the Armed Forces are the defense of national airspace, aid in the air transport of Brazilian Navy and Army personnel, and support of anti-drug missions in peripheral areas. According to Flight International (2023) more than 500 aircraft are active to accomplish the tasks just mentioned. These assets include fixed and rotary wing models with various goals, including fighter, multirole (light strike/reconnaissance), light attack, transport, tanker, patrol, and others (Brazilian Minister of Defense, 2012).

In terms of personnel, the Transparency Portal of the Brazilian Government (Brazil, General Controllershship of the Union, 2023) indicates that the BrAF has an active force of 70,550 people. Its crew is regarded as one of South America's most professional and well-trained. In addition, compared with its regional peers, it is at the forefront of technological advancements and regional integration (Janes, 2022).

The estimated budget for 2023 is approximately US\$5.0 billion, representing about 25% of the Brazilian defense budget (Janes, 2022). The most considerable portion of this budget is spent on personnel. The current economic situation has obligated the country to cut expenses in other areas, such as procurement and research, development, test, and

evaluation (RDT&E) (Janes, 2022). Considering that a significant portion of the Brazilian defense budget is allocated to personnel expenses, and in light of the necessity to reduce costs, any savings in personnel time are highly valuable.

The force has distributed aircraft in squadrons spread all over the country to accomplish its mission. Figure 1 shows a map of Brazil with the location of each BrAF squadron.



Figure 1. BrAF squadrons. Adapted from Forca Aerea Brasileira (n.d.).

In addition to the units depicted on the map, there are two training squadrons where cadets are taught to become pilots. They are located at the Brazilian Air Force Academy in Sao Paulo state, which is indicated by the blue point within the IV COMAR zone seen in the map in Figure 1. Those training squadrons accommodate the aircraft training models of the Brazilian Air Force.

The base location of each squadron is where the aircraft receive first and second-echelon maintenance service or repair, and any first logistics support needed. The third-echelon maintenance service or repair is done in three depots located, respectively, in Sao Paulo, Rio de Janeiro, and Lagoa Santa; the former is situated near the blue point within the IV COMAR zone, and the last two are located, respectively, near the red and green points in the III COMAR zone seen on the map in Figure 1. These are the most likely locations where BrAF fleet failure data are recorded.

The primary source of failure data for engineers in the preceding context is the BrAF material management system, called the Integrated Logistics System for Material and Services (SILOMS). In this computerized data system, the maintainers must fill out a report each time a failure is detected; this report is called a Defect Data Collection Form (FCDD). This report has essential information regarding the failure, such as the date, the system's operation hours, and a description of the failure.

The failure data registration process in the BrAF needs to deal with some important issues. One of the problems related to the process is that some of the failure reports registered are not associated with real failures; for example, sometimes the maintainer needs to create an FCDD due to items that must be replaced because they have reached the Time Between Overhauls (TBO). Therefore, this cannot be considered a failure. To provide the correct information, engineers in the BrAF spend time analyzing and processing the failure data to make it reliable. The consequence is that gathering accurate information from the failure data available is time consuming. With that in mind, Silva et al. (2021) proposed a machine-learning approach to separate actual failure data from failure reports wrongly inserted into the Brazilian Air Force database. They used failure data from a single aircraft model from 2018 and 2019 (Silva et al., 2021). This research applies the machine-learning approach across a more extensive time range and analyzes failure data from additional aircraft models. Finally, this research explores the feasibility of the application of the model beyond the scope in which it was first tested.

In addition, through a document that determined its science, technology, and innovation policy, the BrAF directed the Air Force Institute of Logistics (ILA) to develop research in the following broad area: the use of artificial intelligence in aircraft fleet

maintenance planning (Brazilian Air Force [BrAF], 2021). ILA provides logistics consultancy for the entire branch in various aspects, mainly concerning material support to the fleet. Thus, this task reveals the BrAF's concern for developing such capability within its military personnel.

B. RELIABILITY-CENTERED MAINTENANCE

As with any other aircraft operator, especially the military ones, the BrAF is always expected to achieve high levels of fleet readiness to be able to fulfill its planning of air missions. Readiness means that the system or equipment is ready to perform a task. This concept is intrinsically related to the concept of reliability. Reliability is the probability of a system or equipment satisfactorily performing the task assigned and to function with no failures during a specific period. In other words, like any other operator, the BrAF wants its equipment ready to accomplish the mission as much as possible.

Considering that the BrAF operates 24 hours, every day of the year, it is not feasible to achieve an aircraft availability of 100%. Any equipment demands at least preventive maintenance to operate safely and therefore be able to carry out the missions. Thus, there will be some time when the system will not be ready to use, as it will be subject to repair or preventive maintenance, which reduces its availability. In the case of aircraft, safety is critical because failure can risk lives and generate high economic losses.

The commercial aviation industry is similar. Companies, in general, expect high availability; however, they still must be concerned with safety. The Reliability Centered Maintenance (RCM) methodology was developed in this context. Its aim is to identify maintenance activities that can be performed on a predetermined schedule to prevent unexpected and premature failures, which consequently enhances overall system reliability and availability (Jones, 2006). Using this methodology, engineers started to view the maintenance issue systematically instead of analyzing each subcomponent of the system. According to Jones (2006), a number of studies were conducted to determine the most effective approach for achieving the aforementioned objective. The outcome of these studies was a document known as Maintenance Steering Group 3 (MSG-3), which provides guidance on how to formulate a maintenance program for any asset, such as an aircraft.

This is the current methodology used to build the maintenance program of new aircraft developed by the industry. This top-down approach to the system was developed in the 1980s to attend to the growing expectations of system maintenance, such as a better cost-benefit, higher readiness and reliability, and concerns about safety (Moubray, 1997).

MSG-3 provides a logic tree similar to the one shown in Figure 2 to build maintenance programs. In this methodology, the engineers must list all possible failure modes of the system. After that, the consequences of the failure modes must be analyzed and classified. After analyzing the information, the engineers can specify the recommended preventive maintenance, or they can decide to change the design to prevent its occurrence (Jones, 2006).

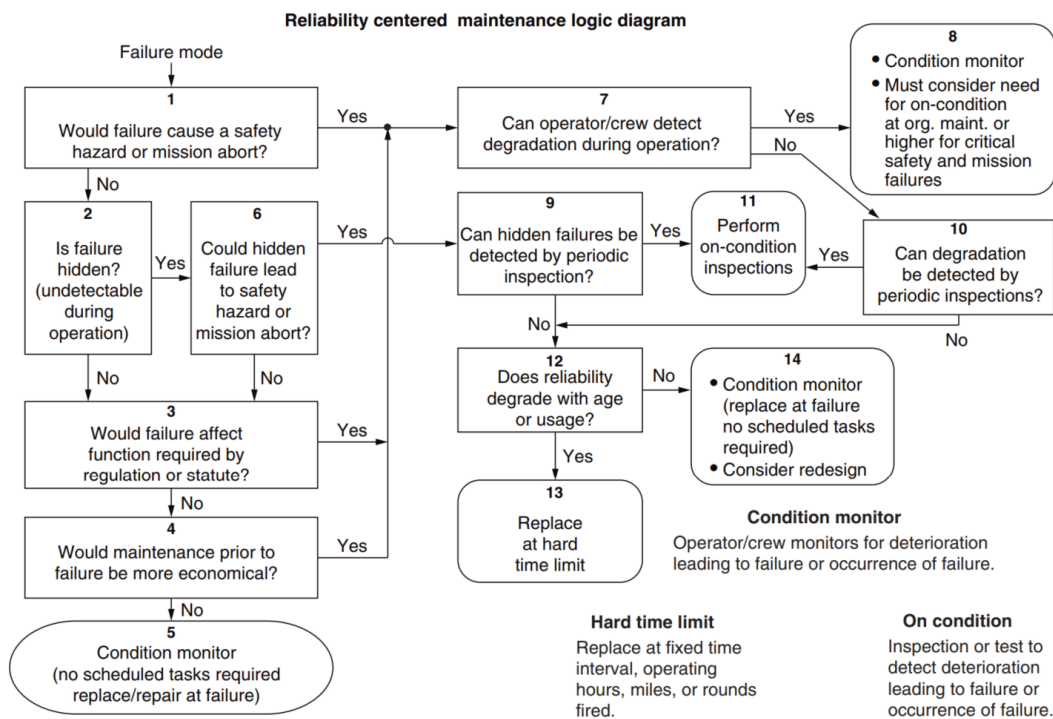


Figure 2. RCM decision diagram. Source: Jones (2006).

According to Moubray (1997), using this methodology has many positive consequences, including increased safety operations, enhanced performance, a more favorable maintenance cost-benefit ratio, and longer functional life for the system.

C. RCM IN THE BRAZILIAN AIR FORCE

In the BrAF, these concepts became especially important after the release of the Air Force Command Manual 400–15 (MCA 400–15) in 2006. This directive introduced the RCM concept and provided a guide to applying this methodology in the BrAF’s departments. The MCA 400–15 initially provides a guide on establishing and managing the RCM program within the BrAF; then, it explains the RCM analysis process in detail, as summarized in the last section. It goes deeper still, explaining how to collect and process the failure data and how to apply this knowledge to reliability analysis, introducing the concepts of reliability function, failure rate function, statistical distributions, confidence intervals, probability density function, and accumulated probability density function. In summary, this directive was what the engineers needed to start looking into the reliability analysis in an official and formal way.

Beyond the RCM application scope, engineers responsible for the fleet supportability in BrAF also use reliability analysis in other specific applications, such as level of repair analysis, definition of spare parts inventory quantity, safety analysis, and decisions to prolong the use of a component beyond the time defined by the manufacturer, among others. However, as this directive states, any prediction validity is related to the quality and accuracy of the provided data. Reliable data, paired with an adequate model, frequently results in solid forecasts. Wrong predictions are usually the result of inaccurate or insufficient data.

The MCA 400–15 dedicates a section exclusively talking about data preprocessing. It does not detail how to do this procedure but emphasizes that the data must pass through a “cleaning” routine to ensure data quality (BrAF, 2006a). According to the directive, this procedure verifies the consistency and accuracy of the data to be utilized in the statistical analysis. Even though the procedure is not detailed in the publication, it is extensively reported by military engineers in their reliability analyses. Silva (2012), Vieira, (2016b), Filho and Martins (2016), and Sousa et al. (2022) took similar procedures for filtering and processing data to perform reliability analysis over different aircraft models. It was observed that one of the steps includes individually analyzing each failure report to verify whether it is a real failure or not, based on the description of the failure. Thus, an automated

process using machine learning would greatly benefit the implementation of this methodology, mainly regarding time savings (Silva et al., 2021).

Overall, it has been noticed that machine-learning techniques bring novel approaches to reliability analysis. These techniques can extract more precise insights from event datasets than can be achieved with conventional analysis techniques (Z. Xu & Saleh, 2021). Some of these techniques use past data to learn a pattern and try to predict the future (Chauhan & Bahal, 2020). In the case of analysis, the program uses processed data to learn and apply the same pattern to filter a new dataset. Beyond the benefits of reducing the time to process the failure data, the assessment of the program built with a broader dataset will inspire the application of a novel data-cleansing instrument for use in reliability analysis (Silva et al., 2021).

In conclusion, there is a need to overcome the repetitive process of filtering and processing failure data present in the BrAF database. As the process is based on the repetition of the identical procedures over different datasets, machine-learning techniques have been proven to be an excellent tool to improve and bring novelty to the field for this kind of problem. While Silva et al. (2021) proposed and tested their model on a narrow dataset, this model must be evaluated against a broader dataset for more extensive use. The research analyzes the use of this model on different aircraft models and compares the results to the ones achieved by Silva et al. (2021) in their work. In summary, this research aims to provide the BrAF with a tool that could decrease the time needed to preprocess failure data and enhance our understanding of how machine-learning techniques could improve reliability analysis within the institution.

D. RESEARCH QUESTIONS

The current procedure for analyzing failure data in the BrAF can be improved, and the present study achieves a method to do this by answering the following questions.

Primary research question: How can the Brazilian Air Force's repetitive and time-consuming data processing procedures be improved using machine-learning techniques and their current applications?

Secondary research questions:

1. What is the current procedure to register, store, and manage failure data in the BrAF?
2. What is the current procedure to process the failure data in the BrAF?
3. What are the theoretical foundations and operational mechanisms of the machine-learning techniques implemented in the code under assessment?
4. What are the current applications of these techniques?
5. How long does it take to classify the data manually using the current procedure? How long does the code take to do the same work? Is there any improvement?
6. Can the results achieved by using the constructed code be considered reliable, assuming that the manual classification is correct?
7. Can the developed code be applied to aircraft models that operate in Brazil's more dispersed regions? Do the diverse ways of registering the failure reports throughout the regions of Brazil influence the results achieved by the code?

E. ORGANIZATION OF THE STUDY

This study begins with this introductory chapter, which provides background about the topic and what will be analyzed in the research. The second chapter presents a literature review of the pertinent topics starting with data management within the Brazilian Air Force, followed by an overview of Python Programming Language and machine-learning techniques, and ending with natural language processing. Chapter III provides the study's methodology, including the data sources and approach adopted. In Chapter IV, some improvements from this research are highlighted. In Chapter V, the results are presented and discussed. Finally, Chapter VI presents the study's conclusions and possible areas for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

This literature review covers some relevant aspects of the knowledge needed to support the research. First, it explains how the BrAF handles its data and later explains the importance of the Python programming language and machine learning in the current big data environment. Finally, concepts and techniques available to manipulate data using machine learning are introduced and explained.

A. THE BRAZILIAN AIR FORCE RELIABILITY SYSTEM

In the same year that BrAF released the directive MCA 400–15 introducing the RCM concept, the BrAF also released another guideline, Air Force Command Instruction 400–21 (ICA 400–21), regarding the reliability system. This directive establishes the Reliability System Organizational Framework (SISCONF) adopted by the logistics area of the Air Force Command. According to this regulation (BrAF, 2006b), the main goal of SISCONF is to communicate, encourage, and utilize RCM to enhance preventive maintenance programs and improve flight safety, fleet operational capacity, and reduce maintenance costs. This document states each BrAF organization's primary responsibilities and roles regarding reliability analysis.

The BrAF (2006b) defines that SISCONF comprises three main links: central body, executive body, and advisory body. The central body's actions are performed by the Directorate of Air Force Materiel (DIRMAB), which is responsible for the regulation and management of the system. The executive body's actions are accomplished by the materiel depots, which are responsible for the management and execution of the tasks listed for the system. The advisory body comprises the military personnel performing the first, second, and third echelon aircraft maintenance activities, and are responsible for entering the maintenance data in SILOMS (BrAF, 2006b). Figure 3 shows this organization's structure.

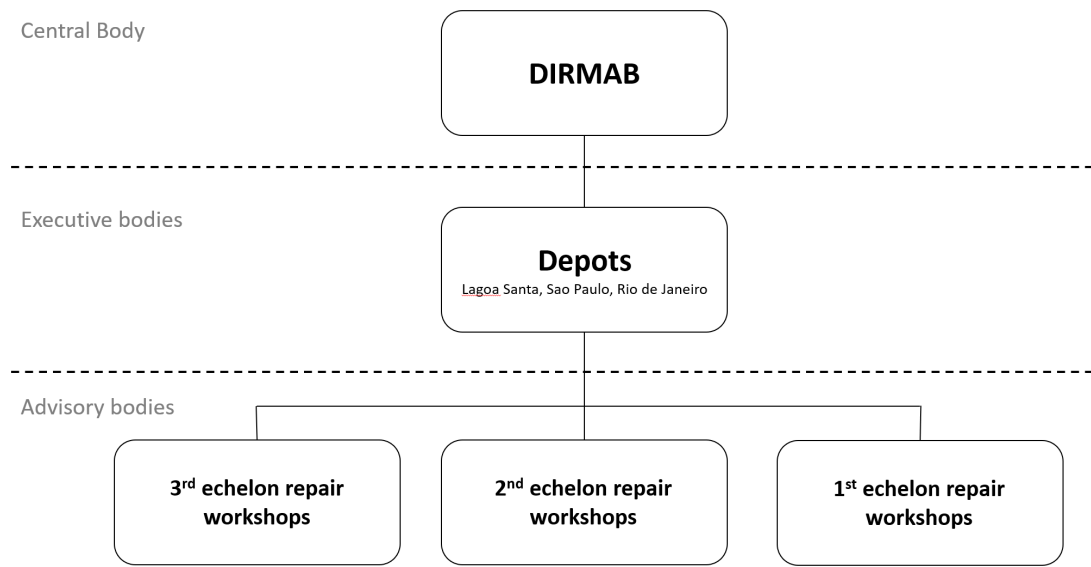


Figure 3. SISCONF's links. Adapted from BrAF (2006b).

According to ICA 400–21, it is the responsibility of the executive body to conduct an RCM analysis and monitor the critical components of the fleet systems to verify their effectiveness in terms of cost and performance, in light of BrAF's needs and operational conditions. In addition, the same source adds that the executive body is accountable for conveying the importance of accurately recording failure data collected during system maintenance procedures. The advisory body also has a task that deserves attention, as ICA 400–21 clearly states that this body is the only one qualified to collect failure data for systems and components and register that data in the SILOMS database. Therefore, BrAF (2006b) already clarifies the significance of the failure data recording process and creates an organizational structure that encourages the completion of reliability studies and the implementation of its results to improve the fleet's operational performance.

Although the previous documents establish the organizational structure and procedures for continuous RCM program management, they do not specify the procedures to register the failures. Another institutional document — the MCA 66–7 Maintenance Manual: Maintenance Doctrine, Processes, and Documentation (BrAF, 2017) — covers all aspects of maintenance within the BrAF, including the documentation of failures. However, what is of interest to this research is the section that explains the FCDD. The

FCDD is the form in which the advisory body registers the failures and stores that information in the SILOMS database. According to MCA 66–7, this document standardizes fleet failure identification, emission, and register control. It must be filled out in the event of a failure, any occurrence that caused damage or malfunction, or any noncompliance observed in a system’s operation. In MCA 66–7 (BrAF, 2017) each field is described in full. According to this document, FCDD forms must be digitally filled out in the SILOMS system, or by physically printing the form and doing it manually. Figures 4 and 5 show a blank FCDD form.

 SISTEMA INTEGRADO DE LOGÍSTICA DE MATERIAL E DE SERVIÇOS PARQUE DE MATERIAL AERONAUTICO DE LAGOA SANTA FICHA DE COLETA DE DADOS DE DEFEITO		Pag.: 1 de 2 Data: Hora:	
Identificação OM : Nº FCDD: Categ. :		Ocorrência Data: Hora: Status:	
Item Defeituoso PN : Nomenclatura: CODEMP : Nome Empresa: SN : Controle :			
Instalado em CJM? Matricula : Instalação: Projeto : Dt. Mt. : TSN :			
Remoção : Qtde Item: 1 Nº Lote : TSO :			
TMC : WDC : SIST/WUC : Relator : Discrepância: Observação :			
PN LHA PN : CODEMP : SN : WUC : Posição:		PN CJM PN : CODEMP: SN : Matricula:	
Dados da Missão Missão : Fase : Altitude: Atitude: Fator de Carga : Velocidade Meteor. :			
Análise de Oficina Nº O.S. : St. O.S. : Setor: Data: Garantia: S Resp. Oficina : ATC : WUC: Coment. :			

Figure 4. FCDD form front. Adapted from BrAF (2017).

 SISTEMA INTEGRADO DE LOGÍSTICA DE MATERIAL E DE SERVIÇOS PARQUE DE MATERIAL AERONAUTICO DE LAGOA SANTA FICHA DE COLETA DE DADOS DE DEFEITO		Pag.: 2 de 2 Data: Hora:
Identificação OM : N° FCDD : Categ. :		Ocorrência Data : Status : Hora :
Parecer Engenharia Resp. : Coment. : Provid. :		
Dados do Bélico Medida Estocagem: Fabricação: Validade: Qtde. Estoque: Local última revisão :		
Defeito Lançador N° Tubo N° Lancto Tubo		
Defeito HMC HMC		

Figure 5. FCDD form back. Adapted from BrAF (2017).

The FCCD form has 65 fields to fill in, divided into ten sections. The sections are:

1. Identification (Identificação): this section provides information about the organization that registered the form.
2. Occurrence (Ocorrência): this gives the date and time that the defect occurred.

3. Defective Item (Item Defeituoso): this is the main section of the form, which gives all details about the defective item; this is the section on which this study focuses most.
4. Part Number Last High Assembly (PN LHA): usually, this section identifies the aircraft that was affected by the failure that is being recorded.
5. Part Number High Assembly (PN CJM): usually, this section identifies the system that was affected by the failure that is being recorded, for example: the engine, the landing gear, propellers, etc.
6. Mission (Dados da Missão): this section details information about the mission that was being accomplished.
7. Workshop Repair Analysis (Análise da Oficina): the workshop repair comments about the failure are detailed in this section.
8. Engineering Recommendation (Parecer Engenharia): this is a conclusion about the failure given by the engineers.
9. Armament Material Data (Dados do Bélico): if the item is an armament material, it has some specificities, and this section is specifically included to add this kind of information.
10. How Malfunctioned Code Defect (Defeito HMC): this field presents a code that represents how the failure occurred.

Additionally, MCA 66–7 provides a list of sample cases in which an FCDD should be recorded. Among a list of nine cases, two deserve a comment because they are not directly related to failure cases. The first is for an item that has had its maintenance program modified, which must be sent to the material depot. The second is related to items suspected of being part of an accident. Both cases do not involve failures, and the insertion of these forms in the SILOMS database distorts the data. Therefore, even if all the written instructions were followed, the data would still have to be cleaned and filtered to be accurately analyzed.

Every day several failures occur in the BrAF fleet. Each failure initiates an FCDD. The FCDD form is filled out and electronically stored in the SILOMS database. This database provides the data that engineers need to do reliability analysis. To manage the data and perform statistical analyses, the engineers download a .csv (comma-separated value) spreadsheet containing the entire FCDD in a single document row. Therefore, each row of the document holds a failure record. Each column holds a field of the FCDD. Figure 6 displays an example of a FCDD spreadsheet screenshot.

	A	B	C	F	G	H	I	J
1	Nº FCDD	UNIDADE	PF	MAT	PN ITEM	SN ITEM	Column1	ANO
2	70736	AFA	T1	1425	ANV T-27	312379	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
3	70754	AFA	T1	1393	ANV T-27	312144	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
4	70806	AFA	T1	1442	ANV T-27	312396	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
5	5457	PAMALS	T1		AE10008-001331		PAINEL TRNSF R	2018
6	70844	AFA	T1		2B7-38	10M343N	PUMP SUBMERGED AIRCRAFT	2018
7	70843	AFA	T1		2B7-38	1AD57J	PUMP SUBMERGED AIRCRAFT	2018
8	70853	AFA	T1	1360	ANV T-27	312074	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
9	70866	AFA	T1	1341	ANV T-27	312055	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
10	70852	AFA	T1	1416	ANV T-27	312189	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
11	70862	AFA	T1	1386	ANV T-27	312104	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
12	70854	AFA	T1		29255-10A-A803174		REGULATOR OXYGEN DILUTER DEMAND	2018
13	70883	AFA	T1	1372	ANV T-27	312087	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
14	70885	AFA	T1	1386	ANV T-27	312104	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
15	70882	AFA	T1	1416	ANV T-27	312189	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
16	70884	AFA	T1	1425	ANV T-27	312379	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018
17	70874	AFA	T1	1431	ANV T-27	312390	AERONAVE ASA FIXA EMBRAER 312 (T-27 TUCANO)	2018

Figure 6. Illustrative representation of an FCDD spreadsheet.

In conclusion, the BrAF has an established reliability system that supports the engineers in their duties. This system is formally structured by many official documents with detailed instructions about performing the steps to pursue better system performance outcomes. There are directions on how the failure data is recorded, stored, and managed. However, even if the instructions are properly followed, it is still necessary to clean the failure data before using it for statistical analysis.

B. FAILURE DATA PREPROCESSING PROCEDURE IN THE BRAZILIAN AIR FORCE

Any reliability work needs relevant data that must come from a reliable resource. In addition, analysts should always ensure that the information can serve its purpose in the

particular context, according to the theory used for analysis. In the previous section, it was shown that MCA 66–7 lists two cases in which a FCDD should be recorded, but which do not represent the occurrence of a failure. In addition, errors are inherent in humans observation of phenomena (Sousa et al., 2022). Therefore, knowing how the BrAF engineers preprocess the failure data used in their reliability analysis is crucial.

The important concept of complete and incomplete failure data must be introduced here as the proposed mathematical models are based on both to predict system behavior. For reliability analysis, the data of interest is usually the time until system failure. The best way to gather this information is through field data collection. It should also be noted that when collecting data in the field, it is common to observe systems that do not fail. On an aircraft, for instance, the pilot may observe a malfunction in the air conditioning system while other subsystems, such as the engine and the landing gear, remain in excellent working condition. The flight times of these other subsystems are also useful statistics for reliability analysis. Hence, the system's life data can be categorized either as complete data when a failure was observed during field collection, or as incomplete data when the system was still operating at the time of data collection (BrAF, 2006a). Incomplete data can also arise in situations where a system needs to undergo preventive maintenance, such as a comprehensive overhaul, which requires the system to be temporarily shut down. In such cases, we may have information about the total operational time of the system but lack precise details about the exact time of failure, because the system was shut down before its failure.

The aforementioned concepts are depicted in Figure 7, wherein complete data is represented by items A, C, and D, while incomplete data is exhibited by items B and E. The reason for such incompleteness is attributed to the fact that these items did not fail during the observation period. In order to provide a clearer interpretation of Figure 7, it is useful to consider each letter as representing a fuel pump sample undergoing a test for a span of t_{test} hours. All the samples initiate the test together at time zero. According to the graph, pump D was the first to fail; it resisted for t_D hours. Pumps A and C lasted longer in the test, for t_A and t_C hours, respectively. Pumps B and E completed the test entirely, and it is not known how many more hours they would run until failing. Therefore, data for pumps

A, C, and D is, respectively, t_A , t_D , and t_C . The data for pumps B and E is t_{test} ; however, it is incomplete because even though the test ended, the pumps are still working. According to Reid (2022), those items have right-censored data, where their failure time is unknown, but it is known for how long the item has been running with no failures.

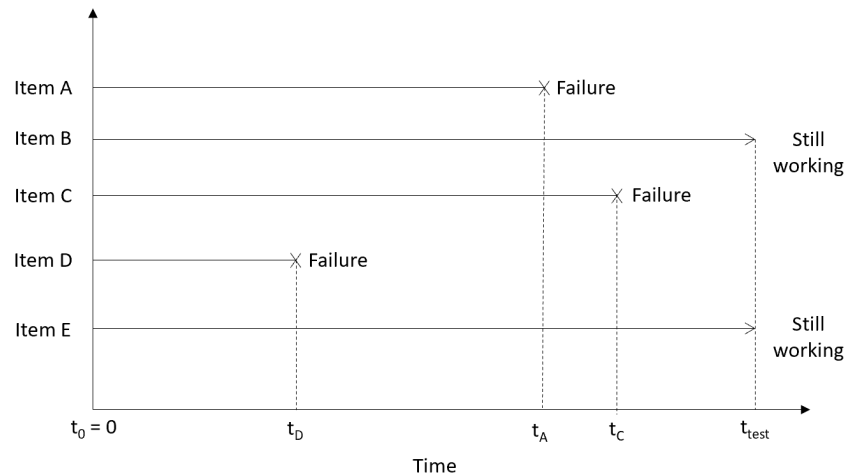


Figure 7. Representation of complete and incomplete failure data for hypothetical pumps. Adapted from BrAF (2006a).

In order to verify how the BrAF engineers preprocess the failure data, 14 reliability reports were analyzed. Among these reports, two of them—Vieira (2015b) and Vieira (2015c)—had no evidence of preprocessing the failure data; instead, they used the data as it was collected. The analyst did not use the SILOMS database in those reports as a resource. The source was data collected and kept by individuals and not shared in the database.

Vieira (2019), Sousa (2020), Sousa (2021), and Sousa et al. (2022) did not specify what the criteria were for filtering the failure data. However, they clarified that some filter and pre-analysis of the data was performed before using that data to build the reliability models.

The most common reason reported for invalidating the data from SILOMS was the occurrence of preventive maintenance. Vieira (2016a), Cavalcante and Ferreira (2015), Silva (2012), Filho and Martins (2016), Chaves (2020), and Silva et al. (2021) needed

complete data to perform their analysis. Therefore, they removed any data whose descriptions in the FCDD were related to preventive maintenance, for example, a description saying that the item had achieved its time between overhauls (TBO). However, it is essential to note that some engineers still use incomplete data in their analysis. Thus, whether or not incomplete data is used depends on the analyst's goal.

Another reason to invalidate an FCDD is duplication of the same failure. In the BrAF database, it is common to find more than one FCDD related to the same item on the same day; this is evidence that a failure was recorded in duplicate. Vieira (2016a), Filho and Martins (2016), Chaves (2020), and Silva et al. (2021) clearly express this as one of the steps they used to filter their data.

Vieira (2016a), Filho and Martins (2016), and Silva et al. (2021) also report a poor description of the failure as a motive to invalidate a recorded failure. These authors noted that some descriptions do not clarify whether a failure occurred or not. An example of this would be that the component was described as needing to be replaced, but the description did not specify why. During the recording process, maintainers should be more precise about the reason for the replacement. It might be to support another aircraft, for instance, or because the item was not functioning properly. In addition, Vieira (2016a) and Chaves (2020) claim that one description indicated that the FCDD was registered because a particular item needed to be swapped between aircraft, which had no connection to failure. This is a common practice, known as cannibalization, which is used in maintenance to enable an asset to return to service. Jacobs (2000) defines cannibalization as “removing a serviceable component from one aircraft and installing it in another aircraft to restore it to a serviceable condition” (p. 17). It is not unusual to find FCDD discrepancy descriptions that describe a cannibalization situation.

A last example of invalid FCDDs found in the reports was an item or component that experienced two consecutive failures within a short period or in no operating time at all. Vieira (2015a) and Vieira (2016a) considered such cases as “infant mortality,” and according to the hypothesis he employed, this data should have been discarded in his studies.

In conclusion, the reasons why the engineers in BrAF invalidate some FCDDs in their analysis are many, and it depends on the goal of each analysis. It is also clear that the FCDDs are not filled out according to the instructions in effect. The need to filter data is unquestionable. However, the parameters and criteria for how it should be done is a gray area that must be assessed case by case.

C. PYTHON PROGRAMMING LANGUAGE

Python is a programming language that has gained much popularity over the past years and its use is continuously growing. One of the reasons for its wide acceptance is that Python is a high-level programming language, which means that the way it is written is readable for humans; therefore, it is easier to learn and read others' code written in Python than in other programming languages. Another reason Python is gaining popularity is the high availability of its packages. Hence, it is common to find Python used in many fields, from biology and education to data mining and engineering.

Python has an extensive list of applications across many fields. According to Welcome to Python.org (2023), this language has third-party modules for web and internet development, scientific and numeric computing, education, software development, and business applications, and it supports a desktop Graphical User interface (GUI).

Stack Overflow, a traditional website where enthusiasts of programming share questions and answers about the subject, serves as an indicator of how widely used each programming language is worldwide. As shown in Figure 8, since 2017 most of the question views in Stack Overflow have been related to Python. In addition, from the year 2016 to the year 2017, Python experienced a growth of 27% in the share of traffic (Robinson, 2017), much higher than the other prominent programming languages such as Javascript and Java, which experienced an almost null growth. Thus, Python is the only one considered large and growing rapidly.

This rapid growth seen in Figure 8 is closely related to the importance and application of artificial intelligence. Earlier, machines and computers were employed just to execute computations at an extremely high rate. However, now countless researchers and large corporations are working tirelessly to develop intelligent systems capable of

completing jobs typically performed by humans, using the same machines and computers and their high capacity to execute repetitive tasks (Zestminds, 2022).

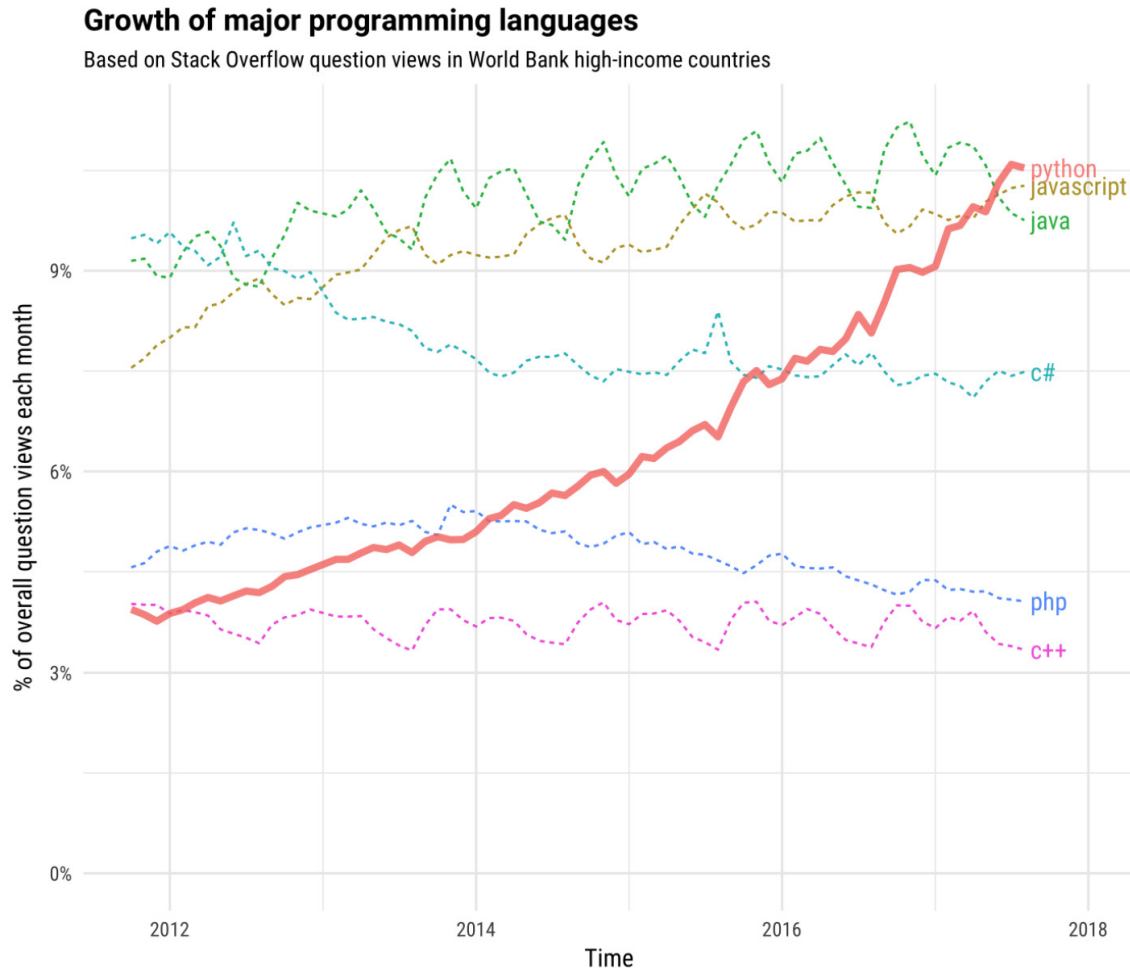


Figure 8. Question views in Stack Overflow. Source: Robinson (2017).

Data-Driven Decision Making, which refers to basing a decision on data analysis rather than the pure knowledge of an individual (Provost & Fawcett, 2013), has recently gained much attention. Nowadays, the number of decisions and the velocity at which they must be made is increasing, and companies must be proactive to thrive in the market. Therefore, decision makers are transferring this task to automated systems that can complete the task more efficiently. Figure 9 illustrates this process by which data science

techniques are used to collect data and transform it into useful information spread across the company.

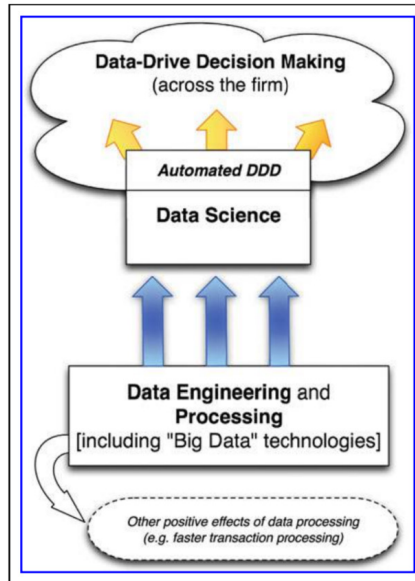


Figure 9. Data-driven decision-making process.
Source: Provost and Fawcett (2013).

The Python programming language has become especially important in this process because it has several libraries capable of handling each part of the process. For data processing, the “numpy” and “pandas” modules provide efficient ways to manipulate numeric data, such as math operations. The scikit-learn module contains extensive applications of machine-learning techniques, which drive automated decisions. It is also important to note the matplotlib, a module that can build stationary, dynamic, and collaborative charts, which makes the numbers easy to understand.

Numerous studies have utilized this language to develop their discoveries. In the logistics field, Custard et al. (2016) used Python to compare backorders and excessive stock in their study to reduce the repairable pieces in a naval aviation depot; Rundong (2020) built models to evaluate the reliability, operation, and maintenance performance of autonomous guided vehicles in various circumstances; and Iwata and Mavris (2013) created a virtual environment to simulate operation and logistics support for future systems

developments. Therefore, the use of the language is extensive and varies in complexity. In the cited examples, Python has been used for everything from basic table comparisons to highly complicated simulations of whole environments.

This section has presented evidence of the rising popularity of Python in the domains of machine learning and artificial intelligence. Consequently, this language was the ideal tool to incorporate machine-learning techniques in the automated classification of failure data from the BrAF database. Additionally, it is important to note that the fields of artificial intelligence (AI) and Machine-learning (ML) are still in the process of being disseminated and established within the BrAF. Therefore, the convenience of Python compared to other programming languages makes it a more accessible option for novice users, whom the BrAF apparently aims to develop, as was demonstrated in its document that determined its science, technology, and innovation policy (BrAF, 2021).

D. MACHINE LEARNING

In order to elucidate the various techniques utilized to construct the data classification model, it is essential to understand the fundamental principles of ML. Although each technique adopts distinct approaches to accomplish the overarching goal of ML, they are all grounded in the same objective. Thus, the subsequent paragraphs delve deeper into the field of ML to provide a comprehensive understanding of the developed model's foundations.

According to Zhou (2021), the ML process can be comparable to human predictions about routine experiences. It is commonly observed that a clear, starry night is often followed by a pleasant weather day. The explanation for that comes from past experience. A clear night in the current day is frequently followed by a day with beautiful weather. A similar process occurs with computers. However, in this case, past experience comes from data. Therefore, ML refers to the process of building models from past data, which can predict future behaviors on new observations (Zhou, 2021).

Centric Consulting (2019) describes the ML process as a sequence of steps. Initially, data is collected, and subsequently cleaned and partitioned into two distinct sets, namely the training set and the testing set. A model is then constructed and evaluated

utilizing the aforementioned datasets. Upon successful completion of this stage, the model is implemented and employed to predict new observations using an independent set of new data. After deploying the model and getting user feedback, the developer can improve the model by adding other data elements or changing the prior technique chosen. Figure 10 illustrates this process.

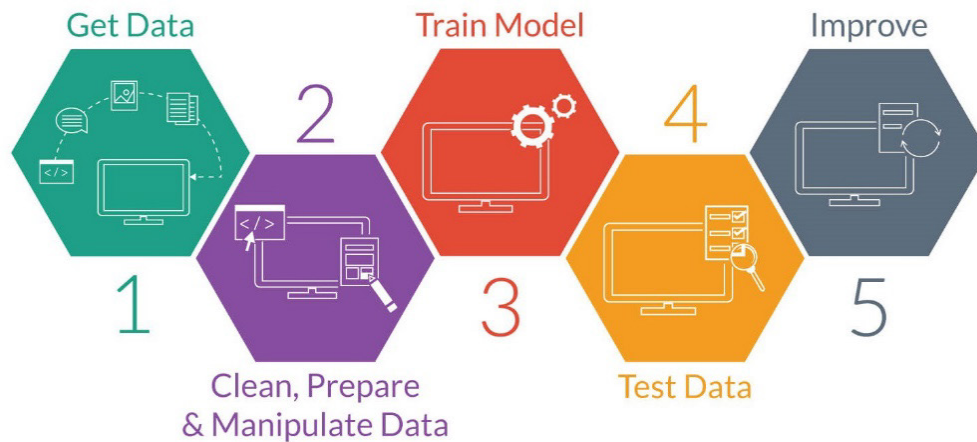


Figure 10. Machine-learning process. Source: Centric Consulting (2019).

There are three categories of ML: supervised, unsupervised, and reinforcement learning. In supervised learning, past inputs and their related outputs are provided to the system. This sort of learning is mainly utilized to match inputs with desired outcomes (Akbari & Do, 2021). The failure data classification process falls into this category. In contrast, unsupervised learning does not rely on previous knowledge of inputs and outcomes; instead, it combines comparable data inputs and calculates their density and probability (Akbari & Do, 2021). An example of unsupervised learning is companies' use of customer data to predict their customers' consumption behavior. Reinforcement learning uses an algorithm that evolves and adapts during the learning process in an effort to improve conventional approaches (Akbari & Do, 2021). A current application of this method is the recommendation systems that online retail shops use; based on the rating of other customers, the systems recommend a specific product brand. Examples of each of these ML techniques are presented in Figure 11.

As illustrated in Figure 12, many techniques apply to each objective depending on the data processing goal. The figure shows four main domains that analysts could predict from data. The analysts could classify, cluster, predict a number (represented in the regression area), or reduce the number of features or variables in a dataset while retaining as much information as possible, as demonstrated in Figure 12 by Pedregosa et al. (2011). Thus, before applying ML, it must be clear what the analyst's goal is.

Based on these four domains and the three categories of ML previously presented, it is possible to identify machine-learning applications within the BrAF context beyond the spectrum analyzed in this study. In their literature review, for example, Akbari and Do (2021) found that between 1994 and 2019, 86% of studies related to the application of ML in Logistics and Supply Chain Management were about predictions and optimizations. Therefore, any problem that involves prediction or optimization and has available data (at least 50 samples), based on the scikit-learn cheat sheet, are good candidates for the ML processes applications.

Some of the most common ML techniques used are Artificial Neural Networks, K-Means Clustering, Decision Trees, algorithms for reducing the number of dimensions, K-Nearest Neighbors, Linear Regression, Support Vector Machines, Naive Bayes Classifiers, Random Forests, and Logistic Regression (Akbari & Do, 2021). The model built by Silva et al. (2021) used the last four techniques cited and another one called LightGBM.

These five techniques, plus the technique of K-Nearest Neighbors (KNN) are further explained and explored in the following subsections. These are the techniques most commonly used in classification problems (Do, 2022; Keita, 2022; Ray, 2017; Wolff, 2020), and this is the reason why they were considered to develop the model in this study. Future studies should exploit other techniques to provide additional possible applications within the BrAF context.

1. Logistic Regression

In a classification problem, the objective is to predict a qualitative condition, unlike models where the goal is to predict a numerical quantity. For instance, when an airplane exhibits malfunctions and is sent to the hangar, the mechanics may predict the issue based

on the description of the event. Logistic regression is one of the options available to address such classification problems.

During the latter part of the 20th century, logistic regression has received significant attention as an effective method for analyzing a response variable with two possible outcomes (Peterson, 1998). Mehrjoo and Bashiri (2013) observed that binary logistic regression is commonly used when the response variable has two possible outcomes, and the predictor variables can be either categorical or continuous. Therefore, it is mainly used in applications where the expected answer is “Yes” or “No” or binary responses “1” or “0.”

According to James et al. (2021), the method is based on the logistic function given by the following formula.

$$p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Following mathematical manipulation, the aforementioned expression can be reformulated as the following equation (James et al., 2021):

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta_1 x$$

When the prediction $p(x)$ is based on more than one variable X , it is called Multiple Logistic Regression, and the formula becomes the following (James et al., 2021).

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

These equations are the basis for performing a logistic regression and building a model, aiming to predict the probability value of $p(x)$, which will be given by a number between 0 and 1. Therefore, the closer the probability is to “0” the outcome is classified in one category, and the closer the probability is to “1” the outcome is classified in the other category. The challenge is to find the values of parameters β , which is done by the scikit-learn module in Python based on the Maximum Likelihood theory, which is beyond the scope of this study. More details about this subject can be found in James et al. (2021).

To illustrate the difference between linear and logistic regression, James et al. (2021) applied both theories to a dataset in which they tried to predict the probability of default in a bank based on the customers' account balances. In this case, the answer is binary; either the customer will be in default ("Yes"), or the customer will not be in default ("No"). This is a classification problem, and therefore, logistic regression is more appropriate in this case. Figure 13 shows the models created based on both regressions. On the left is the linear regression; on the right is the logistic regression. The models are the curves in blue, and the original data are plotted in yellow. It can be seen that the logistic regression fits the data better.

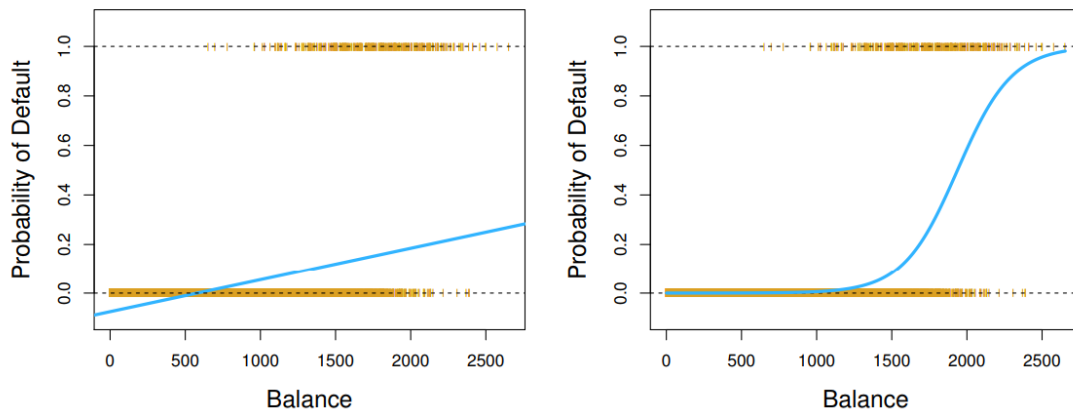


Figure 13. Comparison between linear and logistics regressions. Source: James et al. (2021, p. 133).

The use of logistic regression is extensive and not new. Peterson (1998) explained studies from 1992 using this method. In addition, he stated that the increased use of the technique began in the 1960s. In his work, he used the theory to classify patients as diabetic or not diabetic based on blood measures. Additionally, he use a similar approach to verify if there was a spread of cancer in prostate cancer-diagnosed patients based on medical examinations. Talpová (2014) reported on studies in biomedical research, ecology, finance, educational research, and meteorology. She discussed the potential use of the methodology, particularly in management research. Similarly, Keizers et al. (2003) documented studies in management research in 1996 that used logistic regression to determine whether a

project would be delivered on time based on the project's characteristics. Mehrjoo and Bashiri (2013) developed a model using the technique to predict if it would be possible to complete a daily vehicle production plan based on production characteristics like vehicle color, inventory available, and type of vehicle, among others. They achieved good results in predictions for an Indian automotive company. In conclusion, the method's effectiveness has been demonstrated in various sectors and in a reasonable time frame.

2. Support Vector Classifier

The Support Vector Machine (SVM), a tool created by computer scientists in the 1990s, is used to categorize data, and it has become increasingly popular ever since. SVM performs well when the data groups are complicatedly intertwined, and it can be utilized in diverse areas to gain insights from vast quantities of data. While SVM is a more general algorithm that can be used for both classification and regression tasks, Support Vector Classifier (SVC) is a specific type of SVM just for classification tasks. According to Ma et al. (2020), SVC is widely recognized and efficient when classifying text. Therefore, this tool would be a good candidate for the problem being studied as it will deal with some failure discrepancy descriptions.

Basically, SVC separates data by a hyperplane in a multidimensional space. A hyperplane is a subspace of a space that is one dimension lower than the number of dimensions in the multidimensional space being considered (James et al., 2021). For example, in a plane, which has two dimensions, a hyperplane would be a line that has just one dimension. On the left side of Figure 14, the hyperplane line in black separates datasets into two regions, blue and pink. If data falls in the blue region, it is classified according to this region. Otherwise, if data falls in the pink region, it receives another classification. The issue lies in the existence of multiple hyperplanes capable of effectively segregating the data, as illustrated on the right side of Figure 14. However, this scenario results in the emergence of divergent regions, leading to the disparate classification of certain data points.

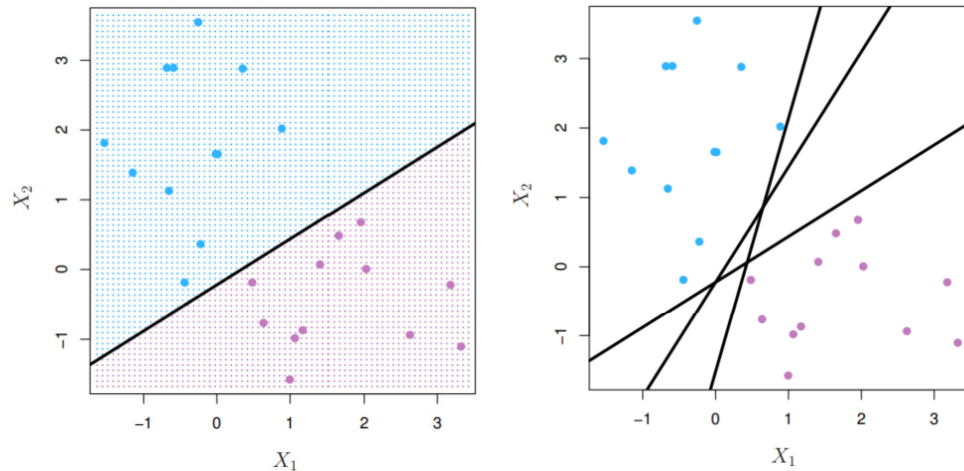


Figure 14. A hyperplane in a two-dimensional space. Adapted from James et al. (2021, p. 370).

James et al. (2021) affirmed that a natural choice to solve this problem was to find a hyperplane that maximizes the distance between the hyperplane and the nearest points, which the authors referred to as the “optimal separating hyperplane.” By definition, the distance between a point and a plane is given by the length of the vector that links the point and the plane with a 90-degree angle. These vectors, illustrated in Figure 15, are called the support vectors and originate the technique’s name.

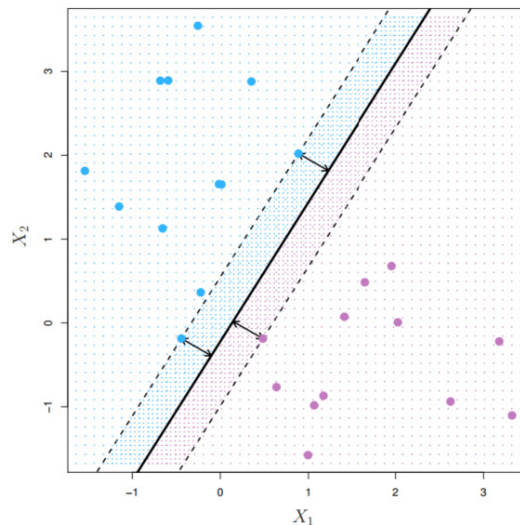


Figure 15. Illustration of the support vectors. Source: James et al. (2021, p. 372).

The model built with an SVC tool still permits some kind of error; in other words, it permits some training data points to fall into a gray area of the space or even on the wrong classification side. Figure 16 illustrates this process; on the left side, data numbers 1 and 8 are in the gray area; on the right side, data numbers 1, 11, and 8 are in the gray area, and data number 12 is on the wrong side. This margin must be set, increasing the model's resilience to specific data points and an improved categorization of most training data (James et al., 2021).

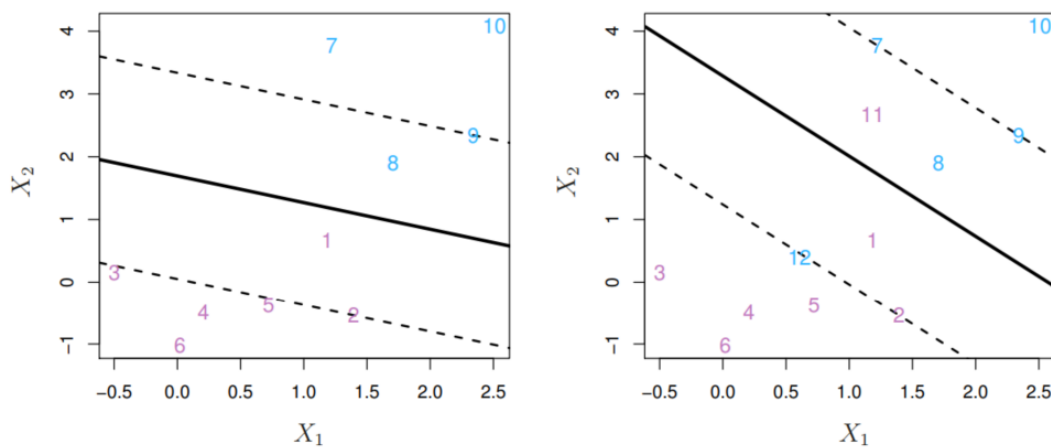


Figure 16. Support Vector Classifier and its margins. Source: James et al. (2021, p. 376).

The SVC model is an equation of this hyperplane, correlating the predictors to the prediction. The mathematics to find this equation is beyond the scope of this study, but it is well explained by James et al. (2021), and it will be performed in the scikit-learn module in the Python programming language.

SVC assumes a linear correlation to build the hyperplane equation. In contrast, SVM assumes more complex correlations, and the hyperplane definitions and boundaries can assume any format, not just the linear one. Some applications of the SVM method are illustrated in Figure 17.

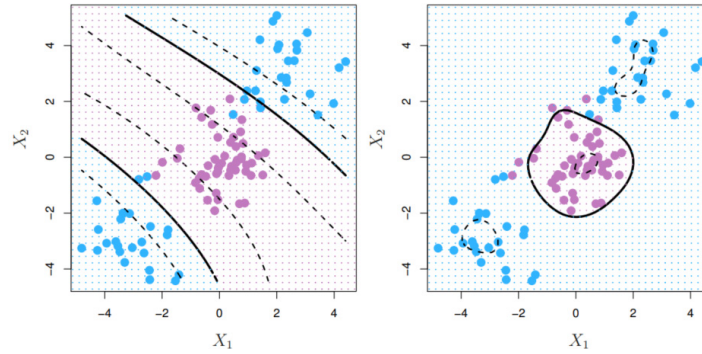


Figure 17. Support Vector Machine examples. Source: James et al. (2021, p. 383).

The applications of SVC and SVM are extensive. Fernandez-Grandon et al. (2021) employed SVM using chest X-ray images to identify whether patients had COVID-19, pneumonia, or healthy lungs. Shamohammadi et al. (2023) used GPS information to build an SVM model to identify a population's primary transport mode (car, train, bus, bicycle, or foot) for better public transportation planning. Ma et al. (2020) compared the performances of SVM and Naïve Bayes Classifier techniques in classifying e-mails as spam. Abedini et al. (2019) used the technique to check how susceptible an area is to landslides. In their study, Zhao et al. (2021) used SVM to classify suppliers' credibility. In a more complex application, Hua and Zhang (2006), Y. Li and Li (2019), and R. Li et al. (2021) used the technique to predict the demand for different products. These examples show how broadly applicable and practical this tool can be.

3. Multinomial Naïve Bayes Classifier

Naïve Bayes Classifier (NBC) is another machine-learning technique for building models to perform qualitative predictions. This classifier is based on Bayes' theorem, which expresses the probability of an event happening given the previous occurrence of another event. Denoting $P(A)$ the occurrence probability of an event A , and $P(A|B)$ the probability occurrence of event A given that event B had already occurred, the Bayes Theorem can be written as the following formula.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

As with any other classifier, the goal is to identify to which class C a given data Y belongs, given a set of attributes X that this data has. According to Bayes' theorem:

$$P(C|X) = \frac{P(X|C) * P(C)}{P(X)}$$

$P(X)$ can be disregarded in this context because it is the same for all classes. This means that it does not impact the relative probabilities of those classes. Therefore, it is noticeable that the $P(C|X)$ is directly proportional to the product $P(C) * P(X|C)$. Given that X is an array composed of n attributes, if we assume they are independent, the previous product can be rewritten as the following (Domingos & Pazzani, 1997).

$$P(C) * \prod_{i=1}^n P(x_i|C)$$

According to Domingos and Pazzani (1997), in most cases, it is not possible to assume that the attributes are independent given the classes, but this is disregarded. This is the reason why the technique has “naive” in its name. Even though the independence among the attributes and classifications cannot be assumed in most cases, this classification done by NBC is still helpful and can minimize errors in classifying problems (Domingos & Pazzani, 1997). Langley et al. (1992) also achieved good results using the Bayes Classifier, even disregarding its assumptions. Zhang (2004), in his research, provided a possible explanation for these surprisingly good results: in this author's assessment, the dependency relationships may cancel each other when the whole system is considered.

Finally, from the previous statements, the conditional probability may be rewritten as in the following equation (Alsanad, 2022).

$$P(C|X) \propto P(C) * \prod_{i=1}^n P(x_i|C)$$

The $P(C)$ value is determined from the relative frequency of each class obtained in the training dataset (Alsanad, 2022).

The remaining question concerns the distribution of the $P(x_i|C)$ factors. In the Naïve Bayes Multinomial Classifier, it is assumed that this factor follows a multinomial distribution. The multinomial distribution is an extended concept of the binomial distribution to a more general rule. The binomial distribution has just two outcomes (success or failure), whereas the multinomial distribution has as many outcomes as necessary depending on the case, each with its own probability.

In order to find the values of $P(x_i|C)$, a parameterized vector is created for each possible C class, denoted as $\theta_k = \{\theta_{k1}, \dots, \theta_{ki}\}$, in which each k represents a different class C and i represents the number of attributes. Thus, the parameter θ_{ki} represents the probability $P(x_i|C)$ for a given class k . The θ_{ki} parameters are determined through a smoothed variant of maximum likelihood given by the following equation (Alsanad, 2022).

$$\theta_{ki} = \frac{N_{ki} + \alpha}{N_k + \alpha * n}$$

In this equation N_{ki} represents the number of times the attribute i appears in class k , and N_k represents the total number of attributes in the class. The parameter α is the smoothing parameter. This is needed to avoid divisions by zero in the parameter's calculations and to guarantee that the sum of probabilities for all attributes for a given class is one. It assumes a value greater than zero and less than or equal to one (Alsanad, 2022). They are calculated using the training dataset in the model building stage.

Zhang (2004) affirmed that NBC cannot provide an accurate probability estimation; thus, its performance for predicting numbers has been proven to lead to errors. However, the method shows promising results when it is used for classification. In other words, the method is useful for providing information regarding which class a given attribute dataset belongs to; however, the method fails if one needs to provide a specific probability value (for example, a probability of 0.90 for a certain event).

Text categorization and sentiment analysis (also known as opinion mining) in written comments are often performed using NBC, which is widely chosen for such cases over other methods. Dey et al. (2016) compared NBC and other classifying methods to categorize movie and hotel reviews. Rahmaningrum and Oktaviana (2020) did similar

research to perform a hotel service review sentiment analysis. In another example, Alsanad (2022) used the multinomial NBC to perform a sentiment analysis of Arabic language speakers' comments on the Twitter social media platform. Using this method to classify the analyzed text data, those researchers achieved good performance metrics. Therefore, this is another suitable candidate method to analyze the failure discrepancy descriptions data in evidence in this research.

4. Decision Trees

The following two remaining techniques to be discussed are specific applications of the broader concept of decision trees. Decision trees are a well-known and commonly utilized ML approach for both classification and regression issues. As the name implies, decision trees utilize a tree-like structure to make decisions according to a set of rules or criteria. The tree consists of internal nodes representing assessments of qualities or attributes and branches representing their results. James et al. (2021) have stated that decision trees may categorize or forecast future instances by recursively scanning the tree's branches depending on the attributes of the input data. One of the primary advantages of decision trees is their interpretability since people can easily perceive and comprehend the tree structure and decision-making processes. Moreover, decision trees can process both qualitative and numeric data and are reasonably quick to train and forecast (James et al., 2021). Figure 18 is an illustrative decision tree and its representation in a two and in a three-dimensional space.

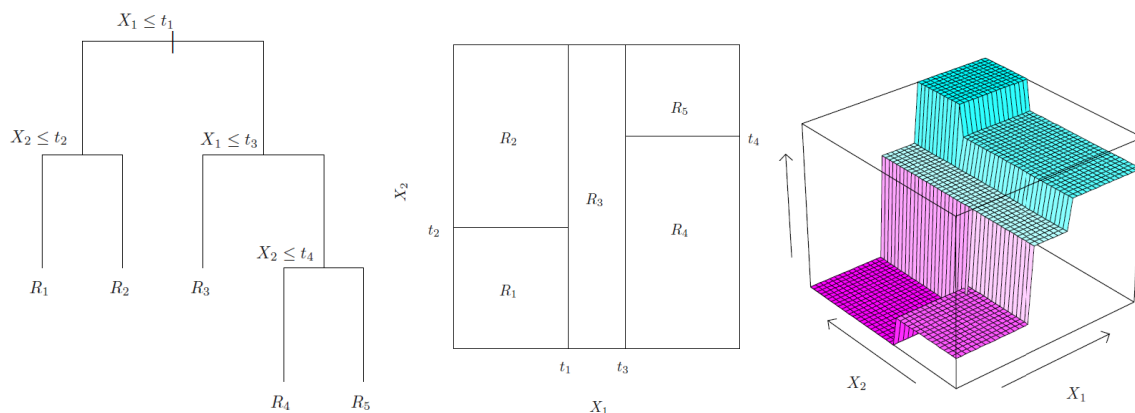


Figure 18. Decision tree example. Adapted from James et al. (2021, p. 332).

The use of decision trees for classification depends on how the data is split through the space in analysis. In Figure 19, James et al. (2021) showed that decision trees split the space into big squares. This method fits better when the dataset can be split according to how the decision tree is built, as seen in the bottom of the figure with the better fit provided by the decision tree on the right. On the other hand, other methods like the SVC provide a better fit for data like the that shown on the top of the figure, as can be seen with the better fit coming from the SVC method on the left.

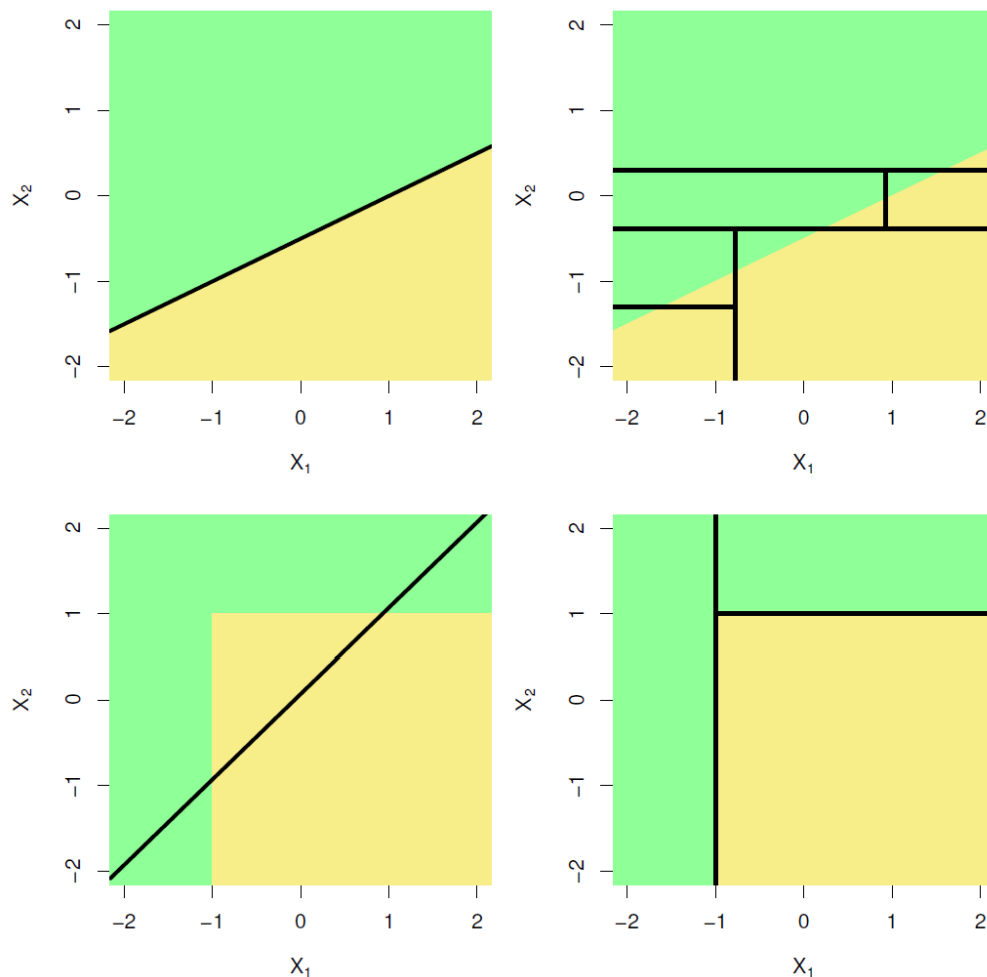


Figure 19. Decision trees versus linear regression classifiers. Source: James et al. (2021).

As with any other ML technique, the decision tree method utilizes a training dataset to build its terminal nodes or leaves, in other words, to divide the space into regions, as shown in Figure 19. The algorithm does that by minimizing the proportion of data points in a category classified in another category. Small values of this proportion indicate that the region predominantly contains observations from that category (James et al., 2021). Thus, an important parameter to be defined when performing this classification is how deep the tree should go, in addition to determining how many regions the space should be divided into.

Researchers have been using the methodology in many fields. Song and Lu (2015) applied the methodology to identify patients with a risk of major depressive disorder. In another application, Liu and Yang (2022) created a decision tree to help bank managers develop marketing content to target their customers more effectively. It is also possible to find logistics applications: Tirkolaei et al. (2021) applied the methodology to build a supplier selection model. Those are just a few recent examples to show that although the technique seems simplistic, it is still valid and helpful.

The following subsections give more details about two popular decision tree-based algorithms that were used to build the models in the present study.

a. Random Forest Classifier

Breiman (2001) defines a Random Forest as “a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.” In essence, a Random Forest involves the creation of a multitude of decision tree models from a singular training dataset, achieved through recurrent sampling of the identical dataset. Each sample has different characteristics, but, as it is from the same population, it has the same distribution. Afterward, there is a voting process for the most predominant way of classifying the data among all trees. Finally, this last model, containing information from all previously built decision trees, is the predictor to classify future observations. A representative illustration of the process is given in Figure 20, in which is drawn from the research by Padovese and Padovese (2019).

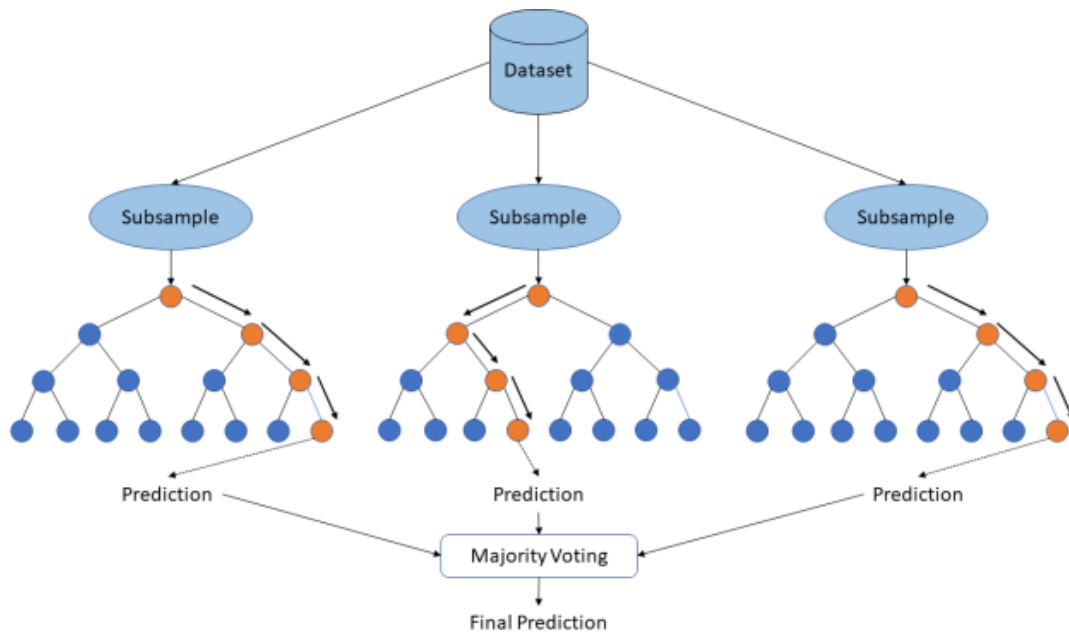


Figure 20. Representative illustration of the Random Forest process. Source: Padovese and Padovese (2019).

According to James et al. (2021), each tree has a lot of variation but not a lot of bias. By combining multiple trees, the overall variance is reduced. This technique has significantly enhanced accuracy, as hundreds or thousands of trees can be combined into a single procedure.

There are many ways of performing the subsampling to create the forest. The Random Forest Classifier (RFC) uses the concept of Random Forest. However, to build the individual trees that compose the forest, this method limits the number of attributes for constructing each tree. For example, if in the training dataset there are p attributes used to classify that data point, each tree created in the RFC will use a number of m attributes which is usually equal to the square root of p . This process reduces the correlation between the trees, contributing to a less variable and more reliable outcome (James et al., 2021).

b. LightGBM

LightGBM stands for Light Gradient Boosting Machines. This is a technique introduced by Ke et al. (2017). According to the researchers, the LightGBM technique was

developed to process large data samples and attributes. Their results showed the technique could perform well with reduced computational speed and memory consumption.

The basic theory behind the LightGBM is the same as that for the Boosting technique, which is another way of constructing the forest. The idea behind Boosting is to sequentially build the trees on different subsets of the training data. The method is discussed in detail by Friedman (2001). The technique iteratively combines multiple weak decision trees' predictors to form a stronger one (Saini, 2021). It starts by training a simple decision tree on the training data. The first tree is used to make predictions on the training data, and the errors or residuals are calculated by comparing the predicted values with the true values (Saini, 2021). It then generates subsequent trees by focusing on the misclassified examples from the previous ones (Saini, 2021). According to Saini (2021) the algorithm adjusts the weights of the training examples, giving more weight to the examples that were misclassified in the previous iteration, forcing the new tree to learn patterns missed by the previous trees. A representation of the process is illustrated by Figure 21; the method was used by Deng et al. (2021) in their research for predicting neonatal jaundice.

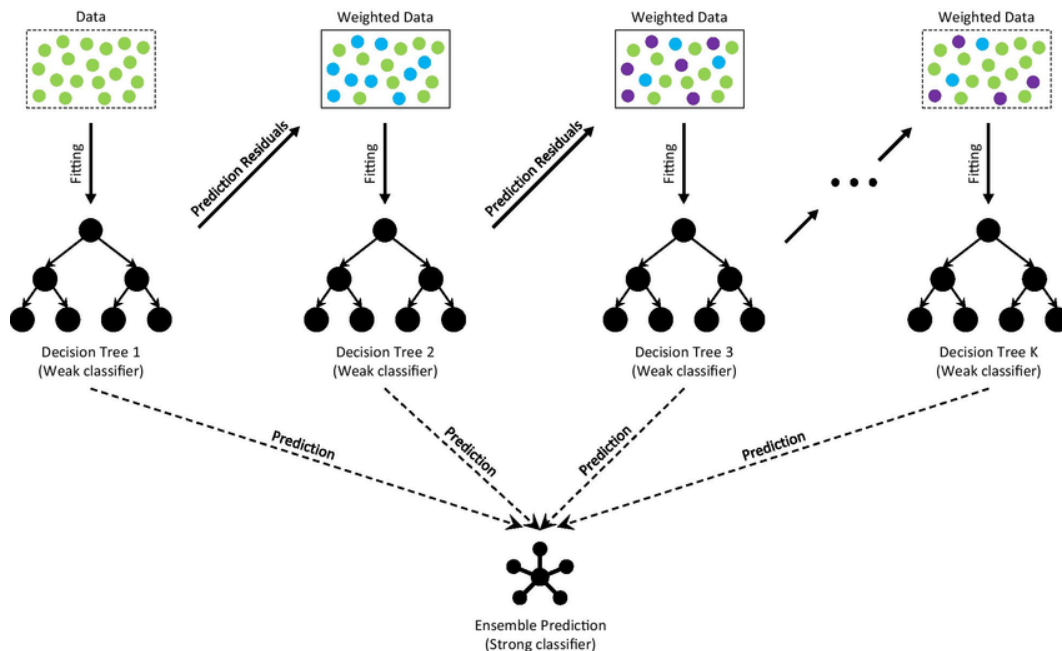


Figure 21. Boosting technique representation. Source: Deng et al. (2021).

5. K-Nearest Neighbors Classifiers

According to James et al. (2021) the K-Nearest Neighbors is an alternative method of calculating the conditional probability of an event given that another event has already happened. In other words, it is an alternative method to apply Bayes' theorem without using the equation already explained in the NBC subsection (James et al., 2021).

This technique classifies a data point according to the classification given to its neighboring data points. It checks the classification of the K closest points around the data point x_0 that is under classification, and this set of closest points is called N_0 (James et al., 2021). It then counts the numbers of neighboring data points in each class, and finally assigns the data point x_0 to the class that had the highest number of neighbors (Steinbach & Tan, 2009).

Mathematically, James et al. (2021) defined that the conditional probability that the data point x_0 belongs to a class j , surrounded by a neighborhood N_0 , with a number of K neighbors, is given by the following.

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

In this equation, the value of $I(y_i=j)$ is equal to 1 if the data point y_i belongs to the class j and equal to 0 otherwise. The KNN classifier assigns to x_0 the class that has the greater probability calculated according to the equation shown (James et al., 2021).

According to James et al. (2021), the choice of the number K can totally modify the way that the KNN classifier approaches a dataset. Figure 22 shows the same dataset classified with different numbers of K . In this image, the black solid line is the KNN classifier boundary.

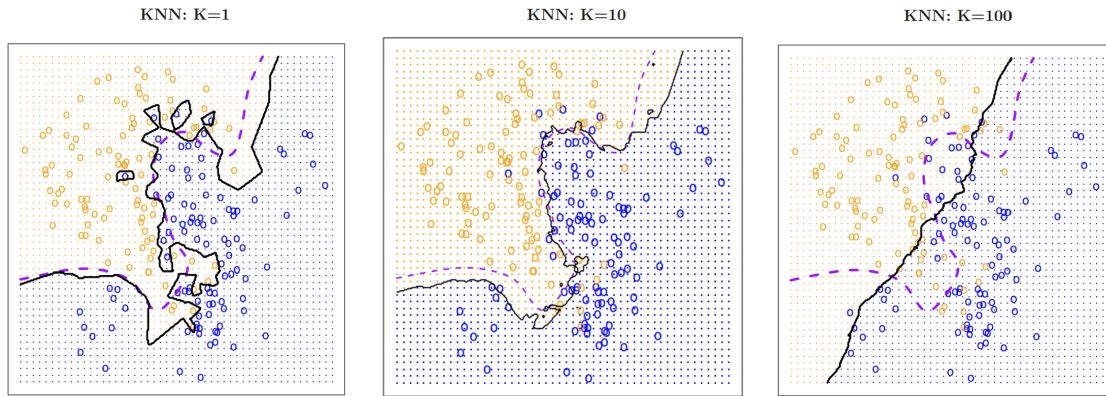


Figure 22. KNN classifier boundary with different values of K . Adapted from James et al. (2021).

Steinbach and Tan (2009) added that the wrong choice of K can lead to odd classifications. Figure 23 shows an example of this odd classification. On the left side of the figure, the data point “X” would be just a noise because there is no neighbor data point in the KNN boundary, and on the right side, the data point “X” would be labeled as “-,” because the boundary is so big that it includes many data points that should not be included. The image in the middle shows a correct choice of K .

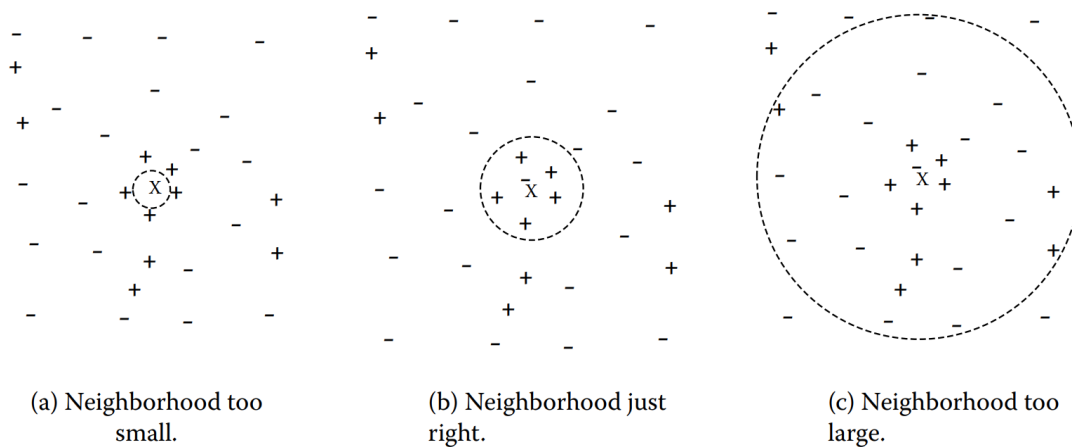


Figure 23. Consequences of different choices of K in a KNN classifier.
Source: Steinbach and Tan (2009).

The KNN technique can be used for both classification and regression problems (Violetto & Noro, 2019). Violetto and Noro (2019) used the technique to forecast the energy demand of an industrial facility. In a different application Mohit et al. (2021) applied the method in an attempt to diagnose diseases in patients. One way or another, the method has proved to be a good ML technique worth testing.

6. Machine-Learning Model's Performance Measures

Performance measures are an essential aspect of evaluating the effectiveness and accuracy of ML models. These measurements provide a quantitative assessment of how well a model is performing and help compare the performance of different models. The choice of performance measures depends on the specific problem, as different applications require different evaluation metrics. In classification problems, measures such as accuracy, precision, recall, F1-score, and confusion matrix are commonly used. The following paragraphs discuss the measures used to assess the created models.

Choosing the correct measure is especially important when the data analyst deals with a classification problem containing an imbalanced dataset. According to Yang and Wu (2006), this is one of the ten most challenging issues related to machine learning. Bekkar et al. (2013) give an example where the goal is to discover a fraudulent case among a dataset with only 1% of the population as a fraud. The authors state that a model that predicts zero fraudulent cases in the data being analyzed will achieve an accuracy of 99%. However, the model could not detect fraud and did not reach the goal desired by the analyst. Therefore, analyzing the model's accuracy in conjunction with other metrics is crucial to provide a better model performance measure.

In a classification problem with two possible outcomes, one way of visualizing the results is by building a confusion matrix. The matrix under consideration comprises two columns and two rows, with the rows representing the actual classifications of the dataset and the columns encapsulating the model's predictions. This matrix is represented in Table 1.

Table 1. Confusion matrix. Adapted from Bekkar et al. (2013).

		Model	
		Predicted Negative	Predicted Positive
Dataset	Actual Negative	TN (# of True Negatives)	FP (# of False Positives)
	Actual Positive	FN (# of False Negatives)	TP (# of True Positives)

Accuracy is given by the sum of the total true predictions divided by the total predictions, as in the following equation.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Recall measures the accuracy of the positive predictions (Bekkar et al., 2013) among all the actual positive data points. In some instances, for example in medical diagnostics, it is crucial to have a high recall in order to prevent ignoring prospective cases of a disease (False Negative), even if this results in the identification of some healthy individuals (False Positives).

$$Recall = \frac{TP}{TP + FN}$$

The precision metric serves as an indicator of the model's degree of accuracy regarding its positive predictions among all the data points that were predicted as positive. Precision is recommended in cases where the cost of a False Positive is high. For example, within a legal context, the repercussions of wrongly convicting an innocent person (False Positive) are considerably more significant than erroneously declaring a guilty individual as innocent (False Negative).

$$Precision = \frac{TP}{TP + FP}$$

To avoid problems such as the imbalanced dataset, another measure utilized is the F1-score, which is given by the following equation (Silva et al., 2021). The F1-score considers both precision and recall, providing a balanced evaluation of the model's ability to correctly identify positive instances while taking into consideration the impact of false positives and false negatives. As previously explained, and as can be seen in the respective formulas, recall doesn't consider False Positives, while precision doesn't consider False Negatives. The F1-score take into consideration both.

$$F_{1-score} = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

It is evident that the True Negatives (TN) do not contribute to this metric, primarily for two reasons. Firstly, in binary classification scenarios, it is often more crucial to accurately identify positive cases than negative cases. Secondly, in non-binary classification problems, in which there are more than two possible classes, the existence of True Negative (TN) instances is absent. Consequently, the application of the same metric in both cases is feasible.

To address the lack of TN in the F1-score calculation, a modified version of the F1-score known as the F1-score macro average has been proposed. As reported by Silva et al. (2021), the F1-score macro average, referred to as F1-macro for brevity, is computed by evaluating the F1-score metric alternatively for confirmed and unconfirmed failures as the positive class (the reference class for calculating Precision and Recall), followed by calculating the arithmetic mean between the two values. The F1-macro score provides an overall performance measure that is impartial towards any specific class and is particularly beneficial for datasets that suffer from imbalanced class distribution.

These are the metrics used to assess the developed model in the analysis.

E. NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is a subfield of artificial intelligence concerned with how computers interact with human language. Simply put, it is the capacity of machines to comprehend, analyze, and produce human language. In contrast to computer

languages and mathematical notations, natural languages such as English, Hindi, and Portuguese are intricate and change with each generation. NLP encompasses a vast array of activities, ranging from simple word frequency analysis to the comprehension of whole human conversations and the production of relevant answers (Bird et al., 2009).

One of NLP's most valuable and well-known applications is the development of chatbots. Recently, there has been a rising emphasis on utilizing natural language to communicate with networked computing devices. Google, Android, and IOS have significantly expanded this technology area (Abdul-Kader & Woods, 2015). Nowadays, it is common to see chatbots in use in the communication strategy of many big companies. Understanding customers is crucial in today's market, and NLP and machine learning have made remarkable advancements in this complex area (Doshi, 2021).

Apparently, chatbots can be the next revolution on the Internet and the way people look for information. In this field it is worth mentioning the release of ChatGPT, an AI-based chatbot capable of producing correct answers for complex questions, from producing an essay to solving a complete physics problem through a program coded in Python (Wang, 2023). Various companies, including Google, are endeavoring to develop similar chatbots that can compete with ChatGPT. Nonetheless, it is important to note that inaccurate or flawed results, such as a product providing incorrect information when queried about telescopes, could have severe negative consequences. For example, Bard, a potential competitor created by Google, experienced this situation, resulting in a 7.7% decline in the share price of Alphabet, the parent company of Google, on the day following the incident (Kelly, 2023).

In summary, NLP is a valuable way to extract relevant information from a large amount of unstructured data. According to B. Xu and Kumar (2015), unstructured text accounts for around 75% of an organization's information. With this mind, NLP has been used in spell-checking software for computers and mobile devices, for categorizing content such as news articles, assessing sentiment on social media, translating spoken speech to written text, offering text translation tools, and identifying spam and fraudulent emails (Silva et al., 2021).

Unstructured data is poorly defined; there is no label for this data. Basically, such data cannot be organized in a spreadsheet with rows and columns (Weiss et al., 2015). Therefore, a subsequent step of text categorization must be performed. Given that NLP consists of various strategies that can transform text into a representation that allows people to extract meaningful information without reading all accessible datasets, this tool can be used to examine a discrepancy description on an FCDD form in order to determine whether it corresponds to a failure or not.

In Python, the Natural Language Toolkit (NLTK) is the most efficient module for NLP. NLTK is an open-source platform that provides a broad collection of tools and resources for various NLP tasks. The toolkit provides many functions, making it a practical resource for NLP practitioners and researchers. This was the library used to build the models in this analysis.

According to Silva et al. (2021), the most widespread text processing algorithms for modeling are those that translate texts into numerical matrices after utilizing different transformations, which may be employed individually in each entry or tailored to a whole series of them. In the following subsections, the primary strategies utilized in this study's code are discussed in further detail.

1. Tokenization

Weiss et al. (2015) assert that the initial stage in analyzing a text is to break down the sequence of letters into individual words, commonly known as tokens. The authors emphasize that each token represents an instance of a particular type. For example, in the sentence "The deer jumped over the fence," there are two tokens spelled "the," and they are both instances of the type "the."

Further, there is also the concept of term. Manning et al. (2009) acknowledge the removal of certain types during the preprocessing phase, such as the commonly occurring stop word "the." The remaining types after all preprocessing procedures are referred to as terms.

Converting a string of characters into tokens can pose a challenge for computer programs with limited language abilities, as certain characters can either function as token delimiters or not, depending on the application at hand, as noted by Weiss et al. (2015). An illustrative example of such characters is white space characters, such as spaces, tabs, and new lines, which are typically recognized as delimiters and, as such, are not included in the token count (Weiss et al., 2015). An example of a character that does not function as token delimiter is the apostrophe used to indicate possession. For instance, in the sentence “Mark’s company is regarded as one of the finest globally.”

2. Stemming

Stemming is a typical approach used in NLP to reduce words to their basic root form, or “stem,” by eliminating any prefixes or suffixes that might alter the word’s meaning. This normalization approach is vital for several NLP applications, including information retrieval, text categorization, and sentiment analysis. Stemming is very useful when preserving the different grammatical features of words is unnecessary. However, it might reduce the readability of the final word list because the stemmed words may not be real words (Rafail & Freitas, 2020). According to Weiss et al. (2015) stemming reduces the number of unique types in a given text and increases the occurrence frequency of some specific types. Therefore, this step makes comparing and evaluating enormous text datasets easier.

In their work, Silva et al. (2021) observed that some words and their variations could represent important information to classify the data. For example, one of the words in Portuguese that indicates that an item has its TBO expired is “vencer.” The technicians usually write this word in its many different verb tenses and forms, “vence” for the present tense, “venceu” for the past, “vencimento” for the noun form, and “vencido” or “vencida” for the past participle of the verb, among others. The authors observed that applying this technique would reduce all these words to the root form of “venc” and would increase the frequency of this root form in the analysis of all of the discrepancy descriptions available. Therefore, it could evidence a stronger indication of an FCDD that was not representing a failure. The same process occurs for many other tokens.

Further, Weiss et al. (2015) observed that, for algorithms that consider the frequency of a word, the use of this technique is relevant. As explained later, in the case under analysis, the words in discrepancy descriptions are transformed into numbers through the method of Term Frequency – Inverse Document Frequency (TF-IDF), which as the name itself suggests, takes frequency into account. Thus, the stemming process in this case is an important step of the text processing.

According to Koirala and Shakya (2020) this step is done in most NLP applications. The authors noted that this technique has been extensively utilized in the English language. Koirala and Shakya (2020) and Al-shalabi et al. (2022) have developed methods for the application of this technique in analyzing the Nepali and Arabic languages, respectively. These instances exemplify the extension of the technique to languages that present a vastly distinct word representation compared to English. Therefore, this technique has outstanding significance in current NLP applications.

3. Stop Words Removal

According to Manning (2009), “stop words” are words that are very common and extensively used but do not add any meaningful value to the text. Vijayarani et al. (2015) added that stop words make the text bulkier and less attractive to the researcher. These authors noted that eliminating stop words decreases the depth of term space, which facilitates analysis. Therefore, removing these words would increase the efficiency and accuracy of text analysis.

The utilization of the aforementioned technique is prevalent in NLP applications. Notably, Khader et al. (2018) and Patil et al. (2014) employed this methodology in their respective research endeavors to conduct sentiment analysis. Furthermore, Ahuja et al. (2019) highlighted that it constitutes a text preprocessing task for any sentiment analysis undertaking. Given the substantial enhancements that this technique can offer to analysis and its widespread implementation, it is judicious to incorporate it in the evaluation of the FCDDs’ discrepancy descriptions.

4. TF-IDF

TF-IDF (Term Frequency – Inverse Document Frequency) is a metric that considers both the Term Frequency and the Inverse Document Frequency. Each component of this measure holds significant value. Notably, in the quest to analyze unstructured data, researchers have devised methods to assign scores to terms present in the texts under analysis. Manning et al. (2009) proposed that one effective approach is to evaluate the frequency of a term within a given document. It is reasonable to posit that the more frequently a term appears in a document, the greater its importance. Consequently, the Term Frequency (TF) is defined as the number of times a specific term occurs in a particular document. Conversely, the IDF serves as an indicator of the rarity of a term within a collection of documents: in simple words, it measures how rare a term is in this collection. When a term is present in every document of a collection, it signifies that the term lacks rarity and thus merits a low IDF score. In contrast, if a term appears in only a limited number of documents within the collection, it indicates that the term is not commonplace within that particular corpus, thereby justifying a higher IDF score.

As it is reasonable to infer, the TF gives equal importance to any term present in the text. However, depending on the context, some words may not carry as much significance as others. Manning et al. (2009) cited the previously discussed stop words as an example. Another example can be found in the context of the aeronautical industry, given that the term “aircraft” is expected to appear many times in documents, even if it does not bring any significant information. To overcome this issue, one possible approach involves reducing the term score for terms that have a high detection frequency in the documents collection being analyzed (Manning et al., 2009). Therefore, the aim is to reduce the TF by a factor that is inversely proportional to the number of occurrences of the term in all the document collection.

Manning et al. (2009) performed this approach by first defining the document frequency of a term, that is, the number of documents in a collection that include the term. As it was previously mentioned, the aim was to find a weight to scale down the TF. Therefore, the higher the number of documents containing a term, the lower the term should be. Denoting N as the total number of documents in a collection, and $df(j)$ as the

document frequency of term j , Manning et al. (2009) and Weiss et al. (2015) defined the IDF of the term j as follows:

$$IDF(j) = \log \left(\frac{N}{df(j)} \right)$$

Consequently, if a term is scarce in a collection, its IDF will be elevated; otherwise, if a term is common in this collection, its IDF will be minimal.

Finally, combining the definitions just given, the weighted score for a term in a collection, according to Weiss et al. (2015) and Manning et al. (2009), is given by:

$$TF\ IDF(j) = TF(j) * IDF(j)$$

Using an aviation dataset to illustrate the importance of this score, Silva et al. (2021) noted that the term “flight” may often appear in all entries but is irrelevant for data extraction, while the phrase “leak” is likely to be beneficial to a user investigating failures. TF-IDF is designed to emphasize this frequency ratio.

To provide an example for this calculation, Silva et al. (2021) presented a step-by-step procedure that utilizes three samples of FCCD discrepancy descriptions, as shown in Table 2.

Table 2. FCDDs discrepancy descriptions samples. Adapted from Silva et al. (2021).

FCDD 1	Shows a stuck left brake / inefficient right brake
FCDD 2	Shows inefficient right brake
FCDD 3	Shows a flat right tire

Table 3 shows the TF for each term in each FCDD.

Table 3. TF from FCDDs in Table 2. Adapted from Silva et al. (2021).

	Shows	a	stuck	left	brake	inefficient	right	flat	tire
FCDD 1	1	1	1	1	2	1	1	0	0
FCDD 2	1	0	0	0	1	1	1	0	0
FCDD 3	1	1	0	0	0	0	1	1	1

Finally, Table 4 shows the calculation of IDF for each term. It considers that the three FCDD samples represent the whole collection of documents. The table also shows the calculation of TF-IDF for each term within each FCDD.

Table 4. IDF and TF-IDF calculations from FCDDs in Table 2. Adapted from Silva et al. (2021).

	Shows	a	stuck	left	brake	inefficient	right	flat	tire
df(j)	3	2	1	1	2	2	3	1	1
IDF = $\log [3/df(j)]$	0.000	0.176	0.477	0.477	0.176	0.176	0.000	0.477	0.477
TF-IDF FCDD 1	0.000	0.176	0.477	0.477	0.352	0.176	0.000	0.000	0.000
TF-IDF FCDD 2	0.000	0.000	0.000	0.000	0.176	0.176	0.000	0.000	0.000
TF-IDF FCDD 3	0.000	0.176	0.000	0.000	0.000	0.000	0.000	0.477	0.477

In Table 4 the green color highlights the most important terms within this collection. The terms “Shows” and “right” have small significance as they appear in all documents within the collection. Conversely, the terms “stuck,” “left,” “flat,” and “tire” appear just in one of the documents within this collection and, consequently, they have higher TF-IDF scores.

This technique is remarkably useful for building ML classifier models based on unstructured data. Willianto et al. (2020), Rahman et al. (2021), and Pimpalkar and Retna Raj (2020) are some of the authors who developed ML classifier models based on TF-IDF scores of the text in analysis. In conclusion, TF-IDF helps extract meaningful information from large datasets and aids in improving the accuracy of data analysis, especially of unstructured data.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

The methodology utilized in this research is divided into two primary components. The first is conceptual, consisting of a comprehensive study of documents from the Brazilian Air Force and an extensive review of academic literature concerning machine learning, which comprises much of what is presented in Chapter II. The second component is practical, wherein the developed program was built and employed to classify new data, and the information classified by the program was then compared to the classification performed manually. The following subsections explain the practical aspects of this methodology in detail.

A. DATA PREPROCESSING PROCEDURES

Two preparatory steps were taken to apply machine-learning techniques to the data: data collection and data preparation. The latter included data cleaning and data classification procedures. The following subsections provide a detailed overview of each task and the corresponding processes involved.

1. Data Collection Procedures

The data utilized for analysis in this study was obtained from the SILOMS database, which is the main repository for BrAF information. More specifically, FCDDs electronically recorded in the SILOMS database were selected for analysis. The FCDDs referred to the following three categories of aircraft: training, transport, and attack. The data retrieved encompassed 2018 to 2021, inclusive. The retrieval process was accomplished by utilizing two filters, one for specifying the project (which indicates the aircraft type) and another for specifying the year. Consequently, each combination of filters by aircraft type and year provided a .csv file containing all the FCDDs for that particular aircraft type and year, making a total of 12 files.

For this study, we utilized the training aircraft data from 2018 to 2019 previously cleaned and classified by Silva et al. (2021) as 2 of our 12 data sets. Apart from that, the

remaining 10 other datasets, consisting of various aircraft types and years, were preprocessed according to the steps described in the following subsections.

2. Data Cleaning Procedures

The first step was to eliminate the many duplicate rows from the retrieved raw data, which is referred to as the data cleaning process. This issue arises due to the recording of an FCDD for a system that can be controlled by more than one parameter. For instance, an aircraft can be tracked based on flight hours, number of landings, or time since last programmed maintenance, resulting in the recording of one FCDD for each control type in the SILOMS database. Removing duplicates was necessary to prevent the trained model from considering the same failure more than once.

In order to expedite the data cleaning process, a Python program was developed. The program read the data file and subsequently removed FCDD rows that contained identical values in columns: “defect item part numbers,” “defect item serial numbers,” “year,” “month,” “date,” “time,” and “discrepancy descriptions.” Only the first row with these shared characteristics was retained while the remaining duplicate rows were deleted. Data cleaning procedures were applied to all datasets, except for the `attack_2020` and `transport_2020` datasets, using the aforementioned Python program. The forthcoming subsection will elucidate the rationale behind the different approach taken for cleansing the data within the `attack_2020` and `transport_2020` datasets.

Although the cleaning process was primarily conducted as described in the preceding paragraph, there were still instances where duplicate rows persisted. Specifically, some FCDDs had similar or identical discrepancy descriptions and were recorded at times that were in close proximity to each other, but one referred to a system while the other referenced a higher assembly part number. Additionally, there were cases where the discrepancy descriptions, part numbers, serial numbers, and dates were comparable or even identical, but the recorded times were less than 60 minutes apart. Both situations were considered as duplicates in this study, and just one of them remained in the cleaned and classified data.

3. Data Classifying Procedures

The second step involved analyzing the discrepancy description of each FCDD and categorizing it as a failure or non-failure. This was achieved by adding a new column to each file called “Validation.” If the word “Yes” appeared in this column, the FCDD was considered a failure. Conversely, if the word “No” appeared, it was not deemed a failure. Thus, each FCDD in the file now had an additional feature recorded in the “Validation” column. This process is referred to as the data classification process and was done manually for all datasets used in this research. This step was essential to train the models. Additionally, this classified data was used as the basis of comparison to measure the performance of the classification made by the machine-learning models trained. Once all the data was cleaned and classified, all remaining duplicate entries were removed. Table 5 presents key details regarding each data file.

Table 5. FCDDs in each data file.

Data Name	Original FCDD quantity	FCDD quantity after the cleaning process	FCDD quantity after classifying process (% of initial quantity)
attack_2018	5,685	2,809	2735 (48.1%)
attack_2019	9,283	2,811	2771 (29.9%)
attack_2020**	9,204	2,438	2438 (26.5%)
attack_2021	11,607	3,008	2937 (25.3%)
training_2018*	-	3,907	3907 (-)
training_2019*	-	3,409	3409 (-)
training_2020	22,472	4,463	4380 (19.5%)
training_2021	28,046	5,119	5006 (17.8%)
transport_2018	10,731	3,226	3112 (29.0%)
transport_2019	10,780	3,230	3119 (28.9%)
transport_2020**	8,266	2,942	2942 (35.6%)
transport_2021	7,724	2,757	2684 (34.7%)
TOTAL	123,798	40,122	39440 (31.9%)

*Data retrieved from previous work of Silva et al. (2021).

**Data manually cleaned and classified.

In order to evaluate the efficiency of the code in terms of time savings, the two datasets mentioned previously (attack_2020 and transport_2020) were cleaned manually as well as being classified manually. During this manual procedure, the time taken to clean and classify these two datasets was recorded and subsequently compared with the time taken by the model to clean and classify a dataset of similar size. This comparative analysis provided a quantifiable measure of the potential acceleration in the failure data classification process that could be achieved through the utilization of the code, as compared to manual classification.

The manual classification process involved subjective decision-making on the part of the individual responsible for this task. As previously mentioned, this is the standard approach utilized by the BrAF and, for the purposes of this study, it is assumed that the data classification was accurate. To facilitate future replication of the research, Table 6 provides examples of common discrepancy descriptions and their corresponding classifications.

Table 6. Common descriptions of discrepancies and their classifications.

Discrepancy Description	Classification
Vague descriptions, such as “item has a breakdown.”	Failure
Descriptions that indicate that the system does not work as it should, such as low efficiency or moving with resistance.	Failure
Any message that appears to the pilot.	Failure
Tires that need to be changed because they have achieved their minimum thickness.	Not Failure
Descriptions that detected a failure of a subsystem during its installation in the aircraft.	Not Failure
Battery discharge.	Not Failure
Corrosion.	Failure
Descriptions indicating noises.	Not Failure
Any kind of leakage.	Failure
Cleaning, tests, expired items, assembly, and disassembly.	Not Failure
Indications of low oil levels in actuators.	Failure
Canopy seal damage.	Failure

B. NATURAL LANGUAGE PROCESSING PROCEDURES

At the outset, the intended approach for this study was to employ the exact code utilized in the research of Silva et al. (2021) to construct the ML models. However, it became apparent that certain adaptations would be necessary, including modifications to the file path for loading the data, changes to the code due to updates in Python libraries, and adjustments to the general process to enable the code to function appropriately. Consequently, to gain a better understanding of each step and prevent modifications that could lead to errors, a new code was developed based on the previous code. Fundamentally, the same steps employed to generate the classification model were followed and are elaborated upon in the following paragraphs.

To begin the model building process, the first step was to read and load the data. During this process, the data in the “Validation” column, which contained the values “Yes” or “No,” were modified to the values “1” and “0,” respectively. This facilitated the use of the data in the ML models, which typically require binary inputs.

To proceed with the data preprocessing, the next step was to concatenate, or join, the data from each file read into a single dataset. During this process, all columns from each file were deleted, except for the “Discrepancy” column (which contained the discrepancy description) and the “Validation” column, as they were the only ones needed for the model building process.

Following those steps, the text in each discrepancy description row passed through the following procedures, which were the same ones employed by Silva et al. (2021):

- Any number followed by letters was considered a measure, and it was replaced by the word “measure.”
- Words that mixed letters and numbers were replaced by the generic term “SNPNPUB,” which indicates the presence of serial number (SN), part number (PN), or technical publication.
- All characters were converted to lowercase characters.

- All the special characters like “%,” “\$,” etc., were removed.
- A word-by-word stemming was applied in order to reduce the words to their radicals.
- All stop words were removed.
- Texts were transformed into numerical columns by the TF-IDF method. Only radicals that appeared at least four times were used in TF-IDF in order to avoid the model adjusting to infrequent words.

C. MODEL CONSTRUCTION AND TRAINING

The process of generating the models was done using the scikit-learn library from Python. Basically, in this library it is possible to generate a model using the technique chosen, its hyperparameters (explained later in this subsection), and a training dataset. The model construction process involves selecting a technique and training the models accordingly. The process of training consists of running the technique chosen on training data, which also output the best set of hyperparameters. Additionally, it is possible to create ensembles, which are a combination of techniques often chosen based on the performance of the trained single-technique model.

The initial step in the construction of the ML models involved partitioning the data into training, validating, and testing subsets. The nine datasets comprised of data for the three aircraft types from the years 2018, 2019, and 2020, were segregated for the training and validating phases. A total of 80% of the FCDDs from each of these datasets were utilized to train the models’ performance, with the remaining 20% used to validate the models. The “train_test_split” function from the scikit-learn library in Python performed this task, which randomly splits a given dataset into training and validation subsets based on a percentage defined by the user. This process is essential as it ensures that the model is validated with new data which was not used to train the model. Therefore, a model’s accuracy and generalizability to new data can be assessed. The three remaining datasets, encompassing data for the three aircraft types of the year 2021, were reserved to test the final model.

To train the models, this study replicated the approach adopted by Silva et al. (2021), training a different model for each of the six ML techniques discussed in Section II.D. Each ML technique involves hyperparameters that must be selected through a process called “tuning.” The hyperparameters tuned in this process were described by Silva et al. (2021) as outlined in the following paragraphs for each respective technique. For each technique, the tuning process involved testing several different parameter values specified as follows.

1. Logistic Regression: tested C (inverse of the regularization factor) values from the set $\{0.01, 0.1, 1, 10, 100\}$.
2. Support Vector Classifier (SVC): tested C (inverse of the regularization factor) values from the set $\{0.01, 0.1, 1, 10, 100\}$.
3. Multinomial Naïve Bayes Classifier: tested alpha “ α ” (smoothing parameter) values from the set $\{0.01, 0.1, 1, 10, 100\}$ (this hyperparameter was introduced in the literature review in Chapter II).
4. Random Forest Classifier (RFC): tested $n_estimators$ (number of estimators, that is, the number of decision trees) values from the set $\{10, 25, 50, 75, 100, 200\}$ and max_depth (maximum depth of the trees decision) values from the set $\{10, 20, 30\}$.
5. LightGBM: tested $n_estimators$ (number of estimators, that is, the number of decision trees) values from the set $\{10, 50, 100, 200\}$.
6. K-Nearest Neighbors: tested $n_neighbors$ (number of neighbors considered to build the classifier) values from the set $\{1, 10, 25, 50, 100, 500\}$.

More detailed explanations of the hyperparameter “ C ” for the logistic regression and SVC can be found in James et al. (2021).

The process of selecting hyperparameters was performed using the “RandomizedSearchCV” function from the scikit-learn library in Python. This function conducts a randomized search over the specified hyperparameter space and selects the best set of hyperparameters based on a chosen score measure. The hyperparameter space for

each technique in this research is the set of values presented above. The choice of score measure is decided by the programmer and is used to evaluate the performance of the model with different hyperparameter combinations. To perform the hyperparameter selection, the score measure adopted was the F1-score, for reasons that are detailed in Chapter IV.

The process of hyperparameter selection uses the training dataset to verify which set of hyperparameters performs the best among the ones chosen to be tested. Therefore, selecting the hyperparameter values and training the model is done simultaneously. Following the same steps described previously, the code trained predictive models using the following techniques: Logistic Regression, LightGBM, Multinomial Naïve Bayes Classifier, Random Forest Classifier, Support Vector Classifier, and K-Nearest Neighbors.

In their work, Silva et al. (2021) created an additional model based on an ensemble. To build the ensemble the authors chose the three models which achieved the better F1-score measures, which were SVC, MultinomialNBC, and logistic regression. Pedregosa et al. (2011) stated that the objective of ensemble methods is to improve the universal applicability and ruggedness of a single estimator by integrating the predictions of numerous base estimators developed with a specific learning technique. The fundamental goal is to develop a classification model that is more precise and consistent than any particular base estimator (Pedregosa et al., 2011). To implement the ensemble approach, Silva et al. (2021) employed the Voting Classifier, which integrates different ML classifiers that have distinct concepts. The predicted probabilities from each classifier are then averaged to determine the class labels, as described by Pedregosa et al. (2011). More details about the ensemble can be found in the Scikit-Learn documentation web page (Scikit-Learn, 2023).

In the present study, a model incorporating a Voting Classifier ensemble was constructed using the same classifiers utilized by Silva et al. (2021), namely SVC, MultinomialNBC, and logistic regression. A second ensemble model was also developed, using the three classifiers that exhibited superior performance in the current study, with the aim of comparing the outcomes and ascertaining the extent of performance enhancement, if any.

After training the models, the next step was to compare them using various performance measures, which are described in detail in Chapter V. The best-performing model was selected and used to classify the remaining data. The time spent performing this classification was also measured.

These were the steps conducted to build and assess the models. Essentially, the same steps conducted by Silva et al. (2021) were replicated with additional datasets. Chapter IV details some complementary and additional steps that were taken. The methodology described in this chapter, plus the added actions described in Chapter IV, were crucial to constructing the answer to the main research question of the study, and are detailed in Chapter VI.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DIFFERENCES IN THE CODE UNDER ASSESSMENT

During the course of the research, some areas for improvement were identified in the code developed by Silva et al. (2021). First, the data quantity and variety served as the primary motivation to initiate this research. Second, it was noted that Silva et al. (2021) did not attempt to build a K-Nearest Neighbors (KNN) classifier, even though it is one of the most commonly used techniques for classification problems, as supported by other authors (Do, 2022; Keita, 2022; Ray, 2017; Wolff, 2020). Third, in Silva et al.'s (2021) study the researchers used the F1-macro as their main performance measure to evaluate the models; however, considering the features of the data in the present work, the F1-score might be more appropriate. This chapter provides a detailed account of such differences between the current study and Silva et al.'s (2021) research.

A. DATA COMPARISON

Figure 24 shows details about the data used in the present work, with each bar representing a different year of data for a different type of aircraft. The blue portions of the bars in the figure represent the number of FCDDs in the data file that were manually classified as failures, while the gray portions represent the number of FCDDs that were manually classified as non-failures. This color division in each bar provides a visual depiction of the proportion of the dataset that corresponds to failures, in other words, how balanced or unbalanced the dataset is. Each bar's height provides the total count of FCDDs analyzed in the respective dataset. It is worth remembering that a portion of this data consists of all the data employed by Silva et al. (2021), namely the datasets "training_2018" and "training_2019."

Some important differences can be noticed between the data analyzed in the work developed by Silva et al. (2021) and the present work. The first and most prominent is the quantity. The present study utilized a total of 28,813 FCDDs to train and validate the ML models developed. If the data used for testing the final model is included, this number would increase to 39,440 FCDDs. In comparison, the number of FCDDs employed by Silva et al. (2021) was 7,316, which represents only 25.4% of the FCDDs utilized in the training

and validation of the models in the present study. It is intuitive to infer that more data can cover a greater number of failure scenarios, and therefore, the model selected in the present work is likely to perform better as the training set is comprised of more FCDDs.

Another notable difference between the two studies is the types of aircraft analyzed. In the present research, data from three different types of aircraft, namely training, transport, and attack, were considered for model training and testing. However, in the previous research conducted by Silva et al. (2021), only data from the training type of aircraft were utilized. The variety in the type of the aircraft analyzed also brings new ways of describing the failures. Moreover, it can cover different systems that exist among the different types of aircraft.

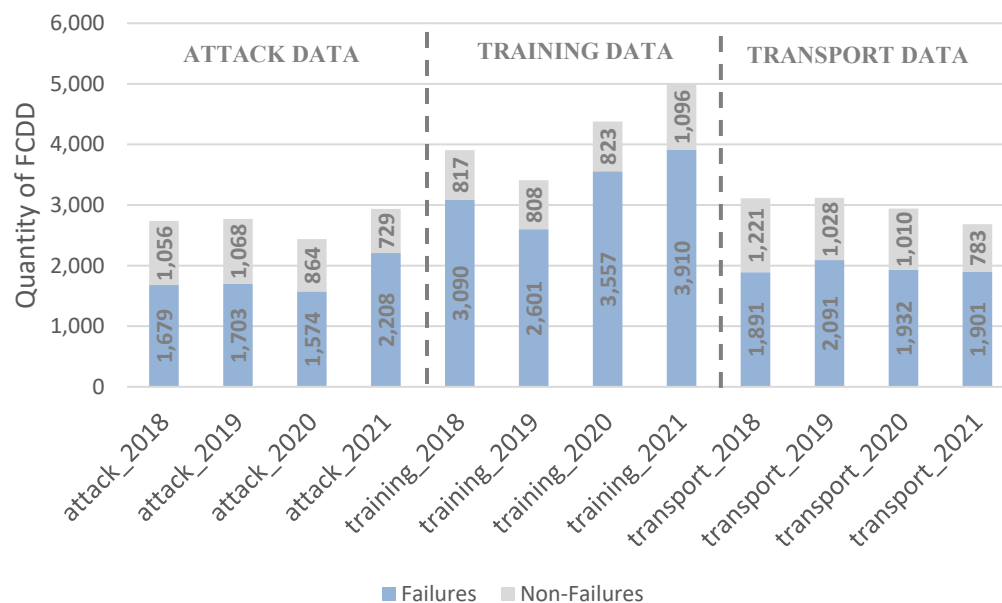


Figure 24. Details about the datasets analyzed, by aircraft type and year.

It is worth mentioning that in this study significant regional heterogeneity was also represented in the analyzed data. Due to its vast size, Brazil presents subtle yet distinct regional variations in the usage of the Portuguese language. As a result, the same failure event may be described differently depending on whether it is reported in the southern rather than the northern region of the country, for example. This regional dispersion is

visually depicted in Figure 25, which displays a map of Brazil with red and yellow markers representing the locations of attack aircraft and transport aircraft, respectively, that were analyzed in this study. The sole green marker represents the only location where the training aircraft type analyzed operates. In the present study, it is evident that the data analyzed encompasses all the macro regions of Brazil, while in the previous study (Silva et al., 2021), only data from one specific region was considered.

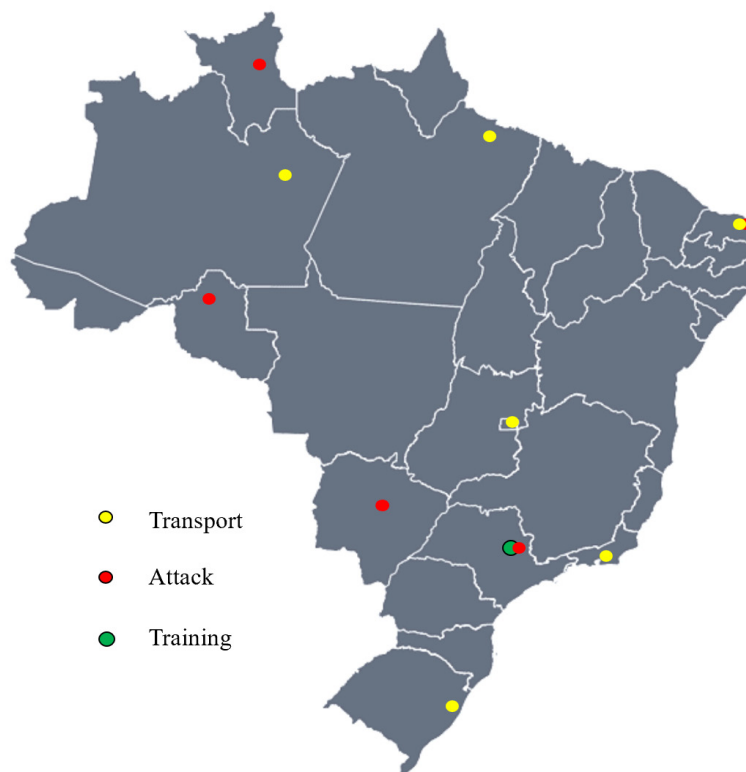


Figure 25. Locations in Brazil where the studied aircraft types operate.

B. ADDITIONAL CLASSIFIER

Silva et al. (2021) opted to assess the effectiveness of four ML techniques, namely SVC, MultinomialNBC, logistic regression, as well as two variations of the decision tree. While the KNN Classifier is also a popular technique for classification problems, it was not utilized in their study. Nonetheless, the rationale for this particular selection of techniques was not explicitly stated by the authors.

In the present research, the inclusion of the KNN Classifier was aimed at covering all potential approaches among the commonly used ML techniques for classification. As will be demonstrated in Chapter V, the KNN Classifier model achieved comparable results to those obtained from the other techniques.

Since one of the objectives of this study was to enhance the knowledge and skills necessary to apply machine learning, the inclusion of the KNN Classifier broadens the potential for future applications and increases the versatility of constructing ML models for any future classification problems that may arise.

C. PERFORMANCE MEASURES USED

Figure 26 offers an alternative visualization of the datasets, showing that the proportion of FCDDs manually classified as failures varies from around 60% to approximately 80% across all datasets. In the present study, considering all data points, 71% of the FCDDs were manually classified as failures.



Figure 26. Proportion of FCDDs manually classified as failures and non-failures, by aircraft type and year.

Upon further analysis of the dataset used in the study conducted by Silva et al. (2021), it was found that the positive class, which refers to the FCDDs classified as failures, constitutes about 78% of the dataset. The authors chose to use the F1-macro measure as the primary performance metric in their study, stating that it would better address the imbalanced nature of the dataset used to develop the models. However, some authors (Lipton et al., 2014; Shung, 2018) claim that the F1-score already addresses this issue.

At this point it is noteworthy to recall the ultimate intended use for the classified data, which is to conduct reliability analysis. In such analyses, the most critical data are the FCDDs categorized as failures. Since failures are considered the positive class, accuracy in identifying true positives is more crucial than accuracy in identifying true negatives. Consequently, as the F1-score is still considered a good “measure to use ... if there is uneven class distribution” (Shung, 2018), and as it gives more importance to the true positives labels than to true negatives (Shung, 2018), in the present study the primary performance metric used in evaluating the models was the F1-score. Using the F1-score instead of the F1-macro would be advantageous as it gives more importance to the class of interest. The F1-score was also the measure chosen to select the models that would be part of the second ensemble model built in this work.

In conclusion, this chapter has highlighted some differences between Silva et al.’s (2021) research and the present study. These differences, including data quantity and variety, the inclusion of a KNN Classifier, and the choice of performance measure, have contributed to a deeper understanding of the classifiers and the failure data in the BrAF database. Future studies can benefit from the noted differences to further enhance the applicability of the classifiers.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS AND ANALYSIS

This chapter is dedicated to presenting the results obtained from the research conducted in this study. It provides a detailed and comprehensive account of the findings obtained from the literature review, focusing on the analysis of BrAF documents and reports. Additionally, this chapter offers an in-depth analysis of the model training and selection process, highlighting the key outcomes and significant findings. Furthermore, it provides a thorough analysis of the main findings of the present research, which contribute to expanding the knowledge related to the field within the BrAF and provide insights that may be useful for future research.

A. FAILURE DATA MANAGEMENT PROCESS

The analysis of the BrAF documentation afforded a comprehensive overview of the procedures pertaining to the registration, storage, and management of their failure data. The aforementioned process is concisely summarized by the schematic representation provided in Figure 27.

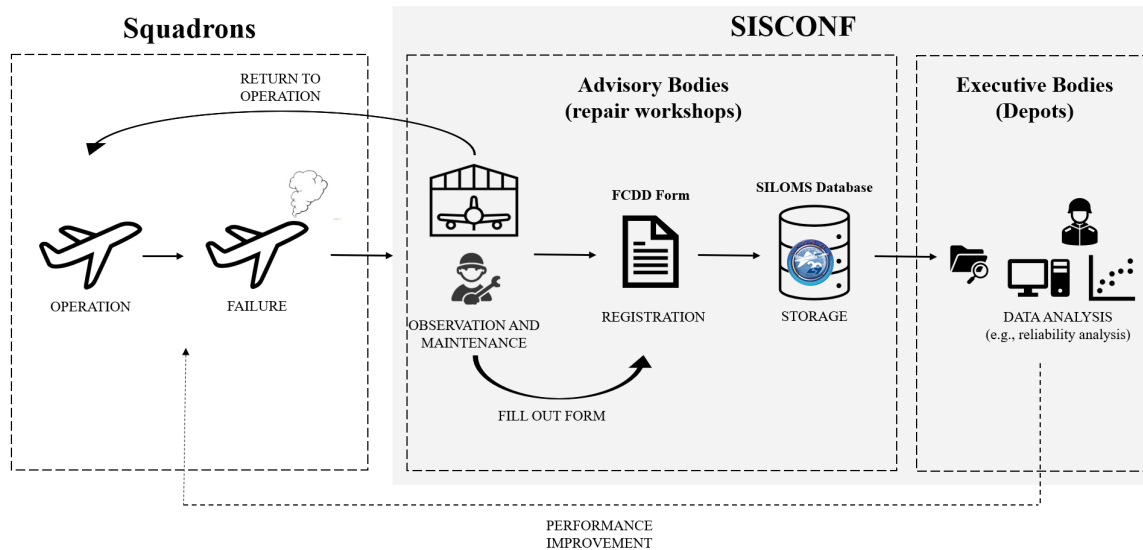


Figure 27. Diagram representing the failure data management process in the BrAF.

The entire process begins in the operation of the systems, specifically the operation of the aircraft, which is conducted in the squadrons. As anticipated, the systems are subject to failures and consequently require maintenance procedures, which are performed by technicians stationed across the three levels of logistics support (first echelon, second echelon, and third echelon). Following the completion of the relevant procedures, the systems are re-established to their operational status. As determined by MCA 66–7 (BrAF, 2017), which was extensively discussed in Chapter II, failures are registered by the maintenance crew filling out the FCDD form. The FCDD is subsequently stored electronically in the SILOMS database, which serves as the principal source for conducting failure data analysis in the BrAF.

The BrAF endeavors to operate, support, and maintain its systems in accordance with the RCM concepts. The regulation ICA 400–21 (BrAF, 2006b) formally establishes the organizational structure, namely SISCONF, to facilitate the application of the RCM concepts. Within this structure, the advisory bodies, consisting of the repair workshops, are responsible for collecting and registering the failures, while the executive bodies are mandated with enhancing the overall performance of the BrAF fleet. The FCDDs stored in the SILOMS database are one of the primary sources utilized to conduct the analysis that is required to attain the goal of improving BrAF fleet performance.

Overall, the established process is sound. Nonetheless, as explained in Chapter II, there are instances, which are even cited as examples in the documents, wherein the FCDDs are recorded even though they do not represent a failure. Moreover, as described in Chapter III, there are instances during the recording process in which one failure results in the generation of multiple FCDDs in the database due to the existence of more than one control type for certain systems. Thus, the data available for analysis by the engineers invariably requires some form of cleaning and filtering to be effectively employed.

Table 7 summarizes the essential criteria BrAF engineers utilize to filter the data. It should be noted that these criteria are subject to variation from one task to another, and there is no universal pattern that is applicable to all situations. Depending on the objectives of each analyst, the criteria are determined and subsequently applied to the data under examination.

Table 7. Criteria utilized to filter failure data.

Criteria	Reports
No filtering	Vieira (2015b) and Vieira (2015c)
Not specified	Vieira (2019), Sousa (2020), Sousa (2021), and Sousa et al. (2022)
Preventive Maintenance	Vieira (2016a), Cavalcante & Ferreira (2015), Silva (2012), Filho & Martins (2016), Chaves (2020), and Silva et al. (2021)
Duplicity	Vieira (2016a), Filho & Martins (2016), Chaves (2020), and Silva et al. (2021)
Poor Description	Vieira (2016a), Filho & Martins (2016), and Silva et al. (2021)
Cannibalization	Vieira (2016a) and Chaves (2020)
Infant Mortality	Vieira (2015a) and Vieira (2016a)

In conclusion, the BrAF has established a well-structured process to manage their failure data, which is essential for improving the overall performance of their fleet. In addition, the SISCONF organizational structure is in place to facilitate this process. The FCDDs stored in the SILOMS database serve as the primary source for conducting failure data analysis. However, there are certain limitations to the process, as highlighted in Chapters II and III of this study. Consequently, to acquire the information necessary to enhance the fleet's performance, BrAF engineers must clean and filter failure data before conducting any analysis. The model developed in this research endeavors to reduce the time required for this cleaning and filtering process, thereby facilitating the improvement of the overall process.

B. MODEL PERFORMANCE AND MODEL SELECTION

Before presenting the findings, it is worth recalling the methodology employed to construct the ML models for classifying the FCDDs. The data utilized for training and validation consisted of FCDDs from datasets of the three aircraft types, covering the years

2018 to 2020. Each FCDD within these datasets was then randomly assigned to the training set with an 80% probability and to the validation set with a 20% probability. Subsequently, the performance of the trained models was evaluated, and the model that exhibited the highest F1-score was selected for further testing. This testing phase involved employing the chosen model to classify the remaining data from the three aircraft types, specifically from the year 2021. It is imperative to emphasize that the process entailed three distinct steps, namely training, validation, and testing. The initial two steps encompassed training the models and also included the utilization of validation data for assessment. All models successfully completed these first two steps. However, only the model with the superior performance proceeded to the final testing step, where it was evaluated against the completely new 2021 data.

One of the steps to build the models was to select the hyperparameters for each technique. The “tuning” process to find each model’s best hyperparameters resulted in the numbers shown in Table 8.

Table 8. Hyperparameters selected.

Model	Hyperparameters
Logistic Regression	C = 10
Support Vector Classifier	C = 1
MultinomialNB	alpha = 1
Random Forest	n_estimator = 200 max_depth = 30
LightGBM	n_estimator = 200
K-Nearest Neighbors	n_neighbors = 25

Hyperparameters are crucial parameters that govern the performance of an ML algorithm. By showing the hyperparameters used in our models, we aim to provide transparency in our experimental setup and enable the reproducibility of our results. Moreover, sharing this information may help other researchers to build on and improve our work.

During the selection of the hyperparameters, models using each of the techniques were trained based on the training set and then validated using the validating set. Figure 28 visually summarizes the performance of these models during the validation step, presenting important evaluation metrics such as F1-score (with failure as the positive class), F1-macro, and accuracy. The chart lists the models in descending order from highest to lowest F1-score, highlighted in blue.

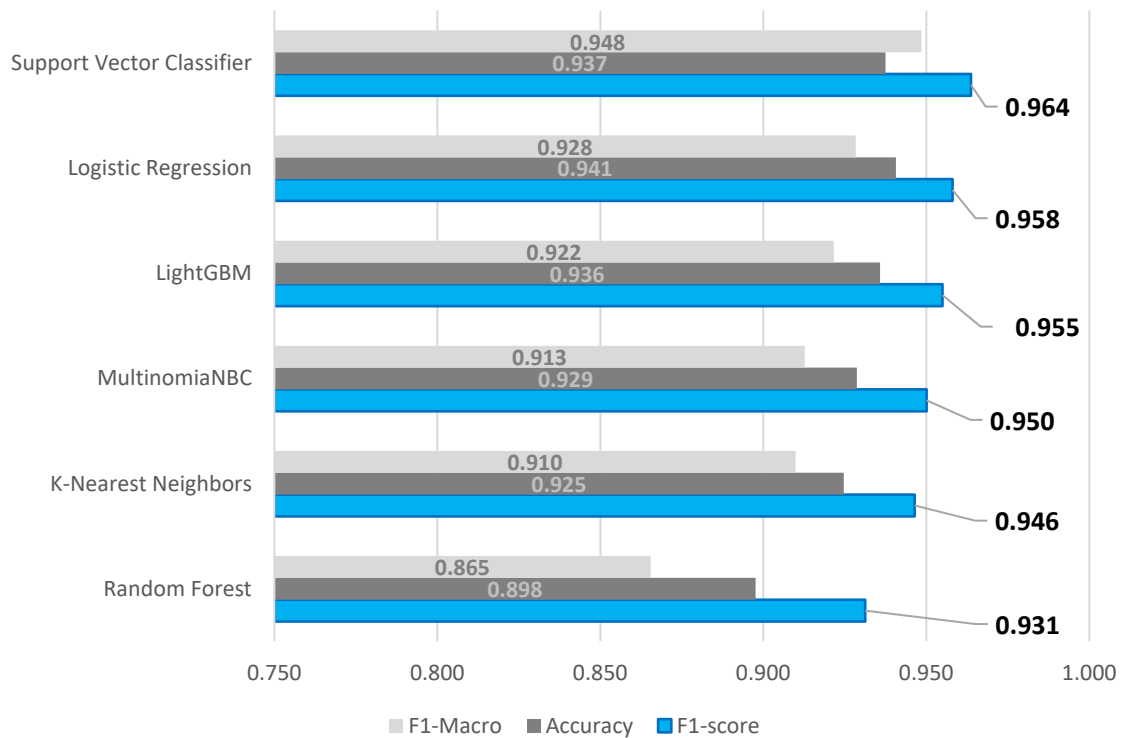


Figure 28. Models' performance.

Upon further examination of the figure, it is appropriate to elaborate on its distinct characteristics. The model for each technique has three performance metrics, namely F1-score, accuracy, and F1-macro. As stated in Chapter IV, the F1-score was the criterion used to identify the best-performing model, and thus it is distinguished in blue, and its values are labeled accordingly. Accuracy remains a crucial metric because as demonstrated in Chapter IV, for this binary classification problem, the degree of imbalanced data is not

substantial. Finally, the F1-macro is utilized to facilitate the comparison of the models developed in this study with the one constructed by Silva et al. (2021).

The examination of the chart provides information that is worth emphasizing. First, except for the Random Forest technique, the models achieved very comparable performance when comparing their F1-scores, with all hovering near a value of 0.95. With the exception of the Random Forest technique, all the other models achieved an accuracy of at least 92%. Overall, the techniques demonstrated very similar results and could be selected for future analysis without significant changes in the results, except for those using the Random Forest technique.

The ensembles, which have the results illustrated in Figure 29, were unable to enhance the performance in terms of F1-score; however, they exhibited superior accuracy, both achieving an accuracy rate of 94.6%. Despite the numerical values depicted in the figure being identical, they exhibit divergence in the fourth decimal place. It is noteworthy, however, that both ensembles demonstrated analogous metric measures.

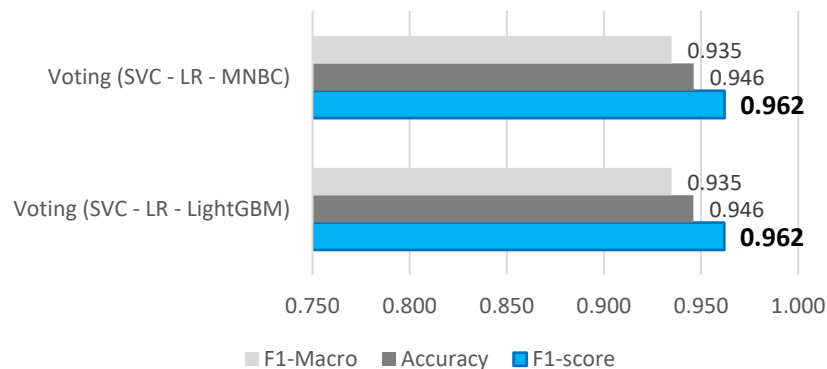


Figure 29. Ensembles' performance.

It is also worth noting that the SVC model achieved the highest F1-score and was thus chosen for testing on the remaining data. Interestingly, the ranking of models based on performance would remain the same if the metric of choice was the F1-macro instead of the F1-score.

As already stated, for comparative purposes, the chart in Figure 28 also includes the F1-macro for each model. The model chosen by Silva et al. (2021) achieved an F1-macro of 0.945 (measure achieved by their chosen model on their validating dataset) whereas the best performing model in this research, the SVC, achieved an F1-macro of 0.948. The SVC model displayed a performance that was closely comparable to the model selected by Silva et al. (2021) in their research, particularly when assessing F1-macro scores.

C. PERFORMANCE OF THE SELECTED SVC MODEL

Figure 30 illustrates the confusion matrix for the SVC model using the validation dataset. The validation set consisted of a total of 5,763 data points.

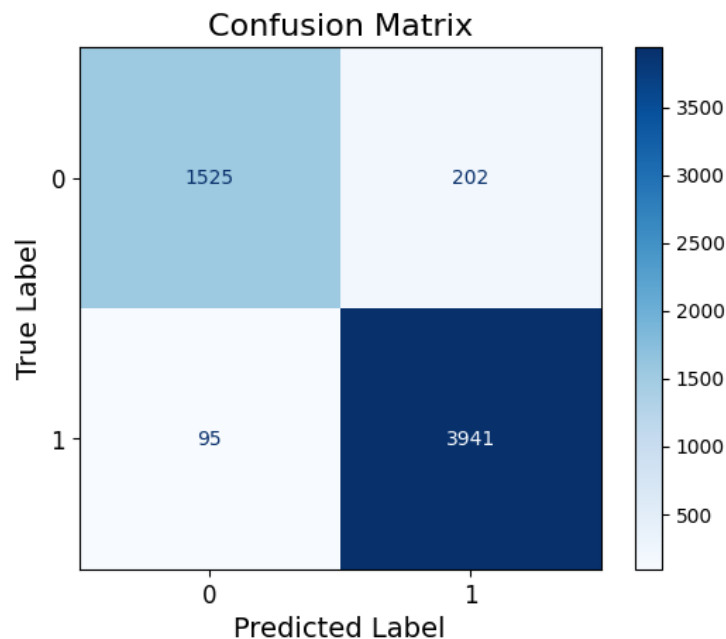


Figure 30. SVC model confusion matrix using the validation dataset.

As anticipated, the SVC model had a higher number of samples in the top-left (correctly identifying non-failures) and bottom-right (correctly identifying failures) quadrants. The preponderance of failures in the bottom-right quadrant is due to the nature of the analyzed data, as demonstrated by the fact that 71% of the data correspond to failures.

The remaining two quadrants contain the misclassified samples. A well-performing model's confusion matrix should have a similar appearance, with dark colors in the diagonal indicating correct classification and light colors elsewhere.

Figure 31 portrays the evaluation of the SVC model on the testing subset, the 2021 data, with results provided for each individual aircraft type as well as for all the aircraft types aggregated together. The chart showcases the F1-score and accuracy values. Once again, the emphasis was given to the F1-score as it was the performance metric used to select the best model. Remarkably, the chosen SVC model maintained its level of performance when applied to entirely new data, both in terms of F1-score and accuracy. Specifically, the model's F1-score and accuracy scores did not show significant changes when compared to its performance using the validation dataset. Therefore, these results indicate the model's proficiency in classifying new data, at least for the specified aircraft types.

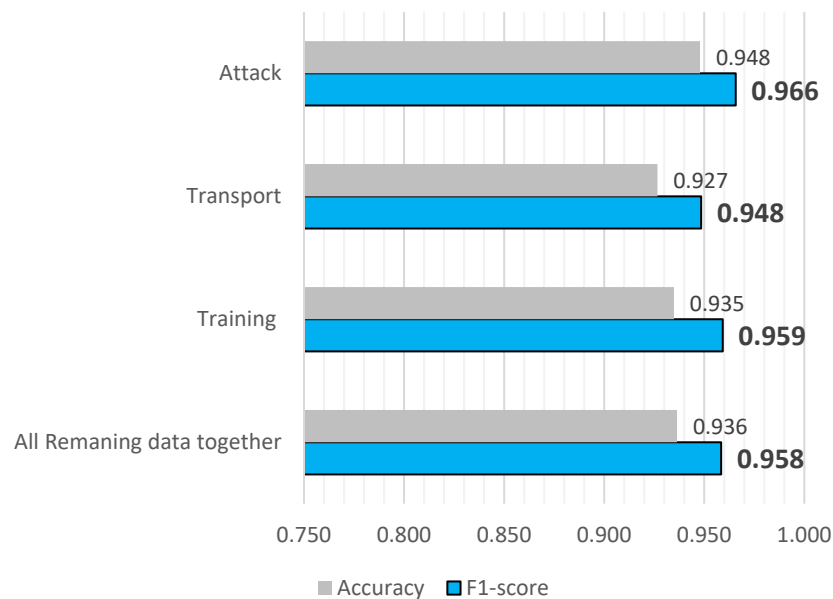


Figure 31. Performance of the SVC model using the remaining data.

D. COMPARATIVE ANALYSIS OF MODELS IN THE PRESENT AND PREVIOUS RESEARCH

To assess the improvement introduced by the present study, the model developed by Silva et al. (2021) was utilized to classify the testing dataset, which constitutes entirely new data. The resulting confusion matrix for both models, the SVC and Silva et al.'s (2021) model, is illustrated in Figure 32. It is evident that the present study has enhanced the performance on the non-failures labeled data. The misclassification of samples by Silva et al.'s (2021) model led to similar quantities in the first row of the confusion matrix, indicating that their model classified a higher number of samples as failures although they were actually non-failures. This significantly impacted the performance metrics, which are depicted in Figure 33.

Figure 33 showcases the models' performance in terms of three performance metrics: F1-score, F1-macro, and accuracy. It is noteworthy that the F1-score was the performance metric utilized to choose the superior model in this study, whereas the F1-macro was the one selected by Silva et al. (2021). As previously stated, accuracy is also a useful metric for evaluating model performance, especially when there are many samples in the top-right quadrant of the confusion matrix, which could negatively impact the F1-macro.

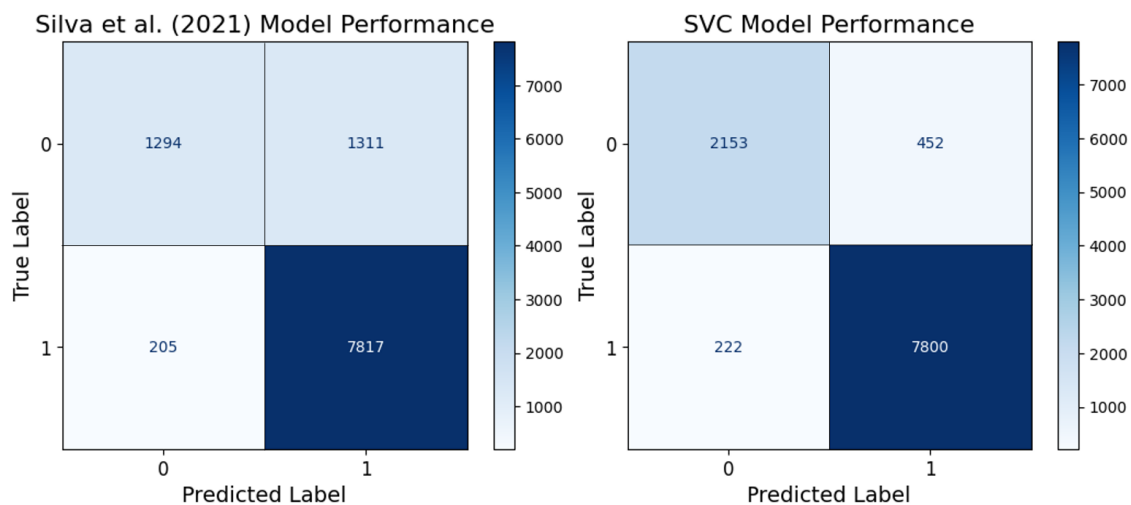


Figure 32. Confusion matrix for the models after classifying the data from the year 2021.

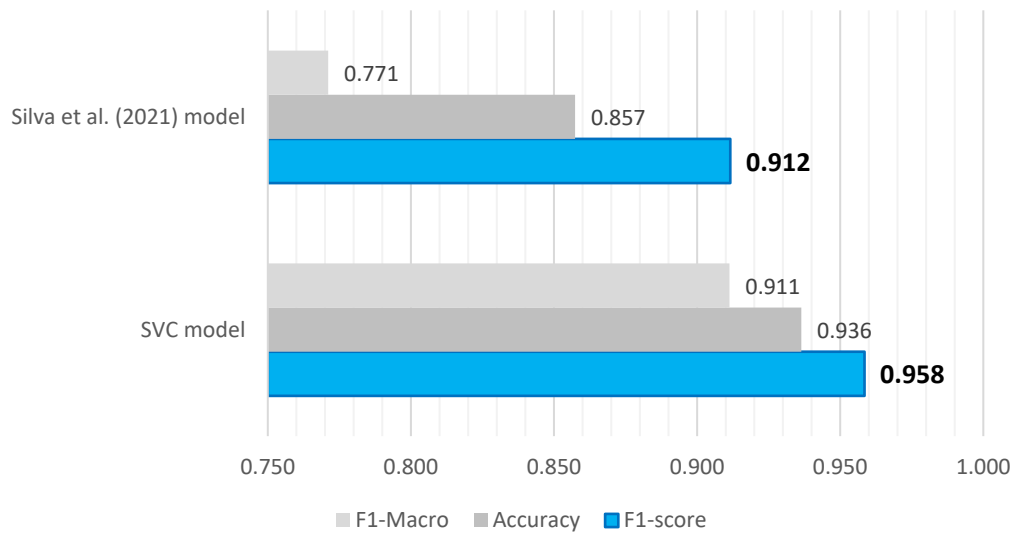


Figure 33. Comparing the performance of the models classifying data from the year 2021.

The model created in this study demonstrated superior performance in classifying entirely new data across all evaluation metrics presented in Figure 33. Notably, the results indicate that Silva et al.'s (2021) model was unable to maintain its performance when applied to completely new data, resulting in a decrease in the F1-macro score from 0.945 to 0.771. This suggests that utilizing their model to classify new data would likely result in an overestimation of the number of failures in the system.

It could be argued that the superior performance of the model developed in the current study was due to the inclusion of other types of aircraft. However, the model's performance in classifying each aircraft type was evaluated separately to address this potential concern. Therefore, the testing dataset was separated by aircraft type, and then both models, the SVC and that of Silva et al. (2021), were used to classify these three separated datasets (attack_2021, training_2021, and transport_2021). The results showed that there were improvements in the performance of the model across all aircraft types. Figure 34 presents the percentage increase in performance metrics comparing the results achieved by the SVC model against the results achieved by Silva et al.'s (2021) model.

In the chart depicted in Figure 34, the blue bar represents the percentage difference between the numbers shown in Figure 33 (SVC model score minus the Silva et al. model score divided by the latter). As can be seen the SVC model performed better than the Silva et al. (2021) model for all types of aircraft on all metrics. Due to the similarities among the attack and training aircraft types and considering that the Silva et al.'s (2021) model was developed using data of the training aircraft type, it was expected that the most significant improvements would be observed in the classification of the transport aircraft type. Nonetheless, it should be noted that even for the same aircraft type used by Silva et al.'s (2021) model, the performance of the SVC model was superior.

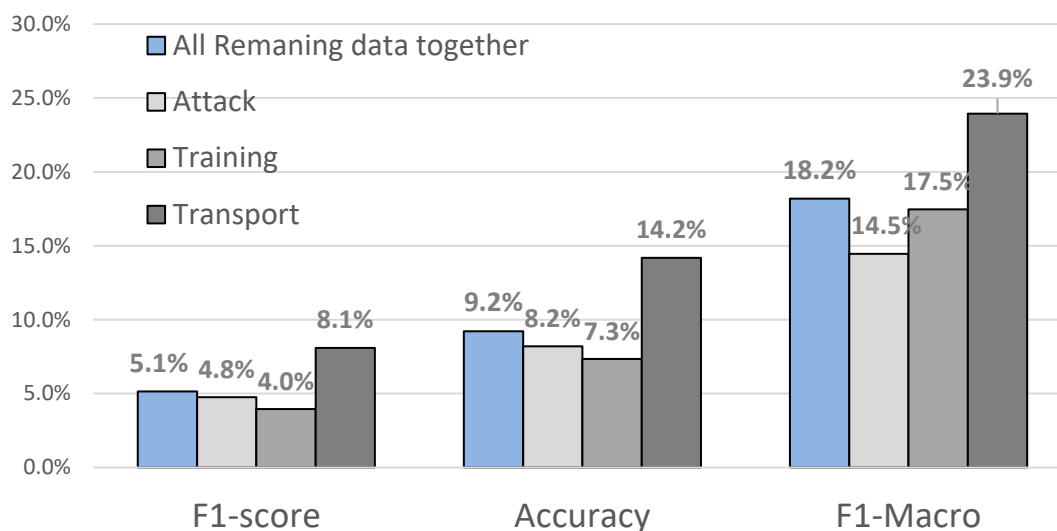


Figure 34. Percentage increase in performance of SVC model compared to Silva et al. (2021) model.

All of the preceding findings support the fact that the current study has significantly advanced the automation of data classification in BrAF.

E. OBSERVATIONS ABOUT AN INITIAL ASSUMPTION

To enhance the quality of the model's performance, an approach that can be adopted is to examine the FCDDs that were misclassified and ascertain the reasons behind

the incorrect classification. This would be especially useful for instances where the model assigned a high probability of an FCDD belonging to one class, but the manual classification placed it in the other class. To assign a class for a specific data point, the model calculates the probability of that data belonging to a defined category, in this case, the likelihood of a data point being a failure. Table 9 illustrates ten cases of FCDDs where the model calculated as highly likely to correspond to a failure but, in the manual procedure, was classified as a non-failure. The cases chosen had the top ten highest probability values of being a failure, as calculated by the model. This table presents the discrepancy description, the model-calculated probability of a data point being classified as a failure, the model's classification, and the manual classification, with an added column showing what should, in reality, be the actual classification based on further investigation.

Table 9. Top 10 FCDDs that have the highest probabilities of being failures.

Translated Discrepancy Description	Probability of Being Failure	Model Classification	Manual Classification	Reality
Inoperative hood.	> 99.99%	Failure	Non-Failure	Failure
Reserve Horizon 2P dead.	99.53%	Failure	Non-Failure	Failure
Micro switch presenting insulation failure.	99.40%	Failure	Non-Failure	Failure
The aircraft after finishing the climb procedure (stabilizing straight and level); it was observed that the needle was marking 6,000 ft for a long time (it was not possible to specify the time).	99.28%	Failure	Non-Failure	Failure
When starting the airplane, the ignition light did not turn on. With the switch on "normal" the vent light on the panel was on during start up; the switch was made to "vent" and then to "normal" again and the light continued on.	99.26%	Failure	Non-Failure	Failure
In the first attempt to activate the rudder trim, it did not respond (1p and 2p). Instantly after it returned to action and during the rest of the flight showed no problem.	99.09%	Failure	Non-Failure	Failure
Flying appropriate with a VOR station having the HSI centered and stable at the station's radial 270°, the RMI remained stable at the radial 260°.	98.76%	Failure	Non-Failure	Failure
No seat handle pin indication.	98.61%	Failure	Non-Failure	Failure
Dent in the left aileron.	98.59%	Failure	Non-Failure	Failure
Cutting rod with internal spring.	98.54%	Failure	Non-Failure	Non-Failure

One of the primary assumptions of the current work was that the manual classification done in the datasets would be considered correct. However, the process of classifying nearly 40,000 FCDDs is time consuming and can be tedious. As humans, we are prone to making mistakes. The “Reality” column was added to Table 6 to show what would be considered the correct classification after a thorough and meticulous review of each description. Surprisingly the table reveals that the model accurately classified nine out of the ten cases displayed, despite being deemed incorrect. In other words, in nine out of the 10 cases shown in Table 6, the manual classification was found to be erroneous. This finding underscores that this primary assumption of 100% accuracy of the manual classifications may not be reliable.

Table 10 presents a similar ‘top-ten’ analysis but focuses on the other type of error that the model may encounter, where the model classified an FCDD as having a very low probability of being a failure while the manual classification indicated it was a failure. As can be seen in the final “Reality” column, the table reveals that the model classification was actually correct in nine out of these 10 cases.

Table 10. Top 10 FCDDs that have the lowest probabilities of being failures.

Translated Discrepancy Description	Probability of Being Failure	Model Classification	Manual Classification	Reality
Perform generator revision (worn brushes).	0.22%	Non-Failure	Failure	Non-Failure
Left landing gear tire with exposed frame.	0.72%	Non-Failure	Failure	Non-Failure
Replace right main landing gear.	1.41%	Non-Failure	Failure	Non-Failure
Corrective Maintenance.	1.83%	Non-Failure	Failure	Failure
Perform inspection to verify the left engine oil system (day 05).	1.92%	Non-Failure	Failure	Non-Failure
Spare and unserviceable materials for aircraft maintained by GLOG 3.	2.93%	Non-Failure	Failure	Non-Failure
Perform cleaning of jet pumps screen filters.	2.94%	Non-Failure	Failure	Non-Failure
Worn brushes.	2.94%	Non-Failure	Failure	Non-Failure
Worn brushes.	2.94%	Non-Failure	Failure	Non-Failure
Worn brushes.	2.94%	Non-Failure	Failure	Non-Failure

In summary, the model, leveraging its construction based on a large dataset, demonstrated its ability to discern patterns and identify failures, and even rectify some of the inaccuracies in the manual classification process. This suggests an enhancement in the quality of the overall classification process.

F. COMPARISONS OF TIME SPENT ON DATA CLASSIFICATION

One of the significant advantages of utilizing an ML model for data classification is the reduction in time required for classification, allowing for more focus on data analysis. The chosen model considerably reduced the time necessary for data classification. Figure 35 presents a summary of the time required for manual classification of the datasets.

The chart illustrates the amount of time taken to classify the data for each dataset. For the attack and transport aircraft type data of 2020, both cleaning and classification processes were done manually. For the remaining datasets, in order to save time as was explained in Chapter III, only the classification process was done manually. All classifications performed by the model took less than one minute, with an average of 15 seconds, and were not represented in the chart.

In the chart the gray bars represent the time spent to solely classify the data manually. The red bars represent the two datasets that were both cleaned and classified manually. The dashed red line is the average time spent on these tasks for those two datasets. The gray dashed line is the average time spent on the classification tasks of the other datasets. The training_2018 and training_2019 are not depicted here because they were cleaned and classified by Silva et al. (2021). In summary, the dashed gray line is the average time to classify one year of data, the red dashed line is the average time to clean plus classify one year of data.

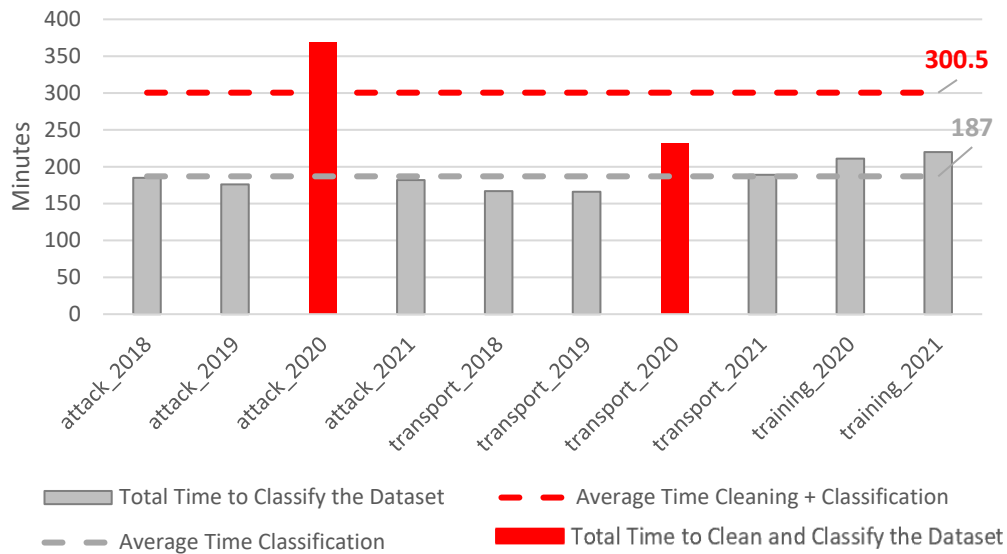


Figure 35. Time spent on manually classifying the datasets.

It is important to clarify that the reported average time of 15 seconds pertains to the time the trained model spent to classify the data. The process of training and validating the models, on the other hand, required a longer duration, ranging from six to eight hours for each model. However, this is considered a one-time cost as the trained and validated model can be used repeatedly without the need for further training and validation. Furthermore, it is worth noting that the training and validation process involves elapsed time of a computer and not human labor time.

Averaging the time for the classification process alone, the model was able to reduce the time spent on this task from over three hours to just 15 seconds for each year's worth of failure data, for each type of aircraft. This underscores the potential of this tool to be a powerful asset for the engineers working in the BrAF.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

The BrAF, like other major flight operators, must effectively manage multiple aircraft fleets, which are costly assets. Incorrect decisions in managing these assets can result in significant financial waste and compromise mission success. Failure data collected from the field regarding these assets serve as crucial information for decision makers to assess system reliability, availability, and maintainability. These metrics aid in better planning for system needs and overall management. However, in order to obtain accurate and reliable information, it is imperative to use high-quality data, particularly in the context of the failure data in this study. BrAF engineers typically preprocess the data by classifying them as failures or non-failures for analysis, but this task is repetitive and time consuming. The purpose of this study was to develop and evaluate a machine-learning model capable of automatically performing this data classification task. Automating this process would minimize the potential for human error, improve the quality of the classified data, and save time for engineers, allowing them to allocate more time to analysis rather than data preprocessing. To guide the study, some research questions were proposed and are addressed in the next paragraphs.

The first question was about understanding the procedures that BrAF uses to manage its failure data, and it was answered through the review of BrAF documentation. The RCM methodology has been widely recognized in the aviation industry for its superior performance in enhancing aircraft fleet operation and maintainability. An analysis of the official instructions and manuals of the BrAF revealed that the organization has an established structure and defined responsibilities for implementing RCM principles, which essentially involves treating and processing failure data. The documentation analysis also indicated that there is an established procedure in place for collecting and storing failure data. However, the process includes cases where data is recorded even if the event is not a genuine failure because it is still deemed necessary to be registered. In fact, it was observed that the BrAF database, from which engineers source their data, contains entries that do not represent failures. As a result, engineers need to filter the data obtained from the source before conducting their analysis.

The second question was related to the procedures that engineers in BrAF follow to process the failure data. A comprehensive review of 14 reliability reports done by BrAF engineers was conducted. The review demonstrated that the procedures vary depending on the goal of each analysis and each engineer. Nevertheless, it is unquestionable that there is a need to filter the data before any analysis.

Given that the task of filtering data is repetitive and relies on observing discrepancies in the failure data descriptions, it is reasonable to expect that a machine-learning approach would be capable of recognizing patterns and performing the task faster and with greater accuracy than manual classification by humans. Silva et al. (2021) developed models using five machine-learning techniques (Logistic Regression, Support Vector Classifier, Multinomial Naïve Bayes Classifier, Random Forest, and LightGBM) to perform this task and achieved promising results. Their study used data from 2018 and 2019 that were taken primarily from training aircraft.

In the present study, a more in-depth exploration of the theory behind each ML technique was conducted to better understand the mechanisms of data classification performed by each method. An exploration of the theoretical foundations and practical applications of each technique employed in the code developed by Silva et al. (2021) addressed the main point of the third and fourth research questions. The K-Nearest Neighbors algorithm was also included as a potential candidate because this method was not considered in Silva et al.'s (2021) research, even though it is among the most used methods to perform classification. Furthermore, the present study incorporated data from two other aircraft types, namely attack and transport aircraft, as well as from the training aircraft type, spanning 2018 to 2021. Chapter IV highlighted some differences between Silva et al.'s (2021) research and the present study. These differences, including data quantity and variety, the inclusion of a K-Nearest Neighbors classifier, and the choice of performance measure, have contributed to a deeper understanding of the classifiers and the failure data in the BrAF database. Future studies can benefit from the already noticed differences to further enhance the applicability of the classifiers. Those incorporated improvements were made to expand the potential application of the selected model to a broader range of possible classifications.

The results of the comparative analysis revealed that out of the eight models constructed, which included six models based on the theories mentioned in the previous paragraph, as well as two different ensembles that combined the aforementioned models, the SVC model yielded the best performance in terms of F1-score. The SVC model achieved an F1-score of 0.964 and an accuracy of 0.937. When compared to the model selected in Silva et al.'s (2021) study, the SVC model developed in the present research outperformed it in terms of both F1-score (0.958 versus 0.912) and accuracy (0.936 versus 0.857) when classifying completely new data for both models.

Furthermore, when analyzing the data of each aircraft type separately, the SVC model still demonstrated superior performance classifying all three aircraft types. Notably, the SVC model was able to accurately recognize more non-failure classifications compared to the model that was built previously, indicating its enhanced ability to distinguish between failure and non-failure events.

The SVC model was able to perform the classification task for one year of data in 15 seconds on average, whereas manual classification of the same data took three hours on average to complete. This finding shed light on the time-saving benefits of utilizing a machine-learning model for classification tasks, thereby addressing the fifth research question, which was related to time spent to preprocess the failure data.

A fundamental assumption in the present research was that manual classification would be considered accurate, providing a baseline for comparison against which the machine-learning model would be trained and evaluated. However, it is well known that the manual classification process can be monotonous and tiresome, and humans are prone to making mistakes. Upon further analysis of the classifications made by the SVC model, it was observed that some of the classifications deemed incorrect in the validation process were actually correct. In these cases, the manual classification was found to be inaccurate.

In conclusion, the findings of this study suggest that machine-learning models, particularly the SVC technique, exhibit significant potential for effectively classifying failure data sourced from the BrAF database, thus providing insights that address the sixth research question pertaining to the reliability of the classification performed by the model.

Furthermore, implementing these models significantly reduces the time required for classification.

It should be noted that the SVC model used in this study incorporated data from various types of aircraft that are distributed across Brazil during its training process. Based on the findings from the comparison of the SVC model and Silva et al.'s (2021) model when analyzing new data for both models, an increase was observed in the overall performance of the model selected in the present research. Given that the model selected by Silva et al. (2021) was trained based on an aircraft that operates in a single region of the country, it shed light on the seventh research question, which was related to the ability of the model to accurately classify data from various aircraft that operate in different Brazilian regions. It demonstrated that having data from aircraft dispersed throughout the country may have influenced the improvement achieved. Nonetheless, it remains uncertain whether this improvement in performance was attributable to the inclusion of different types of aircraft or to the dispersion of data collected across various regions of the country. Future studies could investigate this matter by dividing data from the same type of aircraft across different regions of Brazil and assessing whether significant differences exist in the resulting classification performance.

It is important to note that the results obtained in this research were based on data from three specific aircraft types. Applying the trained model to classify data from other aircraft types may yield different results. Further research and validation on various aircraft types are necessary to determine the generalizability and effectiveness of the model in diverse contexts.

Ultimately, this study succeeded in addressing the primary research question about whether the Brazilian Air Force could enhance its repetitive task of classifying data by utilizing machine-learning techniques. Overall, the results of this investigation emphasize the potential of machine-learning to improve the efficiency and accuracy of failure data classification tasks in the aviation maintenance field. This, in turn, could support the implementation of Reliability Centered Maintenance approaches, ultimately leading to improvements in the operational performance and maintainability of aircraft fleets.

For the Brazilian Air Force, beyond the use of the model to classify failure data, it is worth mentioning two potential branches which, with further slight enhancement, could sharply improve some processes. The first area involves utilizing the foundational ideas of the code to enhance the failure data collection and recording processes. Specifically, the SILOMS database, which records the Defect Data Collection Forms, could analyze the discrepant description before recording it in the database. In cases where the analysis determines that the description on the form does not represent a failure, the system could prevent the maintainer from registering it. This would help ensure the database is filled with accurate and relevant data.

Another area that could benefit from the model's capabilities is inventory management. The Brazilian Air Force currently utilizes the SILOMS system to calculate the replenishment of spares to its inventory, with one of the sources being the Defect Data Collection Form. Using the model to double-check certain items' failure would improve the accuracy in determining needed replenishment. This would be particularly valuable at the lower level of the workshops, as these repair workshops deal with a smaller number of different items. In addition, as is typically the case, each failure precedes a demand for the specific item required for repair, making the model's ability to accurately identify failures a helpful tool to maintain optimal inventory levels.

Future enhancements to the present research can potentially increase the generalizability and effectiveness of the machine-learning models in diverse contexts. Some of these are listed in the following paragraphs.

1. One such improvement could involve evaluating the model's performance in classifying data from other aircraft types, such as rotating wing aircraft types and fighters and assessing the model's applicability in these contexts. It may be necessary to construct the model using data from these specific aircraft types to accurately classify their data.
2. Another potential improvement could involve incorporating data classified by multiple analysts. Since each analyst may have his or her own unique

criteria for classification, relying solely on one person's criteria may yield less optimal results.

3. During the manual classification process, it was observed that a third class could be introduced beyond failure and non-failure. Therefore, testing the models based on three types, namely preventive maintenance, corrective maintenance, and non-failure, could provide additional information for analysis.
4. Additionally, using other relevant columns or features, besides the discrepancy description, as input for model building and classification could enhance the model's performance, as it would have access to more comprehensive information.

These proposed improvements have the potential to refine the research and contribute to the overall accuracy and applicability of the machine-learning models in classifying failure data from the Brazilian Air Force's database.

APPENDIX

The algorithm was split into four sets of code that are provided in this appendix, following the requirements (libraries and their respective versions) used to run them in Python version 3.10.11.

A. LOAD_AND_TRANFORM_DATA.PY

```
#The first step is to import the libraries

import pandas as pd

import numpy as np

import seaborn as sns

import re, nltk

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,
TfidfTransformer

from sklearn.base import BaseEstimator, TransformerMixin

from lightgbm import LGBMClassifier

from sklearn.pipeline import Pipeline

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import MultinomialNB

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score,
auc, roc_curve, f1_score, precision_recall_curve, classification_report, log_loss

from nltk.corpus import stopwords

from tqdm import tqdm
```

```

from sklearn.model_selection import RandomizedSearchCV, cross_val_score,
StratifiedKfold, train_test_split

```

```

from sklearn.inspection import permutation_importance

```

```

import pickle

```

```

import seaborn as sns

```

```

import matplotlib.pyplot as plt

```

#The following function was created to read the data.

```

def read_data(ac_type, year, c_type):

```

```

    data = pd.read_csv('..\|\'

```

```

        + ac_type + '_' + year + '.csv',encoding = c_type)

```

```

    data ['Validacao'] = data ['Validacao'].apply(lambda x: 1 if (x=="Sim") else

```

0)

```

    return data

```

Following that, the data must be joined and split between the column of analysis and the validation

```

def joining_data(join_list):

```

```

    data = pd.concat(join_list)

```

```

    data = data.loc[:,['DISCREPANCIA', 'UNIDADE', 'PROJETO',
'MATRICULA', 'PN ITEM DEFEITUOSO', 'SN ITEM DEFEITUOSO', 'Validacao']]

```

```

    data = data.dropna(subset=['DISCREPANCIA'])

```

```

    data ['DISCREPANCIA'] = data ['DISCREPANCIA'].apply(str)

```

```

    data.reset_index(drop=True, inplace=True)

```

```

    return data

```

```

# the function was used to transform the text data and prepare it to be vectorized
def text_transformer(raw_text):

    def stemmer(text):

        stemmer_nltk = nltk.stem.RSLPStemmer()

        aux = ''

        for sp in text.split():

            aux = aux + ' ' + stemmer_nltk.stem(sp)

        return aux

    x = raw_text

    x = re.sub(r'\b [0-9]\d*[a-z]*', 'measure ', x) #Change the numbers followed by
text to measure

    x = re.sub(r'(?:[a-zA-Z]+[0-9]|[0-9]+[a-zA-Z])[a-zA-Z0-9]*', 'SNPNPUB', x)
#assume that is a PN or publication

    x = x.lower()

    x = re.sub(r'^[\w\s]+', ' ', x) #take off special characters such as %, $, - etc

    x = stemmer(x)#reduce the word to its radicals

    return [word for word in x.split() if word not in stopwords.words('portuguese')]
#here the stopwords are removed

```

```

# The function below splits the data into data for training and data for testing
def split_train_test(joined_data, percentage = 0.2):

    X_train, X_test, y_train, y_test = \

        train_test_split(joined_data ['DISCREPANCIA'], joined_data ['Validacao'],
test_size = percentage)

    return X_train, X_test, y_train, y_test

```

B. TECHNIQUES_BUILDING.PY

```
from load_and_transform_data import *

def lgbm_model(X_train, y_train):

    pipeline = Pipeline([

        ('bow', CountVectorizer(analyzer=text_transformer, min_df = 4)), # strings
to token integer counts

        ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores

        ('classifier', LGBMClassifier(n_jobs=-1, random_state=12)), # train on TF-
IDF vectors w/ Light GBM

    ])

    params_tuning = { 'classifier__n_estimators': [10, 50, 100, 200]}

    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=12)

    search = RandomizedSearchCV(pipeline, params_tuning,

                                cv=cv, n_iter=25, verbose=1, random_state=12, n_jobs=-1,

                                scoring='f1')\

                                .fit(X_train,y_train)

    model = search.best_estimator_

    params = search.best_params_

    return model, params, search

def lr_model(X_train, y_train):

    pipeline = Pipeline([
```

```
('bow', CountVectorizer(analyzer=text_transformer, min_df = 4)), # strings  
to token integer counts
```

```
('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
```

```
('classifier', LogisticRegression(n_jobs=-1, random_state=12)), # train on  
TF-IDF vectors w/ Logistic Regression
```

```
])
```

```
params_tuning = { 'classifier__C': [0.01, 0.1, 1, 10, 100],  
                  }
```

```
cv = StratifiedKfold(n_splits=5, shuffle=True, random_state=12)
```

```
search = RandomizedSearchCV(pipeline, params_tuning,
```

```
                             cv=cv, n_iter=25, verbose=1, random_state=12, n_jobs=-1,
```

```
                             scoring='f1')\
```

```
                             .fit(X_train,y_train)
```

```
model = search.best_estimator_
```

```
params = search.best_params_
```

```
return model, params, search
```

```
def mnnbc_model(X_train, y_train):
```

```
    pipeline = Pipeline([
```

```
        ('bow', CountVectorizer(analyzer=text_transformer, min_df = 4)), # strings  
to token integer counts
```

```
        ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
```

```
        ('classifier', MultinomialNB()), # train on TF-IDF vectors w/ Multinomial  
Naive Bayes
```

```
    ])
```

```

params_tuning = { 'classifier__alpha': [0.01, 0.1, 1, 10, 100]
                  }

cv = StratifiedKfold(n_splits=5, shuffle=True, random_state=12)

search = RandomizedSearchCV(pipeline, params_tuning,
                             cv=cv, n_iter=25, verbose=1, random_state=12, n_jobs=-1,
                             scoring='f1')\
                             .fit(X_train,y_train)

model = search.best_estimator_

params = search.best_params_

return model, params, search

def rfc_model(X_train, y_train):

    pipeline = Pipeline([

        ('bow', CountVectorizer(analyzer=text_transformer, min_df = 4)), # strings
to token integer counts

        ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores

        ('classifier', RandomForestClassifier(n_jobs=-1, random_state=12)), # train
on TF-IDF vectors w/ Random Forest Classifier

    ])

    params_tuning = { 'classifier__n_estimators': [10, 25, 50, 75, 100, 200],
                      'classifier__max_depth': [10, 20, 30],
                      }

    cv = StratifiedKfold(n_splits=5, shuffle=True, random_state=12)

    search = RandomizedSearchCV(pipeline, params_tuning,

```

```

        cv=cv, n_iter=25, verbose=1, random_state=12, n_jobs=-1,
        scoring='f1')\

        .fit(X_train,y_train)

model = search.best_estimator_

params = search.best_params_

return model, params, search

def svc_model(X_train, y_train):

    pipeline = Pipeline([

        ('bow', CountVectorizer(analyzer=text_transformer, min_df = 4)), # strings
to token integer counts

        ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores

        ('classifier', SVC(probability=True, random_state=12)), # train on TF-IDF
vectors w/ Support Vector Machine

    ])

    params_tuning = { 'classifier__C': [0.01, 0.1, 1, 10, 100],

        }

    cv = StratifiedKfold(n_splits=5, shuffle=True, random_state=12)

    search = RandomizedSearchCV(pipeline, params_tuning,

        cv=cv, n_iter=25, verbose=1, random_state=12, n_jobs=-1,

        scoring='f1')\

        .fit(X_train,y_train)

model = search.best_estimator_

params = search.best_params_

```

```

    return model, params, search

def knn_model(X_train, y_train):
    pipeline = Pipeline([
        ('bow', CountVectorizer(analyzer=text_transformer, min_df = 4)), # strings
        to token integer counts

        ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores

        ('classifier', KNeighborsClassifier()), # train on TF-IDF vectors w/ Support
        Vector Machine

    ])

    params_tuning = { 'classifier__n_neighbors': [1, 10, 25, 50, 100, 500],
                      }

    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=12)

    search = RandomizedSearchCV(pipeline, params_tuning,
                                cv=cv, n_iter=25, verbose=1, random_state=12, n_jobs=-1,
                                scoring='f1')\

                                .fit(X_train,y_train)

    model = search.best_estimator_

    params = search.best_params_

    return model, params, search

```

C. APP_TECHNIQUES.PY

```

from load_and_transform_data import *

from techniques_building import *

```

```

class DataLoadSplit:

    def __init__(self, aircraft_years):

        self.aircraft_years = aircraft_years

        self.data = self.join_data()

        self.x_train, self.x_test, self.y_train, self.y_test = self.split_data()

    def failure_data_read(self, ac_type, year):

        if ac_type == 'training' and year == '2018':

            c_type = 'utf-8'

        elif ac_type == 'training' and year == '2019':

            c_type = 'utf-8'

        else:

            c_type = 'ISO-8859-1'

        data = read_data(ac_type, year, c_type)

        return data

    def join_data(self):

        join_list = []

        for ac_year in self.aircraft_years:

            ac_type, year = ac_year [0], ac_year [1]

            data = self.failure_data_read(ac_type, year)

            join_list.append(data)

        data = joining_data(join_list)

        return data

    def split_data(self, percentage=0.2):

        x_train, x_test, y_train, y_test = \

```

```

        split_train_test(self.data, percentage)

    return x_train, x_test, y_train, y_test

class LGBMModel:

    def __init__(self, data_loader):

        self.pipeline = Pipeline([

            ('bow', CountVectorizer(analyzer=text_transformer, min_df=4)),

            ('tfidf', TfidfTransformer()),

            ('classifier', LGBMClassifier(n_jobs=-1, random_state=12)),

        ])

        self.model, self.params, self.search = self.fit_lgbm(data_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train,          data_loader.x_test,          data_loader.y_train,

data_loader.y_test

        self.predictions = self.predict()

        self.clasf_rep = self.cla_rep()

        self.conf_mx = self.c_matr()

        self.evaluation = self.evaluate()

        self.best_param_nestimator = self.params ['classifier__n_estimators']

    def fit_lgbm(self, data_loader):

        model,      params,      search      =      lgbm_model(data_loader.x_train,

data_loader.y_train)

        return model, params, search

    def predict(self, x_data=None):

```

```

if x_data == None:

    x_data = self.x_test

    data_predict = self.model.predict(x_data)

    return data_predict


def cla_rep(self):

    clasf_report = classification_report(y_true=self.y_test,
y_pred=self.predictions, digits=5)

    return clasf_report


def c_matr(self):

    conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)

    return conf_matrix


def evaluate(self):

    clasf_report = self.clasf_rep

    conf_matrix = self.conf_mx

    return '\t\tClassification Report' + '\n' + \

        '-----' + '\n' + \

        clasf_report + '\n' + \

        '-----' + '\n\n' + \

        '\t\tConfusion Matrix' + '\n' + \

        '-----' + '\n' + \

        str(conf_matrix) + '\n' + \

```

‘-----’

```
class LrModel:

    def __init__(self, data_loader):

        self.pipeline = Pipeline([

            (‘bow’, CountVectorizer(analyzer=text_transformer, min_df=4)),

            (‘tfidf’, TfidfTransformer()),

            (‘classifier’, LogisticRegression(n_jobs=-1, random_state=12)),

        ])

        self.model, self.params, self.search = self.fit_lr(data_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train, data_loader.x_test, data_loader.y_train,
data_loader.y_test

        self.predictions = self.predict()

        self.clasf_rep = self.cla_rep()

        self.conf_mx = self.c_matr()

        self.evaluation = self.evaluate()

        self.best_param_C = self.params [‘classifier__C’]

    def fit_lr(self, data_loader):

        model, params, search = lr_model(data_loader.x_train, data_loader.y_train)

        return model, params, search

    def predict(self, x_data=None):

        if x_data == None:
```

```

        x_data = self.x_test

        data_predict = self.model.predict(x_data)

        return data_predict

    def cla_rep(self):

        clasf_report = classification_report(y_true=self.y_test,
        y_pred=self.predictions, digits=5)

        return clasf_report

    def c_matr(self):

        conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)

        return conf_matrix

    def evaluate(self):

        clasf_report = self.clasf_rep

        conf_matrix = self.conf_mx

        return '\t\tClassification Report' + '\n' + \

            '-----' + '\n' + \

            clasf_report + '\n' + \

            '-----' + '\n\n' + \

            '\t\tConfusion Matrix' + '\n' + \

            '-----' + '\n' + \

            str(conf_matrix) + '\n' + \

            '-----'

```

```

class MNNBCModel:

    def __init__(self, data_loader):

        self.pipeline = Pipeline([

            ('bow', CountVectorizer(analyzer=text_transformer, min_df=4)),

            ('tfidf', TfidfTransformer()),

            ('classifier', MultinomialNB()),

        ])

        self.model, self.params, self.search = self.fit_mnnbc(data_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train,          data_loader.x_test,          data_loader.y_train,
data_loader.y_test

        self.predictions = self.predict()

        self.clasf_rep = self.cla_rep()

        self.conf_mx = self.c_matr()

        self.evaluation = self.evaluate()

        self.best_param_alpha = self.params ['classifier__alpha']


    def fit_mnnbc(self, data_loader):

        model,      params,      search      =      mnnbc_model(data_loader.x_train,
data_loader.y_train)

        return model, params, search


    def predict(self, x_data=None):

```

```

if x_data == None:

    x_data = self.x_test

    data_predict = self.model.predict(x_data)

    return data_predict


def cla_rep(self):

    clasf_report = classification_report(y_true=self.y_test,
y_pred=self.predictions, digits=5)

    return clasf_report


def c_matr(self):

    conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)

    return conf_matrix


def evaluate(self):

    clasf_report = self.clasf_rep

    conf_matrix = self.conf_mx

    return '\t\tClassification Report' + '\n' + \

        '-----' + '\n' + \

        clasf_report + '\n' + \

        '-----' + '\n\n' + \

        '\t\tConfusion Matrix' + '\n' + \

        '-----' + '\n' + \

        str(conf_matrix) + '\n' + \

```

‘-----’

```
class RFCModel:

    def __init__(self, data_loader):

        self.pipeline = Pipeline([

            ('bow', CountVectorizer(analyzer=text_transformer, min_df=4)),

            ('tfidf', TfidfTransformer()),

            ('classifier', RandomForestClassifier(n_jobs=-1, random_state=12)),

        ])

        self.model, self.params, self.search = self.fit_rfc(data_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train,          data_loader.x_test,          data_loader.y_train,

            data_loader.y_test

        self.predictions = self.predict()

        self.clasf_rep = self.cla_rep()

        self.conf_mx = self.c_matr()

        self.evaluation = self.evaluate()

        self.best_param_nestimator = self.params ['classifier__n_estimators']

        self.best_param_max_depth = self.params ['classifier__max_depth']

    def fit_rfc(self, data_loader):

        model, params, search = rfc_model(data_loader.x_train, data_loader.y_train)

        return model, params, search
```

```

def predict(self, x_data=None):
    if x_data == None:
        x_data = self.x_test
    data_predict = self.model.predict(x_data)
    return data_predict

def cla_rep(self):
    clasf_report = classification_report(y_true=self.y_test,
y_pred=self.predictions, digits=5)
    return clasf_report

def c_matr(self):
    conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)
    return conf_matrix

def evaluate(self):
    clasf_report = self.clasf_rep
    conf_matrix = self.conf_mx
    return '\t\tClassification Report' + '\n' + \
        '-----' + '\n' + \
        clasf_report + '\n' + \
        '-----' + '\n\n' + \
        '\t\tConfusion Matrix' + '\n' + \
        '-----' + '\n' + \

```

```

str(conf_matrix) + '\n' + \
'-----',

```

```

class SVCModel:

    def __init__(self, data_loader):

        self.pipeline = Pipeline([

            ('bow', CountVectorizer(analyzer=text_transformer, min_df=4)),

            ('tfidf', TfidfTransformer()),

            ('classifier', SVC(probability=True, random_state=12)),

        ])

        self.model, self.params, self.search = self.fit_svc(data_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train,          data_loader.x_test,          data_loader.y_train,
data_loader.y_test

        self.predictions = self.predict()

        self.class_rep = self.class_rep()

        self.conf_mx = self.conf_matr()

        self.evaluation = self.evaluate()

        self.best_param_C = self.params ['classifier__C']


    def fit_svc(self, data_loader):

        model,          params,          search          =          svc_model(data_loader.x_train,
data_loader.y_train)

        return model, params, search

```

```

def predict(self, x_data=None):

    if x_data == None:

        x_data = self.x_test

    data_predict = self.model.predict(x_data)

    return data_predict


def cla_rep(self):

    clasf_report = classification_report(y_true=self.y_test,
y_pred=self.predictions, digits=5)

    return clasf_report


def c_matr(self):

    conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)

    return conf_matrix


def evaluate(self):

    clasf_report = self.clasf_rep

    conf_matrix = self.conf_mx

    return '\t\tClassification Report' + '\n' + \

        '-----' + '\n' + \

        clasf_report + '\n' + \

        '-----' + '\n\n' + \

        '\t\tConfusion Matrix' + '\n' + \

```

```

'-----' + '\n' + \
str(conf_matrix) + '\n' + \
'-----'

```

```

class KNNModel:

    def __init__(self, data_loader):

        self.pipeline = Pipeline([

            ('bow', CountVectorizer(analyzer=text_transformer, min_df=4)),

            ('tfidf', TfidfTransformer()),

            ('classifier', KNeighborsClassifier()),

        ])

        self.model, self.params, self.search = self.fit_knn(data_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train,          data_loader.x_test,          data_loader.y_train,
data_loader.y_test

        self.predictions = self.predict()

        self.clasf_rep = self.cla_rep()

        self.conf_mx = self.c_matr()

        self.evaluation = self.evaluate()

        self.best_param = self.params ['classifier__n_neighbors']

    def fit_knn(self, data_loader):

        model,          params,          search          =          knn_model(data_loader.x_train,
data_loader.y_train)

```

```
return model, params, search
```

```
def predict(self, x_data=None):
```

```
    if x_data == None:
```

```
        x_data = self.x_test
```

```
    data_predict = self.model.predict(x_data)
```

```
    return data_predict
```

```
def cla_rep(self):
```

```
    clasf_report = classification_report(y_true=self.y_test,  
y_pred=self.predictions, digits=5)
```

```
    return clasf_report
```

```
def c_matr(self):
```

```
    conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)
```

```
    return conf_matrix
```

```
def evaluate(self):
```

```
    clasf_report = self.clasf_rep
```

```
    conf_matrix = self.conf_mx
```

```
    return '\t\tClassification Report' + '\n' + \
```

```
        '-----' + '\n' + \
```

```
        clasf_report + '\n' + \
```

```
        '-----' + '\n\n' + \
```

```

'\t\tConfusion Matrix' + '\n' + \

'-----' + '\n' + \

str(conf_matrix) + '\n' + \

'-----',

```

```

class VotingModel1:

    def __init__(self, data_loader, svc_loader, lr_loader, mnb_loader):

        self.model = self.fit_voting1(data_loader, svc_loader, lr_loader, mnb_loader)

        self.x_train, self.x_test, self.y_train, self.y_test = \

            data_loader.x_train,          data_loader.x_test,          data_loader.y_train,
data_loader.y_test

        self.predictions = self.predict()

        self.clasf_rep = self.cla_rep()

        self.conf_mx = self.c_matr()

        self.evaluation = self.evaluate()

    def fit_voting1(self, data_loader, svc_loader, lr_loader, mnb_loader):

        pipeline = Pipeline([

            ('bow', CountVectorizer(analyzer=text_transformer, min_df =
4)), # strings to token integer counts

            ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF
scores

            ('classifier', VotingClassifier(estimators=

                [ ('svc', SVC(C=svc_loader.best_param_C,
probability=True, random_state=12)),

```

```

        ('lr',
LogisticRegression(C=lr_loader.best_param_C, n_jobs=-1, random_state=12)),

        ('mn',
MultinomialNB(alpha=mn_loader.best_param_alpha))

    ],

    n_jobs=-1,

    voting='soft'

)), # train on TF-IDF vectors w/ Voting

])

model = pipeline.fit(data_loader.x_train,data_loader.y_train)

return model

def predict(self, x_data=None):

    if x_data == None:

        x_data = self.x_test

    data_predict = self.model.predict(x_data)

    return data_predict

def cla_rep(self):

    clasf_report = classification_report(y_true=self.y_test,
y_pred=self.predictions, digits=5)

    return clasf_report

def c_matr(self):

```

```

conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)

return conf_matrix

```

```

def evaluate(self):

```

```

    clasf_report = self.clasf_rep

```

```

    conf_matrix = self.conf_mx

```

```

    return '\t\tClassification Report' + '\n' + \

```

```

        '-----' + '\n' + \

```

```

        clasf_report + '\n' + \

```

```

        '-----' + '\n\n' + \

```

```

        '\t\tConfusion Matrix' + '\n' + \

```

```

        '-----' + '\n' + \

```

```

        str(conf_matrix) + '\n' + \

```

```

        '-----'

```

```

class VotingModel2:

```

```

    def __init__(self, data_loader, svc_loader, lr_loader, lgbm_loader):

```

```

        self.model = self.fit_voting2(data_loader, svc_loader, lr_loader, lgbm_loader)

```

```

        self.x_train, self.x_test, self.y_train, self.y_test = \

```

```

            data_loader.x_train,      data_loader.x_test,      data_loader.y_train,
data_loader.y_test

```

```

        self.predictions = self.predict()

```

```

        self.clasf_rep = self.cla_rep()

```

```

        self.conf_mx = self.c_matr()

```

```

self.evaluation = self.evaluate()

def fit_voting2(self, data_loader, svc_loader, lr_loader, lgbm_loader):

    pipeline = Pipeline([

        ('bow', CountVectorizer(analyzer=text_transformer, min_df =
4)), # strings to token integer counts

        ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF
scores

        ('classifier', VotingClassifier(estimators=

[(('svc', SVC(C=svc_loader.best_param_C,
probability=True, random_state=12)),

('lr',
LogisticRegression(C=lr_loader.best_param_C, n_jobs=-1, random_state=12)),

('lgbm',
LGBMClassifier(n_estimators=lgbm_loader.best_param_nestimator, n_jobs=-1,
random_state=12))

],

n_jobs=-1,

voting='soft'

)), # train on TF-IDF vectors w/ Voting

])

model = pipeline.fit(data_loader.x_train, data_loader.y_train)

return model

def predict(self, x_data=None):

```

```

if x_data == None:

    x_data = self.x_test

    data_predict = self.model.predict(x_data)

    return data_predict


def cla_rep(self):

    clasf_report = classification_report(y_true=self.y_test,
y_pred=self.predictions, digits=5)

    return clasf_report


def c_matr(self):

    conf_matrix = confusion_matrix(y_true=self.y_test, y_pred=self.predictions)

    return conf_matrix


def evaluate(self):

    clasf_report = self.clasf_rep

    conf_matrix = self.conf_mx

    return '\t\tClassification Report' + '\n' + \

        '-----' + '\n' + \

        clasf_report + '\n' + \

        '-----' + '\n\n' + \

        '\t\tConfusion Matrix' + '\n' + \

        '-----' + '\n' + \

        str(conf_matrix) + '\n' + \

```

‘-----’

D. MODEL_BUILDER.PY

```
from load_and_transform_data import *

from techniques_building import *

from app_techniques import *


#Choose the data

ac_types = ['attack', 'training', 'transport']

years = ['2018', '2019', '2020']

#sub_year = ['2020', '2021']

data_analyzed = []

for x in ac_types:

    for y in years:

        data_analyzed += [(x,y)]

#Loading the data

all_data = DataLoadSplit(data_analyzed)

with open('all_data.pkl', 'wb') as f:

    pickle.dump(all_data, f)


#Building the LGBM model

all_data_lgbm = LGBMModel(all_data)

with open('all_data_lgbm.pkl', 'wb') as f:

    pickle.dump(all_data_lgbm, f)
```

#Building the LR model

all_data_lr = LrModel(all_data)

with open('all_data_lr.pkl', 'wb') as f:

pickle.dump(all_data_lr, f)

#Building the MNNBC model

all_data_mnnbc = MNNBCModel(all_data)

with open('all_data_mnnbc.pkl', 'wb') as f:

pickle.dump(all_data_mnnbc, f)

#Building the RFC model

all_data_rfc = RFCModel(all_data)

with open('all_data_rfc.pkl', 'wb') as f:

pickle.dump(all_data_rfc, f)

#Building the SVC model

all_data_svc = SVCModel(all_data)

with open('all_data_svc.pkl', 'wb') as f:

pickle.dump(all_data_svc, f)

#Building the KNN model

all_data_knn = KNNModel(all_data)

with open('all_data_knn.pkl', 'wb') as f:

```
pickle.dump(all_data_knn, f)
```

```
#Building the Voting model
```

```
all_data_voting1 = VotingModel1(all_data, all_data_svc, all_data_lr,  
all_data_mnnbc)
```

```
with open('all_data_voting1.pkl', 'wb') as f:
```

```
pickle.dump(all_data_voting1, f)
```

```
#Building the Voting2 model
```

```
all_data_voting2 = VotingModel2(all_data, all_data_svc, all_data_lr,  
all_data_lgbm)
```

```
with open('all_data_voting2.pkl', 'wb') as f:
```

```
pickle.dump(all_data_voting2, f)
```

E. REQUIREMENTS

```
asttokens==2.2.1
```

```
backcall==0.2.0
```

```
click==8.1.3
```

```
colorama==0.4.6
```

```
comm==0.1.2
```

```
contourpy==1.0.7
```

```
cycler==0.11.0
```

```
debugpy==1.6.6
```

```
decorator==5.1.1
```

```
et-xmlfile==1.1.0
```

executing==1.2.0
fonttools==4.38.0
ipykernel==6.21.2
ipython==8.11.0
jedi==0.18.2
joblib==1.2.0
jupyter_client==8.0.3
jupyter_core==5.2.0
kiwisolver==1.4.4
lightgbm==3.3.5
matplotlib==3.7.0
matplotlib-inline==0.1.6
nest-asyncio==1.5.6
nltk==3.8.1
numpy==1.24.2
openpyxl==3.1.2
packaging==23.0
pandas==1.5.3
parso==0.8.3
pickleshare==0.7.5
Pillow==9.4.0
platformdirs==3.0.0
prompt-toolkit==3.0.38
psutil==5.9.4

pure-eval==0.2.2
Pygments==2.14.0
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.7.1
pywin32==305
pyzmq==25.0.0
regex==2022.10.31
scikit-learn==1.2.1
scipy==1.10.1
seaborn==0.12.2
six==1.16.0
sklearn==0.0.post1
stack-data==0.6.2
threadpoolctl==3.1.0
tornado==6.2
tqdm==4.64.1
traitlets==5.9.0
wcwidth==0.2.6

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Abdul-Kader, S. A., & Woods, J. (2015). Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7). <https://doi.org/10.14569/IJACSA.2015.060712>
- Abedini, M., Ghasemian, B., Shirzadi, A., & Bui, D. T. (2019). A comparative study of Support Vector Machine and logistic model tree classifiers for shallow landslide susceptibility modeling. *Environmental Earth Sciences*, 78(18), 560. <https://doi.org/10.1007/s12665-019-8562-z>
- Ahuja, R., Chug, A., Kohli, S., Gupta, S., & Ahuja, P. (2019). The impact of features extraction on the sentiment analysis. *Procedia Computer Science*, 152, 341–348. <https://doi.org/10.1016/j.procs.2019.05.008>
- Akbari, M., & Do, T. N. A. (2021). A systematic review of machine learning in logistics and supply chain management: Current trends and future directions. *Benchmarking: An International Journal*, 28(10), 2977–3005. <https://doi.org/10.1108/BIJ-10-2020-0514>
- Alsanad, A. (2022). An improved Arabic sentiment analysis approach using optimized Multinomial Naïve Bayes Classifier. *International Journal of Advanced Computer Science and Applications*, 13(8). <https://doi.org/10.14569/IJACSA.2022.0130812>
- Al-shalabi, H., Tiun, S., Omar, N., Alezabi, K. A., & Al-Aswadi, F. N. (2022). The effectiveness of Arabic stemmers using Arabized word removal. *International Journal of Information Science and Management*, 20(4), 87–102.
- Bekkar, M., Djemaa, H. K., & Alitouche, T. A. (2013). Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications*, 3(10).
- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python* (1st ed). O'Reilly.
- Brazil, General Controllershship of the Union. (2023). *Detalhamento dos servidores públicos por órgão—Portal da Transparência [Details of civil servants by body—Transparency Portal]*. <https://portal.datransparencia.gov.br/servidores/orgao?ordenarPor=orgaoSuperiorLotacaoSIAPE&direcao=asc>

- Brazilian Air Force. (2006a). *Manutenção Centrada na Confiabilidade [Reliability Centered Maintenance]* (MCA 400–15). <https://www.sislaer.fab.mil.br/terminalcendoc/acervo/detalhe/3560?guid=1673811637892&returnUrl=%2fterminalcendoc%2fresultado%2flistarlegislacao%3fguid%3d1673811637892%26quantidadePaginas%3d1%26codigoRegistro%3d3560%233560&i=1>
- Brazilian Air Force. (2006b). *Sistema de confiabilidade do SISMA e do SISMAB [SISMA and SISMAB reliability system]* (ICA 400–21). <https://www.sislaer.fab.mil.br/terminalcendoc/Busca/Download?codigoArquivo=2275>
- Brazilian Air Force. (2017). *Manual de Manutenção [Maintenance Manual]* (MCA 66–7). <https://www.sislaer.fab.mil.br/terminalcendoc/Busca/Download?codigoArquivo=1766>
- Brazilian Air Force. (2021). *Plano de Ciência, Tecnologia e Inovação da Aeronáutica [Air Force Science, Technology and Innovation Plan]* (PCA 11–217). <https://www.sislaer.fab.mil.br/terminalcendoc/Busca/Download?codigoArquivo=30682>
- Brazilian Minister of Defense. (2012). *Livro Branco de Defesa Nacional [Defense White Paper]*. https://www.gov.br/defesa/pt-br/arquivos/estado_e_defesa/livro_branco/Versao2012dolivroLBDNemingles.pdf
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5–32.
- Cavalcante, A. P. M., & Ferreira, B. F. (2015). *Possibilidade de aumento de TBO dos itens CHDERU e CLDERU [Possibility of increasing the TBO of items CHDERU and CLDERU]* (RT SP15 TENG 010 A-1). Força Aérea Brasileira – PAMASP.
- Centric Consulting. (2019, April 10). Machine learning: A quick introduction and five core steps. *Centric Consulting*. <https://centricconsulting.com/blog/machine-learning-a-quick-introduction-and-five-core-steps/>
- Chauhan, S. P., & Bahal, P. (2020). Introduction to machine learning. *Emerging Trends in Big Data, IoT and Cyber Security*, 115–118.
- Chaves, P. L. (2020, November 26). *Mineração de dados de FCDD por meio de linguagem Python [FCDD data mining through Python language]* [Presentation]. 2º Encontro de Confiabilidade na Aviação, Guarulhos, SP, Brazil. <https://www2.fab.mil.br/ila/index.php/downloads/category/40-2-encontro-de-confiabilidade?download=174:apresentacao>
- Custard, J. L., Lease, Q. E., & Schotman, J. D. (2016). *Study of using excess stock to reduce naval aviation depot-level repairable piece part backorders* [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. <https://apps.dtic.mil/sti/pdfs/AD1030824.pdf>

- Deng, H., Zhou, Y., Wang, L., & Zhang, C. (2021). Ensemble learning for the early prediction of neonatal jaundice with genetic features. *BMC Medical Informatics and Decision Making*, 21(1), 338. <https://doi.org/10.1186/s12911-021-01701-9>
- Dey, L., Chakraborty, S., Biswas, A., Bose, B., & Tiwari, S. (2016). Sentiment analysis of review datasets using Naïve Bayes' and K-NN classifier. *International Journal of Information Engineering and Electronic Business*, 8(4), 54–62. <https://doi.org/10.5815/ijieeb.2016.04.07>
- Do, T. (2022, March 9). *5 best machine learning classification algorithms + real-world projects*. Omdena. <https://omdena.com/blog/machine-learning-classification-algorithms/>
- Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3), 103–130.
- Doshi, J. (2021). Chatbot user interface for Customer Relationship Management using NLP models. *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, 1–4. <https://doi.org/10.1109/AIMV53313.2021.9670914>
- Fernandez-Grandon, C., Soto, I., Zabala-Blanco, D., Alavia, W., & Garcia, V. (2021). SVM and ANN classification using GLCM and HOG features for COVID-19 and pneumonia detection from chest x-rays. *2021 Third South American Colloquium on Visible Light Communications (SACVLC)*, 01–06. <https://doi.org/10.1109/SACVLC53127.2021.9652248>
- Filho, C. M. B., & Martins, L. S. (2016). *Análise dos dados de defeito da frota T-27 [T-27 fleet defect data analysis]* (RT LS 16 010 TENG T-27). Força Aérea Brasileira – PAMALS.
- Flight International. (2023). *2023 world air forces*. <https://www.flightglobal.com/download?ac=90688>
- Forca Aerea Brasileira. (n.d.). *Mapa_FAB.png (6910×5422)*. Retrieved January 14, 2023, from https://www.fab.mil.br/Download/arquivos/mapa_FAB.png
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5). <https://doi.org/10.1214/aos/1013203451>
- Hua, Z., & Zhang, B. (2006). A hybrid Support Vector Machines and logistic regression approach for forecasting intermittent demand of spare parts. *Applied Mathematics and Computation*, 181(2), 1035–1048. <https://doi.org/10.1016/j.amc.2006.01.064>
- IBM. (2023). *What is boosting?* <https://www.ibm.com/topics/boosting>

- Iwata, C., & Mavris, D. (2013). Object-oriented discrete event simulation modeling environment for aerospace vehicle maintenance and logistics process. *Procedia Computer Science*, 16, 187–196. <https://doi.org/10.1016/j.procs.2013.01.020>
- Jacobs, B. K. (2000). *Warranty/cannibalization issues, disruptive forces in the production and maintainability of the E-2C aircraft* [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. <https://apps.dtic.mil/sti/pdfs/ADA379444.pdf>
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning with applications in R. *Statistical Theory and Related Fields*, 6(1), 87–87. <https://doi.org/10.1080/24754269.2021.1980261>
- Janes. (2022, July 21). *Brazil*. <https://customer-janes-com>
- Jones, J. V. (2006). *Integrated logistics support handbook*. McGraw Hill.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: a highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 3146–3154.
- Keita, Z. (2022, September). Classification in machine learning: A guide for beginners. *Data Camp*. <https://www.datacamp.com/blog/classification-machine-learning>
- Keizers, J. M., Bertrand, J. W. M., & Wessels, J. (2003). Diagnosing order planning performance at a Navy maintenance and repair organization, using logistic regression. *Production and Operations Management*, 12(4), 445–463.
- Kelly, S. M. (2023, March 21). *Google begins rolling out its ChatGPT rival*. CNN. <https://www.cnn.com/2023/03/21/tech/google-bard/index.html>
- Khader, M., Awajan, A., & Al-Naymat, G. (2018). The effects of Natural Language Processing on big data analysis: Sentiment analysis case study. *2018 International Arab Conference on Information Technology (ACIT)*, 1–7. <https://doi.org/10.1109/ACIT.2018.8672697>
- Koirala, P., & Shakya, A. (2020). *A Nepali rule based stemmer and its performance on different NLP applications*. <https://doi.org/10.48550/arxiv.2002.09901>
- Langley, P., Iba, W., & Thompson, K. (1992). An analysis of bayesian classifiers. *Aaai*, 90, 223–228.
- Li, R., Chen, X., Balezentis, T., Streimikiene, D., & Niu, Z. (2021). Multi-step least squares Support Vector Machine modeling approach for forecasting short-term electricity demand with application. *Neural Computing and Applications*, 33(1), 301–320. <https://doi.org/10.1007/s00521-020-04996-3>

- Li, Y., & Li, Z. (2019). Forecasting of coal demand in China based on Support Vector Machine optimized by the improved gravitational search algorithm. *Energies*, 12(12), 2249. <https://doi.org/10.3390/en12122249>
- Lipton, Z. C., Elkan, C., & Naryanaswamy, B. (2014). Optimal thresholding of classifiers to maximize F1 measure. In T. Calders, F. Esposito, E. Hüllermeier, & R. Meo (Eds.), *Machine learning and knowledge discovery in databases* (vol. 8725, pp. 225–239). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-44851-9_15
- Liu, Y., & Yang, S. (2022). Application of decision tree-based classification algorithm on content marketing. *Journal of Mathematics*, 2022, 1–10. <https://doi.org/10.1155/2022/6469054>
- Ma, T. M., Yamamori, K., & Thida, A. (2020). A comparative approach to Naïve Bayes Classifier and Support Vector Machine for email spam classification. *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, 324–326. <https://doi.org/10.1109/GCCE50665.2020.9291921>
- Manning, C. D., Raghavan, P., & Schütze, H. (2009). *An introduction to information retrieval*. Cambridge University Press.
- Mehrjoo, S., & Bashiri, M. (2013). An application of principal component analysis and logistic regression to facilitate production scheduling decision support system: An automotive industry case. *Journal of Industrial Engineering International*, 9(1), 14. <https://doi.org/10.1186/2251-712X-9-14>
- Mohit, I., Santhosh Kumar, K., Kumar Reddy, U. A., & Suresh Kumar, B. (2021). An approach to detect multiple diseases using machine learning algorithm. *Journal of Physics: Conference Series*, 2089(1), 012009. <https://doi.org/10.1088/1742-6596/2089/1/012009>
- Moubray, J. (1997). *Reliability-centered maintenance*. Industrial Press Inc.
- Padovese, B. T., & Padovese, L. R. (2019). *A machine learning approach to the recognition of Brazilian Atlantic forest parrot species* [Preprint]. Ecology. <https://doi.org/10.1101/2019.12.24.888180>
- Patil, G., Galande, V., Kekan, M. V., & Dange, K. (2014). Sentiment analysis using Support Vector Machine. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(1).
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (2011). Scikit-learn: Machine learning in Python. *Machine Learning in Python*. <https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf>

- Peterson, J. K. (1998). *Logistic regression applications and cluster analysis* [Master's thesis, Graduate Faculty of Texas Tech University]. <http://hdl.handle.net/2346/19443>
- Pimpalkar, A. P., & Retna Raj, R. J. (2020). Influence of pre-processing strategies on the performance of ML classifiers exploiting TF-IDF and BOW features. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 9(2), 49–68. <https://doi.org/10.14201/ADCAIJ2020924968>
- Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. *Big Data*, 1(1), 51–59. <https://doi.org/10.1089/big.2013.1508>
- Python. (2023, January 10). *Welcome to Python.org*. Python. <https://www.python.org/>
- Rafail, P., & Freitas, I. (2020). Natural Language Processing. In *SAGE Research Methods Foundations*. SAGE Publications Ltd. <https://doi.org/10.4135/9781526421036879118>
- Rahman, S., Talukder, K. H., & Mithila, S. K. (2021). An empirical study to detect cyberbullying with TF-IDF and machine learning algorithms. *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, 1–4. <https://doi.org/10.1109/ICECIT54077.2021.9641251>
- Rahmaningrum, S. A., & Oktaviana, P. P. (2020). Sentiment classification of hotel service review on Traveloka sites using Naïve Bayes Classifier (NBC) and binary logistic regression. *Journal of Physics: Conference Series*, 1490(1), 012065. <https://doi.org/10.1088/1742-6596/1490/1/012065>
- Ray, S. (2017, September 8). Commonly used machine learning algorithms (with Python and R codes). *Analytics Vidhya*. <https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>
- Reid, M. (2022). *Reliability – a Python library for reliability engineering (version 0.8.2) [computer software]* (0.8.2) [Python]. Zenodo. <https://doi.org/10.5281/ZENODO.3938000>
- Robinson, D. (2017, September 6). The incredible growth of Python. *Stack Overflow Blog*. <https://stackoverflow.blog/2017/09/06/incredible-growth-python/>
- Rundong, Y. (2020). *Enhancing the performance of automated guided vehicles through reliability, operation and maintenance assessment* [Doctoral dissertation, Loughborough University]. https://repository.lboro.ac.uk/articles/Enhancing_the_performance_of_automated_guided_vehicles_through_reliability_operation_and_maintenance_assessment/11935443/1

- Saini, A. (2021, September 20). Gradient boosting algorithm: A complete guide for beginners. *Analytics Vidhya*. <https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/>
- Scikit-Learn. (2023). *Ensemble methods*. Scikit-Learn. <https://scikit-learn/stable/modules/ensemble.html>
- Shamohammadi, O., Pahlavani, P., & Sharifi, M. A. (2023). Comparison of the performance of gradient boosting, logistic regression, and linear Support Vector Classifier algorithms in classifying travel modes based on GNSS data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVIII-4/W2-2022*, 103–108. <https://doi.org/10.5194/isprs-archives-XLVIII-4-W2-2022-103-2023>
- Shung, K. P. (2018, March 15). *Accuracy, precision, recall or F1? Towards Data Science*. <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>
- Silva, J. F. (2012). *Avaliação da efetividade da inspeção de 50 horas de voo do programa de manutenção da aeronave T-25 [Effectiveness assessment of the 50-hour flight inspection of the T-25 aircraft maintenance program]* (RT LS 12 TENG 009 T-25). Força Aérea Brasileira – PAMALS.
- Silva, L. A., Ballardin, R. A., & Silva, J. F. (2021). *Machine learning model for T-27 aircraft failure data validation* [Certificate thesis, Instituto de Logística da Aeronautica]. ILA Archive. <http://biblioteca.ila.intraer/ila/>
- Song, Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2), 130–135. <http://dx.doi.org/10.11919/j.issn.1002-0829.215044>
- Sousa, J. R. (2020). *Análise da confiabilidade do item trem de pouso auxiliar que equipa as aeronaves C-95 da Força Aérea Brasileira [Reliability analysis of the auxiliary landing gear item equipping the C-95 aircraft of the Brazilian Air Force]* (RT LS 20 TENG 001 C-95). Força Aérea Brasileira – PAMALS.
- Sousa, J. R. (2021). *Análise de confiabilidade do distribuidor de direção das aeronaves C/P-95 da Força Aérea Brasileira [Reliability analysis of the steering distributor of C/P-95 aircraft of the Brazilian Air Force]* (RT LS 017 TENG 21 C-95). Força Aérea Brasileira – PAMALS.
- Sousa, J. R., Ballardin, R. A., & Silva, J. F. (2022). *Evaluation of the change in the maintenance program for A-29 aircraft of the Brazilian Air Force* [Certificate thesis, Instituto de Logística da Aeronautica]. ILA Archive. <http://biblioteca.ila.intraer/ila/>

- Steinbach, M., & Tan, P.-N. (2009). Chapter 8 kNN: k-Nearest Neighbors. In *The top ten algorithms in data mining* (1st ed.). Chapman and Hall/CRC. <https://doi-org.libproxy.nps.edu/10.1201/9781420089653>
- Talpová, S. (2014). *Logistic regression: An option for a management research?* European Conference on Research Methodology for Business and Management Studies.
- Tirkolaee, E. B., Sadeghi, S., Mooseloo, F. M., Vandchali, H. R., & Aeini, S. (2021). Application of machine learning in supply chain management: A comprehensive overview of the main areas. *Mathematical Problems in Engineering*, 2021, 1–14. <https://doi.org/10.1155/2021/1476043>
- Vialetto, G., & Noro, M. (2019). Enhancement of a short-term forecasting method based on clustering and KNN: Application to an industrial facility powered by a cogenerator. *Energies*, 12(23), 4407. <https://doi.org/10.3390/en12234407>
- Vieira, T. C. J. (2015a). *Análise de confiabilidade dos motores AE3007 da família das aeronaves C-99 [Reliability analysis of the AE3007 engines of the C-99 aircraft family]* (RT GL 06/TENG/2015). Força Aérea Brasileira – PAMAGL.
- Vieira, T. C. J. (2015b). *Análise de confiabilidade dos motores CF34-10E da família das aeronaves VC-2 [Reliability analysis of the CF34-10E engines of the VC-2 aircraft family]* (RT GL 08/TENG/2015). Força Aérea Brasileira – PAMAGL.
- Vieira, T. C. J. (2015c). *Análise de confiabilidade para extensão de reguladores de oxigênio da aeronave A-1 PN: 441791-1 e PN: 441791 [Reliability analysis for aircraft oxygen regulator extension A-1 PN: 441791-1 and PN: 441791]* (RT GL 07/TENG/2015). Força Aérea Brasileira – PAMAGL.
- Vieira, T. C. J. (2016a). *Análise de confiabilidade das bombas hidráulicas MPEV3-0118UK2C [Reliability analysis of MPEV3-0118UK2C hydraulic pumps]* (RT GL 13/TENG/2016). Força Aérea Brasileira – PAMAGL.
- Vieira, T. C. J. (2016b). *Análise de confiabilidade dos itens da frota de aeronaves de C-105 [Reliability analysis of items in the C-105 aircraft fleet]* (RT GL 02/TENG/2016). Força Aérea Brasileira – PAMAGL.
- Vieira, T. C. J. (2019). *Análise de confiabilidade dos motores VK-2500 (projeto AH-2) [Reliability analysis of VK-2500 engines (project AH-2)]* (RT 02/DCGP-MOT/2019). Força Aérea Brasileira – DIRMAB.
- Vijayarani, D. S., Ilamathi, J., & Nithya. (2015). Preprocessing techniques for text mining—An overview. *International Journal of Computer Science & Communication Networks*, 5(1), 7–16.

- Wang, J. (2023). ChatGPT: a test drive. *American Journal of Physics*, 91(4), 255–256. <https://doi.org/10.1119/5.0145897>
- Weiss, S. M., Indurkha, N., & Zhang, T. (2015). *Fundamentals of predictive text mining*. Springer London. <https://doi.org/10.1007/978-1-4471-6750-1>
- Willianto, T., Supryadi, & Wibowo, A. (2020). Sentiment analysis on e-commerce product using machine learning and combination of TF-IDF and backward elimination. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(6), 2862–2867. <https://doi.org/10.35940/ijrte.F7889.038620>
- Wolff, R. (2020, August 26). 5 types of classification algorithms in machine learning. *MonkeyLearn Blog*. <https://monkeylearn.com/blog/classification-algorithms/>
- Xu, B., & Kumar, S. A. (2015). *A text mining classification framework and its experiments using aviation datasets*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2880836
- Xu, Z., & Saleh, J. H. (2021). Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliability Engineering & System Safety*, 211, 107530. <https://doi.org/10.1016/j.res.2021.107530>
- Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 05(04), 597–604. <https://doi.org/10.1142/S0219622006002258>
- Zestminds. (2022, November 25). Top 7 Python development trends in 2022–23—Python trends 2022–23. *Zestminds*. <https://www.zestminds.com/blog/top-python-development-trends/>
- Zhang, H. (2004). *The optimality of Naive Bayes*. 1, 3. <http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf>
- Zhou, Z.-H. (2021). *Machine learning*. Springer Singapore. <https://doi.org/10.1007/978-981-15-1967-3>

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California



DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE