2023 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA) | 979-8-3503-0912-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/SUMMA60232.2023.10349514

Aircraft Accident Prediction Using Machine Learning Classification Algorithms

Rade Kačar
Faculty of Transport and Traffic Engineering
University of Belgrade
Belgrade, Serbia
kacarrade@gmail.com

Darko Ćulibrk
Planning and Development of ISP Network
MTEL
Banja Luka, Bosnia and Herzegovina
Darko.Culibrk@mtel.ba

Olja Čokorilo
Faculty of Transport and Traffic Engineering
University of Belgrade
Belgrade, Serbia
oljav@sf.bg.ac.rs

Petar Mirosavljević
Faculty of Transport and Traffic Engineering
University of Belgrade
Belgrade, Serbia
perami@sf.bg.ac.rs

***Abstract*—Air traffic and transportation is the safest form of traffic despite the fact that every year in the world there are a significant number of air traffic accidents and incidents. Safety management and Risk management have always been extremely important factors in aviation. Safety improvement is possible through the constant detection and control of hazards as well as causes of accidents and incidents and hard work on their mitigation, removal, or reduction of their consequences. The aim of this paper is to present the application of machine learning classification in air crash severity prediction.**

Keywords—Aircraft Accident, Prediction, Machine Learning Classification Algorithms, Multilayer Perceptron, Artificial Neural Networks, XGBoost, LightGBM.

I. Introduction

Air traffic systems became multilayer, hyperdimensional, highly distributed, and interdependent with levels of complexity that were unimaginable until just a few decades ago. That is why maintaining a high level of safety in such a complex environment is more challenging than before [1]. Civil aviation is a complex mixture of many different but interrelated human, technical, environmental, and organizational factors that affect the safety and performance of the system. In the early days of commercial aviation large number of aircraft accidents was a characteristic. The priority of all safety processes is accident prevention, but at the beginning of the aviation era aircraft accident investigation was the main tool of prevention. Nowadays a proactive approach to safety is applied. It means that stakeholders should collect data to predict not only real and current but also upcoming safety risks. In this situation, safety analytics must be improved to predict future safety risks and safety performance. It is of utmost importance that techniques and methods for recognizing and predicting adverse safety events are devised and widely used. Nowadays is a time of data abundance and technological prosperity, which opens a big door where artificial intelligence and machine learning can enter every pore of our reality. In this work, we present a machine-learning algorithm for aircraft accident prediction. The main idea is to support a proactive safety approach. This technique could find its place in SAR (Search and Rescue) missions as an air crash severity prediction tool to optimize the engagement of resources in SAR operations. Machine learning is a very powerful technique that can use data to train algorithms and give computer systems the ability to "learn" (i.e. progressively improve performance on a specific task) from data, without being explicitly programmed [2]. Machine Learning is a modeling technique that classifies data in a way that figures out the model out of data - the model is the final product of Machine Learning [3].

II. Related Work

Today, machine learning is successfully used as a method for safety and risk analysis. Paper [4] proposed a Support Vector Machine, Random Forest, Gradient Boosting Classifier, K-Nearest Neighbors Classifier, Logistic Regression, and an Artificial Neural Network machine learning algorithms used to predict aircraft crash severity. The dataset is obtained from The National Transportation Safety Board (NTSB). Of all parameters given in the dataset, 9 were chosen for the study. The prediction was made on the basis of 9 categories obtained by the combination of two group categories: categories based on damage dealt and categories based on fatalities. Different algorithms gave different accuracies, from 90% up to 91,66%.

Paper [5] suggests the use of Tree-AS, XGBoost Linear, XGBoost Tree, CHAID (Chi-squared Automatic Interaction Detection), and Neural Network. The dataset contains 17 fields with information on the location of the crash. Causes were classified into seven categories. The highest accuracy which was obtained using the mentioned methods was 40%.

Paper [6] proposed the use of Support Vector Machines (SVM), K-Nearest Neighbors (KNN), AdaBoost, and XGBoost.

Paper [7] proposed the use of dataset from Aviation Safety Reporting System (ASRS), use of the hybrid model support vector machine and an ensemble of deep neural networks.

III. Methodolgy

Several stages were used in the proposed methodology to make machine learning (ML) models for the prediction of the severity of aircraft accidents. Those stages are common in machine learning projects [8, 9, 10]. Each model is performing a classification prediction task, i.e., predicting class labels for a given set of inputs. The models developed will predict severity levels in case an accident already happened, and that is the premise to have in mind. Stages are graphically represented in Fig. 1.

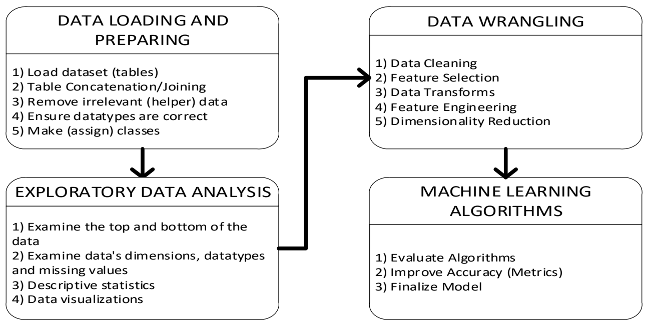
DATA LOADING AND PREPARING
1) Load dataset (tables)
2) Table Concatenation/Joining
3) Remove irrelevant (helper) data
4) Ensure datatypes are correct
5) Make (assign) classes
EXPLORATORY DATA ANALYSIS
1) Examine the top and bottom of the data
2) Examine data's dimensions, datatypes and missing values
3) Descriptive statistics
4) Data visualizations
DATA WRANGLING
1) Data Cleaning
2) Feature Selection
3) Data Transforms
4) Feature Engineering
5) Dimensionality Reduction
MACHINE LEARNING ALGORITHMS
1) Evaluate Algorithms
2) Improve Accuracy (Metrics)
3) Finalize Model

Fig. 2. Stages of predictive modeling

A. Data Loading and Preparing

The Aviation accident database that we used is from the National Transportation Safety Board (NTSB). The data has been extracted from the "avall.mdb" file, NTSB Aviation Accident Database. The database contains facts about accidents and incident events starting from the 1st of January 2008 up to the 30th of January 2022.

From dozens of parameters contained in the database, we have chosen 39, which we considered important for the prediction problem. We took parameters (table columns) from three tables: "events", "Flight_Crew" and "aircraft". Some of the parameters are numerical in nature, and some are categorical. Part of the parameters have helper functions for table grouping, merging, and reformatting, and part is data used mostly as input to machine learning models. Those last represent attributes i.e. features of input variables. Input variables, in form of arrays of values (vectors), are actual events (represented as table rows). The list of chosen parameters is shown in Table I.

TABLE I. PARAMETERS CHOOSEN FOR THE STUDY

Table "events"			
Parameter (Table column name)	**Description**	**Role**	**Kept or Dismissed**
ev_id	Unique Identification for Each Event	helper	dismissed
ev_type	Type of Event	data, input	dismissed
ev_dow	Event Day of the Week	data, input	kept
ev_time	Time of Event	data, input	kept
ev_year	Event Date Year	data, input	kept
ev_month	Event Date Month	data, input	kept
ev_nr_apt_loc	Indicates whether the accident/incident occurred off or on an airport	data, input	kept
apt_dist	Distance from the airport in statute miles	data, input	kept
apt_elev	Airport Elevation	data, input	kept
light_cond	Lighting Conditions	data, input	kept
sky_cond_nonceil	Sky Lowest Cloud Conditions	data, input	kept
sky_nonceil_ht	Lowest Non-Ceiling Height	data, input	kept
sky_ceil_ht	Lowest Ceiling Height	data, input	kept
sky_cond_ceil	Sky Condition for Lowest Ceiling	data, input	kept
vis_sm	Visibility (Statute Miles)	data, input	kept
wx_temp	Air Temperature at event time (in degrees Celsius)	data, input	kept
wx_dew_pt	Dew Point at event time (in degrees Fahrenheit).	data, input	kept
wind_dir_deg	Wind Direction (degrees magnetic)	data, input	kept
wind_vel_kts	Wind Speed (knots)	data, input	kept
gust_kts	Wind Gust (knots)	data, input	kept
altimeter	Altimeter Setting at event time (in Hg)	data, input	kept
ev_highest_injury	Event Highest Injury	data, output	kept
wx_cond_basic	Basic weather conditions	data, input	kept
dec_latitude	Event Location Latitude decimal	data, input	kept
dec_longitude	Event Location Longitude decimal	data, input	kept
Table "Flight_Crew"			
Parameter (Table column name)	**Description**	**Role**	**Kept or Dismissed**
ev_id	Unique Identification for Each Event	helper	dismissed
Aircraft_Key	ID's unique aircraft in collisions	helper	dismissed
crew_no	Unique Identifier for Each Pilot	helper	kept, changed
crew_age	Indicates the age of the flight crew member in years	data, input	kept, changed
med_certf	Medical Certificate Class	data, input	kept, changed
med_crtf_vldty	Medical Certificate Validity	data, input	kept, changed
Table "aircraft"			
Parameter (Table column name)	**Description**	**Role**	**Kept or Dismissed**
ev_id	Unique Identification for Each Event	helper	dismissed
Aircraft_Key	ID's unique aircraft in collisions	helper	dismissed
damage	Damage	data, output	kept
cert_max_gr_wt	Certified Max Gross Weight	data, input	kept
acft_category	Category of the involved aircraft	data, input	kept
homebuilt	Is aircraft amateur-built	data, input	kept
afm_hrs_last_insp	Airframe hours since the last inspection	data, input	kept
afm_hrs	Airframe Hours	data, input	kept

The next step was to create a unique dataset by grouping data inside tables, and merging all tables into one spreadsheet; the goal was to have a unique sample (sample, instance, or observation is represented as a table row) per unique aircraft involved in an accident. In this process, the meaning of some original parameters was changed and substituted with new ones. The meaning of "*crew*_no" is changed to the Number of crew per aircraft, the meaning of "*crew_age*" to the Mean value of crew age per aircraft, "*med_certf*" and "*med_crtf_vldty*" now represent Medical Certificate Class and Medical Certificate Validity of Pilot, per aircraft. Some number of helper variables were removed during this process. Part of this stage was also to ensure that data types in the final table are correct.

B. *Problem Definition*

Our goal was to predict the severity of aircraft accidents based on NTSB classification. Severity is defined by a combination of two parameters - Event Highest Injury (can be Fatal, Serious, Minor, or None) and Damage imposed to aircraft (can be Destroyed, Substantial or Minor). Combining those two parameters we get nine values, or categories for, as we named it, the Severity class. Categories are shown in Table II.

TABLE II. SEVERITY CLASSES

Event Highest Injury	Damage	Severity class
Fatal	Destroyed	fatal-destroyed
Fatal	Substantial	fatal-substantial
Fatal	Minor	fatal-minor
Serious	Destroyed	serious-destroyed
Serious	Substantial	serious-substantial
Serious	Minor	serious-minor
Minor or None	Destroyed	minor-destroyed
Minor or None	Substantial	minor-substantial
Minor or None	Minor	minor-minor

In the taxonomy of machine learning our problem falls under the supervised machine learning branch, multiclass classification task. Every input variable (sample) is mapped to one of the nine output variables (targets, class labels) in the used dataset (ground truth data). The goal of classification (as a subcategory of supervised learning) is to predict a categorical class label of new observation (event) based on past observations (through learned, i.e. fitted, machine learning model).

C. *Exploratory Data Analysis*

In this stage, we did standard Exploratory data analysis (EDA) tasks: examining the top and bottom of data, examining the data's dimensions, data types, and missing values, and did descriptive statistics and data visualization. Although most of those tasks were done in this stage, EDA is an iterative process, and it was used as necessary in any following stage.

D. *Data Wrangling*

This stage is also known as data pre-processing, data munging, or data preparation.

The first task in this stage is Data cleaning. We removed irrelevant and unwanted data, that is rest of helper features (columns) we do not need anymore, any sample where aircraft were home-built ("*homebuilt*" equal to Yes), and any sample where aircraft type is different from an airplane ("*acft_category*" not equal to Airplane). In this way, we focused on the most interesting types of aircraft.

We also removed all samples where the Severity class is equal to the "NaN" value – this was the consequence of unknown or empty values in either the "*ev_highest_injury*" or "*damage*" columns. After this step we already knew we are facing the toughest problem in classification – imbalanced multi-class classification. Table III shows number of data samples per Severity class.

TABLE III. NUMBER OF SAMPLES PER SEVERITY CLASSES

severity class	number of samples
fatal-destroyed	1683
fatal-substantial	1906
fatal-minor	20
serious-destroyed	107
serious-substantial	1271
serious-minor	28
minor-destroyed	170
minor-substantial	11179
minor-minor	546

Next to do in the Data cleaning stage is to check if are there any columns with single values (zero-variance columns). If there are some, we have to remove them because they don't bring any value to the learning process. We also identify columns with very few different values (near-zero-variance columns). For columns with very few values, we should consider transforming them from numerical to categorical before we decide to remove them. In this step, we transformed "*ev_month*" data type from integer (numerical) to string (categorical). Then we identified rows with duplicated data. There were duplicated rows, but in our case, they represent different airplanes with the same values for input features, so we didn't discard any row in this step.

After that, we identified data containing unknown values or wrong ones. Usually, unknown data were marked by the operator in some way (using "999" for example in "*gust_kts*" or "-1" in "*apt_dist*"). In other cases, there were wrongly inputted data (for column *'dec_longitude'* values less than -180 and greater than 180 are treated as wrong, for example). All detected unknown and wrongly inputted data were replaced with "NaN" at this step.

Then we needed to decide how to deal with missing values ("NaN"). There can be many strategies on how to do it; we decide to drop all columns with more than 50% of "NaN-s". The rest of the missing values can be replaced with the mean, median, or mode of their respective columns. We postponed this step to be done after the dataset train-test split, to avoid data leaking.

Data cleaning also implies Outlier detection. Outliers are values out of the expected range. Outliers can be obvious errors (like Air Temperature at an event time equal to 3000 degrees Celsius) or in some cases rare events with no significant influence on the population. Removing outliers can improve machine learning model skills. The standard method used for detection is Interquartile Range (IQR) Method, or in case data have Gaussian or Gaussian-like distribution we can use Standard Deviation Method. After analyzing our dataset, we decide to use the IQR method with the upper 99% percentile to clip very extreme values and keep enough data for model training.

After Outlier detection and removal, and before any statistical operations, we had to split the dataset into training and test sets. All following data preparation tasks should be performed on the training set first. In this way, we are assuring that any information about data in the hold-out test set won't be available to the training set. This problem is known as Data Leaking and can decrease the performance of ML models. We used the simplest method – splitting the dataset into one training and one test set. The more robust solution would be to use three (training, validation, and test) or four (training, training-validation, validation, and test) sets, or to use *k-fold* cross-validation [11, 12].

The second task in the Data Wrangling stage is Feature Selection. It is the process of choosing the most important features the for ML model. By reducing the number of features we are making an effective ML model – it will use less computational power and take less time to run. One of the ways how Feature Selection can be seen is in terms of Unsupervised and Supervised selection. Unsupervised selection methods don't take into count the output variable (target), and Supervised methods do.

Under the Unsupervised method, we first tested how strongly features are related to each other. Statistical measures used for numerical features were Pearson correlation (good for linear relationships between data) and Spearman correlation (good for non-linear relationships between data). Our test showed a high correlation value between the "*wx_dew_pt*" and "*wx_temp*" features (Pearson equal to 0.81, Spearman equal to 0.77), so we have removed the "*wx_dew_pt*" column as a consequence. Then we checked the correlation between categorical features using Phi-k correlation and Mutual information methods. This test didn't show any strong correlation between categorical data.

Under the Supervised method, we used statistical tests to measure the relationship strength between features and categorical targets (Severity class labels). For testing numerical features, we used the ANOVA (analysis of variance) test. The result was "*apt_dist*" and "*gust_kts*" features have no significant relation with the target, and "*ev_time*", "*ev_year*" and "*wind_dir_deg*" have small relation with the target. So, we discarded those five columns. For testing categorical features, we used the Mutual information test. The result was "*ev_dow*" and "*ev_month*" features have no significant relation with the target, so we discarded those two columns.

All statistical tests above have been performed on the training set, and results were applied to training and test sets respectively. The same rule goes for the next two tasks.

The third task in the Data Wrangling stage was Missing Data Imputation. We have replaced "NaN" values in numerical columns with median values. If we are sure some column contains data under Gaussian distribution, we can use mean value, but for the sake of simplicity, we were using median since we didn't test columns on statistical distribution. For categorical columns, we use the mode (most frequent) value to replace "NaN" -s.

The next task in Data Wrangling was Data transforms – changing the scale or distribution of data. Many machine learning algorithms benefit if input variables are scaled to the standard range, often between 0 and 1. It includes algorithms that weight inputs like neural networks, and algorithms that use distance measures, like k-Nearest Neighbors. Also, scaling is useful for optimization algorithms in the core of ML algorithms, like gradient descent [8]. We performed scaling per numerical columns so all values lie in the range between 0 and 1 in the training set.

The fifth task in Data Wrangling was Feature Engineering. Feature Engineering is the process of creating new features based on existing ones. Since ML models require all inputs to be numbers, we perform one-hot encoding on categorical columns. One-hot encoding on one categorical column will take all variables, and for every unique value, it will create a new column. A new column will be composed of 1s or 0s (binary value) – value 1 will signify if that row has a categorical value. The process is repeated for all categorical columns. In the end, we have a dataset with all values being numerical and with an increased number of columns, i.e. features. After one-hot encoding we performed Normalization – we rescaled every row to have a length of one (unit vector). This is useful for sparse datasets (lots of zeros), like the one we got in the previous step [9].

We didn't use any dimensionality reduction method, but in practice, this is a standard task and can improve the performance of the model.

E. Machine Learning Algorithms

The problem we had to solve was classification predictive modeling, but what posed a challenge for us was the unequal distribution of classes in the training dataset. This is a so-called imbalanced classification problem. Machine learning algorithms used for classification are mostly based on the assumption there is an equal, or almost equal, number of examples in every class. However many real-world examples have an imbalanced class distribution, so we need special techniques and methods to use in the modeling process. Another problem is that most of the literature on imbalanced classification is focused on binary classification problems (all examples belong to one of two classes), and it is very hard to find papers or examples of how to solve multiclass classification problems (all examples belong to one of three or more classes). The slightly imbalanced dataset can be treated as a normal classification problem and we can use standard techniques, but severely imbalanced problems (ratio of classes is 1:100 or 1:10000 for example) require special attention and modifications to learning algorithms.

Under this circumstance of data imbalance, most machine learning algorithms will suffer in performance degradation, not only because of imbalance in data (the algorithm will tend to predict the majority class, and can have the same performance as random guessing); other causes of degradation are dataset size effects (insufficiency of information, poor generalization of data characteristics), label noise effects (class noise – mislabeled examples for some class) and data distribution effects (no clear boundaries between classes in feature space, i.e. no good class separability) [13].

In the case of imbalanced multiclass classification, we have to use appropriate evaluation metrics. An evaluation metric is a measure to quantify the performance of a predictive model. Vastly used metric for classification tasks is Classification Accuracy – the number of correct predictions divided by the number of total predictions. But in the case of imbalanced classification, this measure fails. This metric will mirror the distribution of classes and we can get very high results, although in reality trained model is no better than an unskilled classifier. This will lead to erroneous conclusions about our model. This situation is referred to as the Accuracy Paradox. From the plethora of evaluations metrics we choose Weighted F1 Score (1) and Balanced Accuracy (3). We also kept Classification Accuracy as a reference. Weighted F1 Score is defined as:

$$Weighted\ F1\ Score = \sum_{i=1}^{N} w_i \times F1\ Score_i \quad (1)$$

where:

$$w_i = \frac{Number\ of\ samples\ in\ class\ i}{Total\ number\ of\ samples} \quad (2)$$

and N is number of classes in the dataset.

Balanced Accuracy is defined as the average of recall obtained on each class, i.e. the macro average of recall scores per class (as implemented in Python scikit-learn library):

$$Balanced\ Accuracy = \frac{1}{N}\sum_{i=1}^{N} Recall_i \quad (3)$$

By using Weighted F1 Score we want to assign greater contribution of class with more examples in the dataset. Alternatives can be Macro F1 Score to treat all classes equally. Micro F1 Score has the same value as Accuracy in the case of multi-class classification. Balanced Accuracy is a good choice of metric in case true negatives have the same importance as true positives, while F1 Score doesn't care about true negatives. In the case of a balanced dataset, Balanced Accuracy is equal to Accuracy.

The baseline ML model we started with is Multilayer Perceptron (MLP), a class of Artificial neural networks (ANN). Since our baseline MLP network manifested a generalization problem (model overfits training data set), we applied some techniques for better generalization [14], namely: adding Dropout Layers and using Weight Constraints. We didn't use any hyperparameter tuning techniques, we heuristically tried a few hyperparameter combinations and chose one with the best evaluation metrics. Other ML algorithms we used were Extreme Gradient Boosting (XGBoost) and Light Gradient Boosted Machine (LightGBM) – a decision tree-based ensemble learning methods.

All ML algorithms were modified and tested with techniques that take class imbalance in the count. Those techniques encompass configuring algorithms themselves, or additional pre-processing of data (undersampling and oversampling of training data for example). Output variables are also required to be numbers, so we perform output label transformation, depending on the algorithm used. Table 4 shows ML models along with configurations and techniques used. Although the terms algorithm and model can be used interchangeably, we can think about the algorithm as a process of learning from data, and the model as a specific representation learned from data [15].

TABLE IV. MACHINE LEARNING MODELS USED

Model No	Algorithm	Configuration		Additional configuration and preprocessing
1	MLP	***Layers***	512 + 256 + 9 nodes	None
		Loss function	Categorical Crossentropy	
		Optimizer	Adam	
2	MLP	***Layers***	512 + 256 + 9 nodes	Keras class weight parameter
		Loss function	Categorical Crossentropy	
		Optimizer	Adam	
3	MLP	***Layers***	512 + 256 + 9 nodes with Dropout	Keras kernel constraint and bias constraint parameters
		Loss function	Categorical Crossentropy	
		Optimizer	Adam	
4	MLP	***Layers***	512 + 256 + 9 nodes	Oversampling: SMOTE[a], Undersampling: Tomek Links
		Loss function	Categorical Crossentropy	
		Optimizer	Adam	
5	MLP	***Layers***	512 + 256 + 9 nodes with Dropout	Keras kernel constraint and bias constraint parameters. Oversampling: SMOTE, Undersampling: Tomek Links
		Loss function	Categorical Crossentropy	
		Optimizer	Adam	
6	XGBoost	***Objective***	multi:softmax	XGBClassifier parameters max depth, subsample and colsample bytree
		Booster	DART	
7	XGBoost	***Objective***	multi: softmax	XGBClassifier parameters max depth, subsample and colsample bytree. Oversampling: SMOTE, Undersampling: Tomek Links
		Booster	DART	
8	LightGBM	***Objective***	multiclass	LGBMClassifier parameters max depth, subsample and colsample bytree.
		Booster	DART[b]	

[a] Synthetic Minority Oversampling Technique

[b] Dropouts meet Multiple Additive Regression Trees

IV. RESULTS

Table V shows the results of the evaluation of different machine learning models on the test data set.

TABLE V. RESULT TABLE

Model Number	Accuracy	Weighted F1 Score	Balanced Accuracy
1	0.60	0.58	0.17
2	0.54	0.56	0.20
3	0.67	0.55	0.12
4	0.51	0.54	0.19
5	0.25	0.33	0.27
6	0.68	0.60	0.15

7	0.53	0.57	0.21
8	0.68	0.59	0.15

Higher results are achieved using boosted tree algorithms – XGBoost and LightGBM when compare to MLP. Although performance on the training set was great, as a consequence of overfitting (bad generalization) our models didn't achieve results above 0.7 (for Accuracy and Weighted F1 Score) on the validation (test) set. More attention to techniques for better generalization must be paid.

V. Conclusion and Future work

This paper proposed a methodology for predicting the severity of aircraft accidents if such already happened. Severity was categorized into 9 classes based on the highest injury in the event, and the damage level of the aircraft. We have tested 3 different ML algorithms through 8 different ML models.

In the process of predictive modeling, we faced a very common problem in practice – imbalanced multiclass classification. Our models didn't exhibit very high results, and there is a lot of room for improvement. Results can be improved by using more accurate evaluation techniques like K-fold cross-validation. Also, algorithms can be improved by hyperparameter tuning procedures using Python libraries like Scikit-learn's Grid Search, Keras Tuner, or Optuna.

Better results can be reached using more Ensemble learning algorithms (we tried XGBoost and LightGBM), or primitive ensemble methods for multiclass classification represented as a set of binary classification problems – One-vs-Rest and One-vs-One strategies [16].

Dimensionality reduction techniques can also improve the performance of models in some cases, and that can be the subject of future work. Some popular techniques for Dimensionality reduction are Principal Component Analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), and Autoencoders. Considering the [17] regarding the use of computers and software in search and rescue operations, as well as their planning, and especially regarding crisis management and the need to help managers in quick decision-making, further development of software based on this methodology would be a great step forward for science and practice.

References

[1] "EASA Preliminary Safety Review—2017", EASA, 2018.

[2] "AI in air traffic management", SESAR joint undertaking, 2018.

[3] Phil Kim, "MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence", 2017.

[4] Jay Mehta, Vaidehi Vatsaraj, Jinal Shah, Anand Godbole, "Airplane crash severity prediction using machine learning", 12th ICCCNT, 2021.

[5] Ved Prakash Gupta, M Sajid Mansoori, Jitendra Shreemali, Payal Paliwal, "Predicting Causes of Airplane Crashes using Machine Learning Algorithms", International Journal of Recent Technology and Engineering (IJRTE), 2020.

[6] Likita J. Raikar, Sayali Pardeshi, Pritam Sawale, "Airplane Crash Analysis and Prediction using Machine Learning", International Research Journal of Engineering and Technology (IRJET), 2020.

[7] Xiaoge Zhang, Sankaran Mahadevan, "Ensemble machine learning models for aviation incident risk prediction", Decision Support Systems Volume 116, January 2019, Pages 48-63

[8] Jason Brownlee, "Data Preparation for Machine Learning", 2020.

[9] Jason Brownlee, "Machine Learning Mastery With Python", 2016.

[10] Nathan George, "Practical Data Science with Python", Packt Publishing, 2021.

[11] Andrew Ng, "Machine Learning Yearning", 2018.

[12] Dr. Adrian Rosebrock, "Deep Learning for Computer Vision with Python, Practitioner Bundle", PyImageSearch, 2017.

[13] Jason Brownlee, "Imbalanced Classification with Python", 2020.

[14] Jason Brownlee, "Better Deep Learning", 2019.

[15] Jason Brownlee, "Master Machine Learning Algorithms", 2016.

[16] Jason Brownlee, "Ensemble Learning Algorithms With Python", 2021.

[17] ICAO, International Aeronautical and Maritime Search and Rescue Manual, 9th ed., vol. 2, 2022.