Aix Marseille School of Economics

Predictive Methods
Machine & Statistical Learning

Forecasting mishandled baggage with machine learning models

Paris-Charles de Gaulle airport

Author :
Olivier JAYLET

Academic supervisor :
Karine GENTE

ADP Supervisor :
Florian BERTOSIO

Submission date :
October 20, 2024

Remerciements

Je tiens à remercier tous ceux qui m'ont soutenu et guidé tout au long de cette dernière année d'étude.

En premier lieu, je tiens à remercier sincèrement Sémi GABTENI, chef de projet, pour sa confiance accordée depuis un an, ses conseils et son implication, qui ont été déterminants pour la réussite de mon mémoire.

Je suis également reconnaissant envers Florian Bertosio, mon superviseur direct et data-scientist, pour son soutien, sa patience et ses encouragements.

Je tiens à exprimer ma gratitude à toute l'équipe DGEOY dans laquelle je n'ai pas eu de difficulté à m'intégrer et à m'épanouir.

J'adresse mes remerciements à tout les data-scientist d'Eulidia que j'ai rencontré et avec qui j'ai pu collaborer de près ou de loin. Leurs idées et contributions professionnelles ont joué un rôle important dans la réalisation de mon mémoire.

Enfin, j'aimerais remercier le corps enseignant et le personnel des universités publiques de la Sorbonne Paris 1, de Tübingen et de Marseille, dans lesquelles j'ai eu la chance d'étudier. Leur dévouement à fournir une éducation complète et enrichissante a été essentiel à mon développement intellectuel.

Pour finir, je tiens à remercier ma famille et mes amis, qui m'ont toujours soutenus lors de ces années passées à l'Université.

Table of contents

Introduction **9**

1 DGO department and Data team **10**

2 Related work **10**

3 Data and Descriptive statistics **12**
3.1 Perimeter of the study and dataset 12
3.2 Descriptive Statistics 14
3.2.1 Seasonality 14
3.2.2 Baggage paths 16
3.2.3 Paths duration 18
3.2.4 Bags input status 22
3.2.5 Minute until flight departure 23
3.2.6 Geographical links 24

4 Imbalanced dataset issue **25**

5 Models **27**
5.1 Random Forest 27
5.2 Extreme gradient boosting 28
5.3 Probability calibration 30
5.3.1 Platt Scaling 30
5.3.2 Isotonic Regression 30

6 Methodology **31**

7 Results **35**

Conclusion **39**

Appendix **40**

List of Figures

1 Map of the different infrastructures . . . 12
2 Average number and percentage of mishandled bags each days of the week . . . 14
3 Average number and percentage of failed bags each 30 minute of a day . . . 15
4 Simplified diagram of the baggage sorting system TBE, at the sub-system level . . . 16
5 Mishandled bags according to positions . . . 17
6 Distribution of paths duration for non-failed and failed bags (without outliers) . . . 18
7 Distribution and scatter-plot of residuals for path duration between $Q1$ and $Q3$ . . . 19
8 Comparison of actual and generated data for path duration between $Q1$ and $Q3$ . . . 20
9 Comparison of actual path duration with path duration with noise added (Q1 < Path duration < Q3) . . . 21
10 Number and percentage of failed bags per input status . . . 22
11 Minute until flight departure distribution . . . 23
12 Minute until flight departure distribution according to bag status . . . 23
13 Proportion difference of mishandled and successful bag . . . 25
14 Data split . . . 31
15 KFold cross-validation . . . 32
16 Training, calibration and testing methodology . . . 34
17 Probability distribution - Model 1 . . . 36
18 Probability distribution - Model 10 . . . 36
19 Confusion matrix - Model 1 . . . 36
20 Confusion matrix - Model 10 . . . 36
21 Probability calibration for XGBoost . . . 37
22 Probability calibration for Random Forest (RF) . . . 37
23 Shap values of the model 10 (RF + Isotonic regression) . . . 38
24 DGO organization chart . . . 40
25 Confusion matrix - Model 2 . . . 42
26 Confusion matrix - Model 3 . . . 42
27 Confusion matrix - Model 4 . . . 42
28 Confusion matrix - Model 5 . . . 42
29 Confusion matrix - Model 6 . . . 42
30 Confusion matrix - Model 7 . . . 42
31 Confusion matrix - Model 8 . . . 43
32 Confusion matrix - Model 9 . . . 43
33 Confusion matrix - Model 11 . . . 43
34 Probability distribution - Model 2 . . . 44
35 Probability distribution - Model 3 . . . 44
36 Probability distribution - Model 4 . . . 44
37 Probability distribution - Model 5 . . . 44
38 Probability distribution - Model 6 . . . 45
39 Probability distribution - Model 7 . . . 45
40 Probability distribution - Model 8 . . . 45
41 Probability distribution - Model 9 . . . 45
42 Probability distribution - Model 11 . . . 46
43 Probability density - Model 1 . . . 47
44 Probability density - Model 2 . . . 47
45 Probability density - Model 3 . . . 47

46 Probability density - Model 4 . . . 47
47 Probability density - Model 5 . . . 47
48 Probability density - Model 6 . . . 47
49 Probability density - Model 7 . . . 48
50 Probability density - Model 8 . . . 48
51 Probability density - Model 9 . . . 48
52 Probability density - Model 10 . . . 48
53 Probability density - Model 11 . . . 48
54 Cyclical encoding of days and minutes . . . 49
55 Mishandled bags according to modules . . . 50
56 Shap values of a random mishandled baggage (1) . . . 50
57 Shap values of a random mishandled baggage (2) . . . 50
58 Shap values of a random mishandled baggage (3) . . . 50

List of Tables

1 Dictionary of Variables . . . 13
2 Path duration quantiles . . . 19
3 Confusion Matrix . . . 25
4 Final sets of hyper-parameters . . . 33
5 List of models . . . 41

Introduction

Paris-Charles de Gaulle airport, one of Europe's leading air transport hubs welcomes millions of passengers from all over the world every year. Passenger satisfaction is a top priority for airport operators, who are constantly striving to improve the efficiency and quality of the services they provide.

A large part of the logistics involves sorting and routing baggage. For safety requirements, any baggage exceeding 115 centimeters (height + width + depth) must be checked in with the airline company. For this reason, airlines and airports must cooperate to ensure the smooth transit of baggage from check-in desks to aircraft holds. Moreover, when a passenger transits through a third-party airport on a journey and shifts from a plane to another, the baggage also has to transit.

In order to manage these interconnections, Groupe ADP and Air France are working together to ensure the most efficient routing of baggage. In 2007, to automate the sorting and routing of baggage ADP built one of the largest Baggage Handling System (BHS)[1] in Europe at Paris-Charles de Gaulle airport. This device uses various types of sensors to detect, locate, monitor and route baggage through the different sections of the BHS. Thanks to these sensors, ADP operators are able to collect a large amount of data. Thus, they can analyse, understand and optimize baggage flow.

The ability to analyse the reasons why a baggage misses its flight is of upmost financial importance. A mishandled bag will cost ADP or the airline an average of 300 euros. One's ADP objective to be cost effective and to perform, is to reduce the number of mishandled baggages. For this purpose, ADP has to be able to quantify and identify the reasons of baggage mishandling.

In this context, the aim of my report is to train and test several classification machine learning models to forecast whether a baggage will be mishandled when it enters in ADP's baggage handling facilities.

The aim of this research is to provide airport operators with valuable tools and analyses to better understand and anticipate baggage sorting operations, which could help to improve operational efficiency at Paris-Charles de Gaulle airport.

In the following chapters, I will present the environment and the team in which I carried out my study. Some previous contributions will be presented, summarising the few studies similar to my subject. Then, I will analyse the descriptive statistics and justify my feature engineering choices. Finally, chapters 4, 5 and 6 explain how the study was carried out and chapter 7 presents the results.

[1] conveyor system installed in airports that transports checked luggage from ticket counters to areas where the bags can be loaded onto airplanes

1 DGO department and Data team

My study was carried out as part of the Data and Management team[2] which is managed by DGO[3] department created in 2019. This department is at the heart of the One Group industrial project. Its purpose is to support, develop and promote the ADP Group's expertise in airport operations and facilitation. It leads the network of international hubs managed by the Group.

This mission, based on the principles of hospitality and responsibility, takes on added significance in the context of the recovery from the covid-19 pandemic, which has put the business at a standstill for part of 2020 and 2021. It is in line with the Group's ambitions in the face of the environmental and societal challenges facing the air transport industry.

DGO works in cooperation with the Paris airports of Orly, Charles de Gaulle and Le Bourget, as well as all those managed by the Group at international level, of which there are more than 25. It manages the services and consultancy provided to ADP customers around the world, which are set to expand with the ADP Airport Services project.

To carry out its mission, DGO supports airports by defining policies for airport operations (both airside and landside), technology and maintenance. It draws on all the expertise of its employees and the wealth of skills of the group's operational divisions, in a multi-local business approach.

The data and management team was created in 2022, with the aim of exploiting the data from the BHS and conducting exploratory analyses. Its main objective is to contribute to the understanding of mishandled baggage and optimise the operational resources used during the baggage sorting process.

The team, led by Sémi GABTENI, is made up of two data scientists and several interns. To increase its development capacities, Eulidia, a consulting firm specializing in data, was commissioned, and five of their data-scientists work full-time on the various development projects. Finally, a data scientist from the baggage logistics team(CDGB) also works with us, enabling us to better collaborate and understand the various baggage logistics professions.

We use the Azure environment to operate using Agile methods and Azure Devops in particular. Finally, Databricks is used to run data ingestion pipelines, to store data in SQL databases and to develop analyses and models using Python notebooks.

2 Related work

There is literature on the logistics of baggage sorting and Baggage Handling System, but only one study has been carried out that is closely related to my topic of failed baggage.

Leeuwen et al. 2020 trained and tested a Light Gradient Boosting Machine to predict for each bag a probability of being a mishandled baggage. The aim of this study is to identify "at-risk" baggage. A logistic model was also tested, serving as a benchmark model.
This study was carried out in collaboration with an airline, and the model developed predicts probabilities only for baggage in transit. Consequently, none of the data used comes from the BHS , but only from data known by the airline, which therefore offers fewer subtleties. Finally, oversampling[4] was used to overcome the problem of unbalanced dataset, without attempting to calibrate the model beforehand. Therefore, the model was trained with generated data.

[2] also called DGEOY

[3] General Management of Operations (In french : Direction générale des opérations). Figure 24 displays its organization chart

[4] method used to address class imbalanced by increasing the number of instances in the minority class.

The final model with the best results has a recall of 0.565, a precision score of 0.506 and an F1 score of 0.534.

In the same purpose of avoiding mishandling bags, Sanden 2020 predicts the probability of a bag to recirculate at least once in the Baggage Handling System. Using sensors data within the BHS, the prediction model achieved a precision of 0.956 and a recall of 0.033 for detecting behavior. When excluding bags that had recirculated before, the recall improved to 0.153 and the precision dropped to 0.764. Even tho the model wasn't trained in the same BHS, it is an interesting approach, as recirculating is a cause of longer processing time by ADP, being able to predict whether or not a bag will make another loop would most probably help to improve mishandling predictive models.

Abdelghanya, Abdelghanyb, and Narasimhanc 2006 presents a model for planning baggage handling facilities at congested airports. The model assigns baggage from departing flights to the available platforms in the baggage handling facilities. The model uses an activity selection algorithm that is typically used to assign limited infrastructure resources to several concurrent activities scheduled in advance. The main objective of the model is to automate the baggage handling process in order to reduce human error and lower the associated operating costs. The baggage handling process comprises three main functions: moving baggage from the check-in area to the departure gates, transferring baggage from one gate to another, and moving baggage from the arrival gates to the baggage reclaim area. The proposed model offers an effective method for planning baggage handling facilities at congested airports, helping managers to assess the trade-offs between different operational constraints to achieve a satisfactory solution.

Van Den Hoven 2019 focuses on improving the utilisation of a parcel sorting system in the parcel industry. The project examines how to increase the utilisation of the main loop of a sorting system while using a maximum of three feed zones. The study proposes the use of buffers to schedule and release parcels on the main loop. Two types of buffers were considered: one requiring parcels to come to a complete stop before being accelerated, and another allowing their speed to be reduced to one of three configured lower speeds. Initial results show that average utilisation with buffers was lower than without buffers, due to the unpredictable nature of actual cycle times. Future research should explore operational disturbances and the possibility of creating a controller that adjusts windows based on relative acceleration moment.

3 Data and Descriptive statistics

A large part of my apprenticeship was spent studying the data. The Baggage Handling System being very complex, it was not trivial to understand the data. A lot of primary intuition wasn't enough to be able to model baggage paths. Numerous discussions with the data-scientists of my team, as well as with the professionals in the CDGB department[5] helped me to understand many biases and differences in between data and actual processes. Those discussions and the analysis of descriptive statistics gave me some idea on how to model the BHS, and which data I could create or modify for some computing and modelling optimisations.

In this section, I'll first introduce you the perimeter of the study and the dictionary of data. Then we will talk about the most relevant descriptive statistics I made, and the feature engineering methods it leaded to.

3.1 Perimeter of the study and dataset

As Roissy airport is very large, it contains several baggage sorting infrastructures (see Figure 1). As you can see, all of them have different shapes, are made with different technologies, and cover different part of the airport. In my study, I worked with the data from the TBE. This perimeter is the one delimited by the red line, and contain sub perimeters, also called "modules". Those are the TBF, TBM, TME, and TMF. In short, those sorters are all part of the TBE and mainly manage baggage in the Terminal 2.

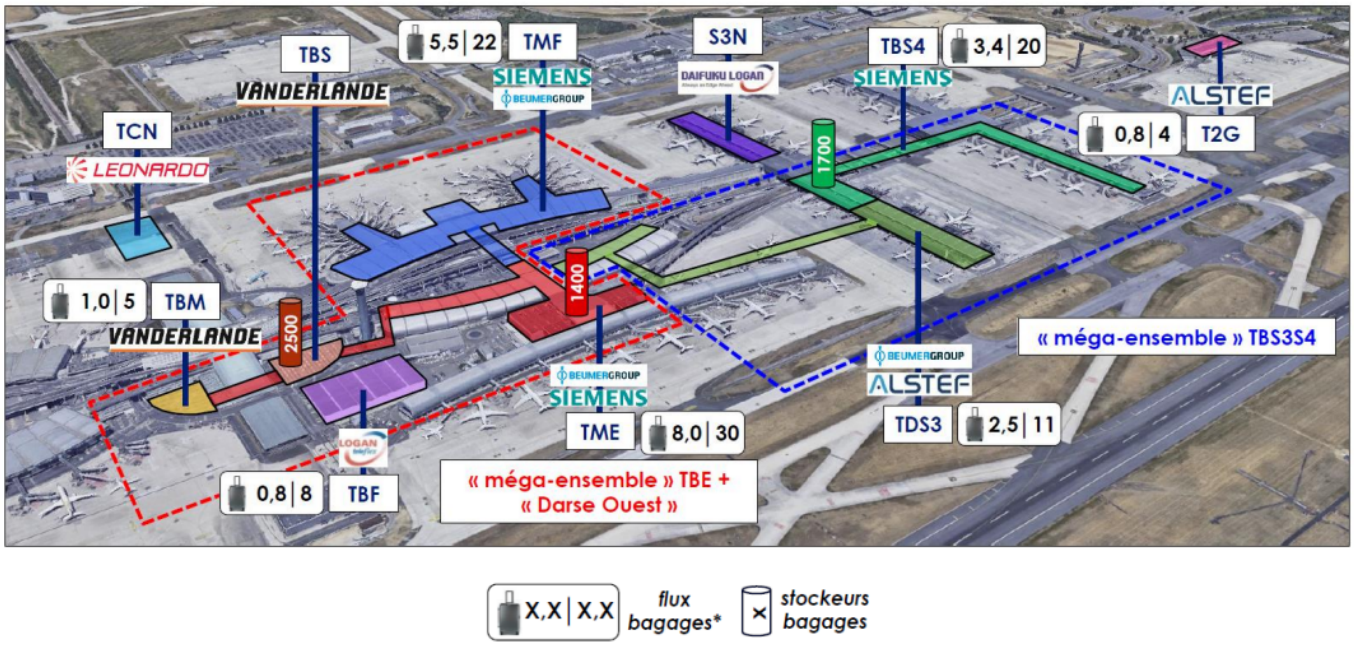


Figure 1: Map of the different infrastructures

Our dataset come from several databases used in the TBE. The first called *fact_paths* contains data of each baggage paths. These data are extracted daily through a pipeline sending requests to the software package used by BHS managers and analysts. In this table, we access the data of each baggage journey (path) in the BHS, from the injection of the baggage into the sorter until its extraction. The second database, *Saria_info_vol* contains flight information such as airlines, departure

[5]Department in charge of baggage logistic

times, departure and destination airports, etc. Thanks to a unique identifier, we are able to link baggage trajectories to information on their inbound and outbound flights. Finally, we could access maintenance data and join some of them to our baggage paths. Those feature not having a significant impact in my models, I won't detail them.

The Table 1 list all data we managed to join. As the aim of this project is to predict the status of baggage when it leaves the BHS, we need to use data available before it enters the sorter. I spent some time listing and sorting data I could use. I ended up with this list, and decided to use only one data usually available after the sorting process. The path duration (*total_path_duration_no_stock*) of a bag is known only after the baggage has been sorted. However, as we'll see in the descriptive statistics, path duration is highly correlated with the probability mishandled bags. And as explained in the introduction, Data and management team is currently training some models to predict the duration of bags before it enters the sorter. In subsubsection 3.2.3 I explain the methodology to add noise to path duration and therefore transform path duration as a proxy of predicted path duration.

Table 1: Dictionary of Variables

Variable	**Type**	**Description**
id_bag_trajet	int64	Bag path id.
input_date	datetime	Date and time baggage entered sorter.
input_module	String	Bag input module.
input_position	String	Sub-system at which baggage enters sorter.
input_status	String	Bag status on entering the sorter.
input_outbnd_flt_dhc	Float	Date and time of flight known when bag enters sorter.
output_module	String	Bag exit module in the sorter.
output_position	String	Bag exit sub-system.
inbnd_flt_hab	datetime	Date and time of outbound aircraft.
input_ICT_BHS	float64	Time in minutes between baggage injection and the SOBT.
total_path_duration_no_stock	float64	Total baggage transit time in minutes, excluding storage
geographic_origin	String	Inbound flight origin.
geographic_dest	String	Outbound flight destination.
SOBT	datetime	Scheduled outbound time of the outbound aircraft.
30mn_nb_bags	int64	Number of bags entered in the last 30 minutes.
30mn_nb_failed	int64	Number of bags failed in the last 30 minutes.
30mn_median_path_duration	int64	Median path duration in the last 30 minutes.
nb_interventions_needed	int64	Number of interventions by a human in the last hour across the TBE
nb_baggage_anom	int64	Number of interventions by a human to correct an issue relating to a baggage (stuck or rolling) in the last hour across the TBE
most_important_priority	int64	Highest priority intervention level in the last hour across the TBE
is_failed	bool	Has the baggage been mishandled ? (Variable to predict)

3.2 Descriptive Statistics

Every analyses and models in this study were carried out using data from August 1st to October 31st 2023. I decided to analyze these three months, as all the data was available and to have two different types of periods. August is known to have a higher passenger (and therefore baggage) flows, while September and October are quieter months.

3.2.1 Seasonality

The sorter opening at around 5 a.m. until 11 p.m. every day of the week, the first intuition we might have about mishandled baggage is the presence of seasonality. For these reasons, it is important to visualize the evolution of missed baggage over time. Figure 2 shows the evolution by day of the week of the number of missed bags, as well as the percentage of rates

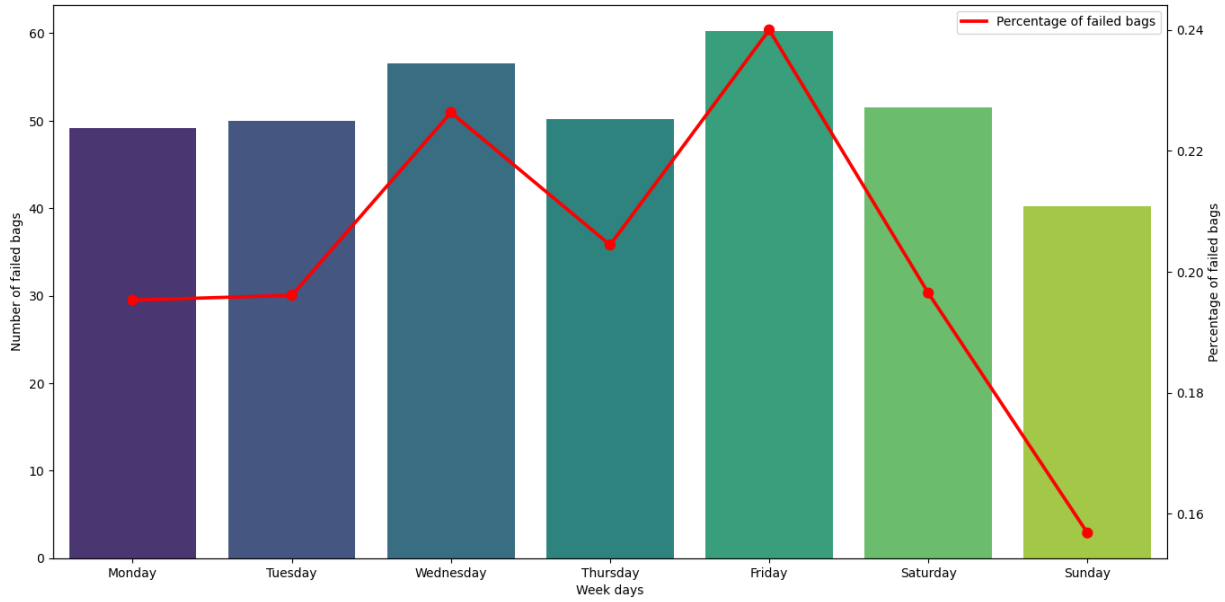


Figure 2: Average number and percentage of mishandled bags each days of the week

It's straight forward to see that failed bags volumes are changing according to the week days. Weekends seem to be less problematic, both in terms of volume and percentage. Intuitively, we might have expected weekends to be more complicated (given the greater number of passengers during weekends). Perhaps because logistics teams are more numerous to better manage the heavy flows at weekends, therefore it can be more difficult to avoid failed bags during the week, with teams that could be smaller. Also, as there is a higher number of flights (and thus of bags) during weekends, if the number of mishandled bags remain constant, its percentage will decrease.

To deeper analyse mishandled bags pattern with regard to time, the Figure 3 displays the average number and percentage of failed bags for each 30 minute of a day.

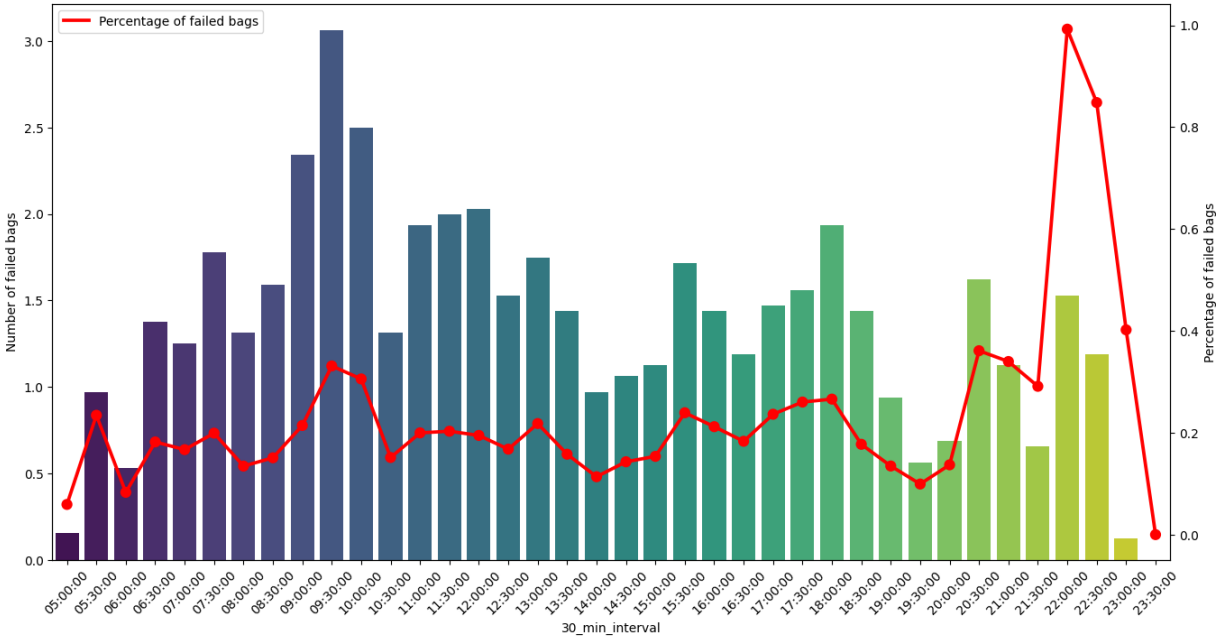


Figure 3: Average number and percentage of failed bags each 30 minute of a day

It turns out there is a peak of mishandled bags at around 9am in the morning. This could be explained by the large number of flight departure in the morning. We can also see that the percentage of mishandled bags is stable until around 9pm, where it suddenly increase to get its peak. This is most probably due to two reasons. Firstly, handlers feed the sorter at the end of the day with a lot of bags which were mishandled and which will get stored into a store during the night. Those bags are already mishandled before entering the sorter and we have to be aware of this bias while interpreting models results. Moreover, there is less flight at this end of the day, so the number of actual traveling bags is lower while the number of mishandled bags (when being injected) is higher.

Thanks to those figures, we can be sure there are some seasonal effects on the probability to fail a bag. To model it and capture seasonal effects, a special encoding process can be applied as day and hours are cyclical features. Following the method of sin and cosines (Pim 2023) seems to be a great strategy as it encodes minute of the day by taking auto correlation in between minutes that might exist. The sinus and cosines transformation are defined as follow :

$$f(x) = \sin\left(\frac{2\pi x}{\text{period}}\right) \tag{1}$$

$$g(x) = \cos\left(\frac{2\pi x}{\text{period}}\right) \tag{2}$$

Figure 54 displays a visualisation of sin and cosine transformations for a few days only. Each days of the week and minutes of the day are smoothly link to each other, and those smoothed lines (especially for minutes of the days) will help the model to recognize seasonal patterns.

3.2.2 Baggage paths

The Baggage Handling System being very complex, one of the biggest challenge for ADP in their project to reduce the number of mishandled bags and optimize their logistic is to understand how bags travel inside the BHS. Using bag paths data, we are able to identify problematic paths, which could be a cause of failed bag. As there is a large number of paths[6], we had to analyse some "typical" or usual paths (paths often used).

To better understand the complexity of the BHS, I will first explain you how bags travels onto sorter and explain what data I decided to use for bag paths. Then I will provide some statistics about those data.

A journey into the BHS Baggage can either come from inbound flights, to be re-routed on inbound flights (i.e. transit baggage), or be injected directly by airlines at check-in counters. Therefore, there are several entry and exit points in the baggage sorting infrastructure.

As showed in Figure 1, Roissy airport is made up of several Baggage Handling System. Siemens and Braumer are the two different manufacturers and technologies that make up the BHS. At the moment, our team only holds data from the BHS using Siemens technologies. The Figure 4 is a simplification of the analysed perimeter (TBE). It provides a macro view of the different routes a baggage can take in the sorter.

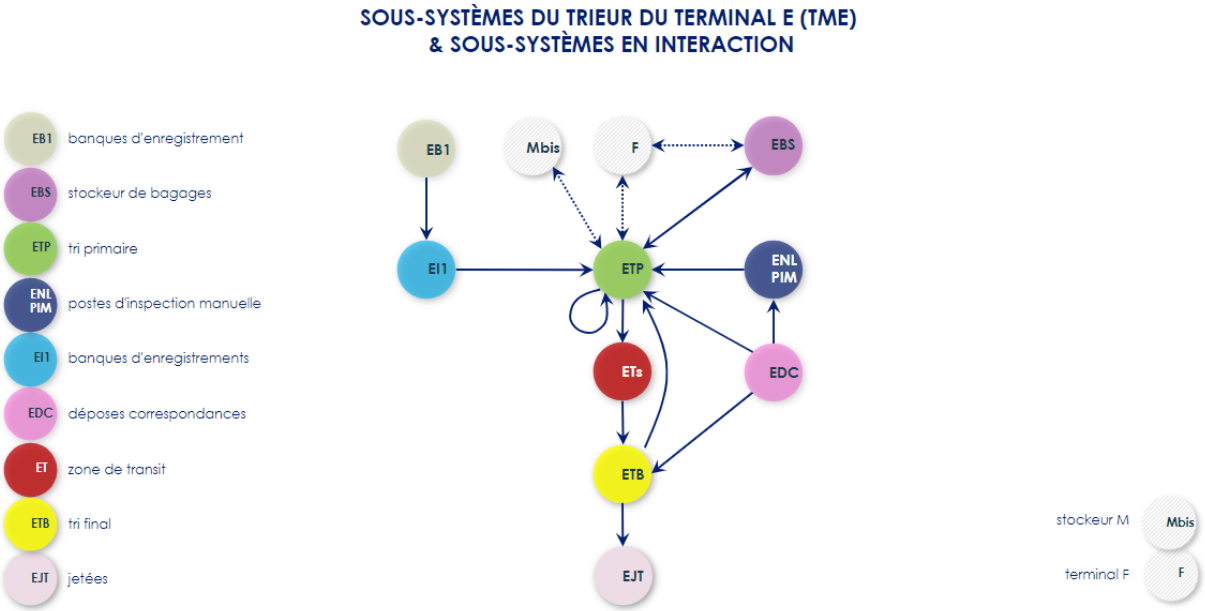


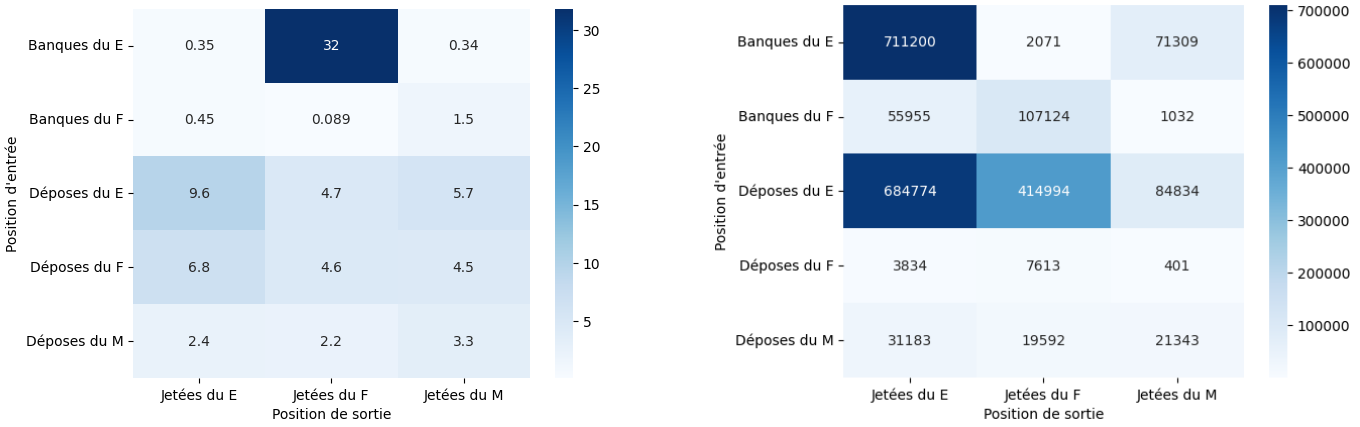
Figure 4: Simplified diagram of the baggage sorting system TBE, at the sub-system level

Here's a very brief and simplified explanation of a baggage path in the BHS : a bag can be injected in the TBE by three entrance, the registration banks, the correspondence deposits, or by the module F (also called TBF). Once a bag enters in the TBE, it is routed to ETP in which it will be rerouted (or sorted) to its final destination. If the baggage's flight has not yet been opened, it will be sent to a storage facility (Mbis or EBS) in which he will wait for his flight to open. If the flight is already

[6]There is actually more than 700 different observed paths

open, the baggage will go to the EJT, where teams of handlers will take it in hand to transport it from the BHS to the plane.

Simplifiying paths In fact, baggage paths within the BHS are far more complex, with many variations. For modeling purposes, I decided to simplify these paths. Instead of taking every checkpoints of a path, I used only the entry and exit points of each route for two different granularities. The first is the module scale (in which modules the luggage enters and exits), the second is the subsystem scale (in which subsystems the luggage enters and exits). These two granularities provide information on the length and difficulty of the journey, without taking in all the information on the journey. Finally, having more than fifty input positions, and 70 output positions, I first made logical groups of positions. For example, to inject a bag in EDC, there are several different positions. Geographically, these positions are all located in the same hangar, and for handlers, all of these positions are in the the same work warehouse. Therefore, it makes sense to reduce the amount of information by grouping input and output positions by subsystem. In the end, we have five input positions and the three main output positions.



(a) Percentage of mishandled bags according to input and output positions

(b) Number of bags according to input and output positions

Figure 5: Mishandled bags according to positions

The Figure 5a allows us to identify the routes for which we have the highest percentage of mishandled baggage. Of all the baggage entering by module banks E (*Banques du E*), and exiting at EJT (*Jetées du E*), 0.35% are mishandled. This ratio is very low, especially when compared with the volume of baggage involved in this route, as shown in the Figure 5b. In fact, this route is the most popular. However, 9.6% of bags entering by EDC (*Déposes du E*) and exiting by EJT (*Jetées du E*) are mishandled, aleven thought this route is also very important in terms of volume. Overall, these two heatmaps allow us to observe the impact of paths from a macro point of view on the probability of a bag being missed.

The same analysis were made with the platforms instead of positions. The Figure 55 displays the number of bags and the difference in percentage of mishandlded bagages according to which modules it enters and exit. Even though there is less variance in the matrix, modules input and output data will be use to train models.

3.2.3 Paths duration

Path duration correspond to the time a baggage spends in the sorter, between entry and exit. In other words, it's the time it takes for ADP's infrastructure to sort and transport a baggage to its destination. These times are very important, because obviously, a baggage spending too much time in the sorter, for whatever reason, could arrive late and miss its flight. They can be seen as indicators of the quality of a baggage's journey through the sorter. The Figure 6 compares the distribution of path duration according to whether they are successful or unsuccessful (mishandled). We can see that mishandled runs have on average much higher times than successful runs (over 20 minutes for unsuccessful runs versus 6 minutes for successful runs).

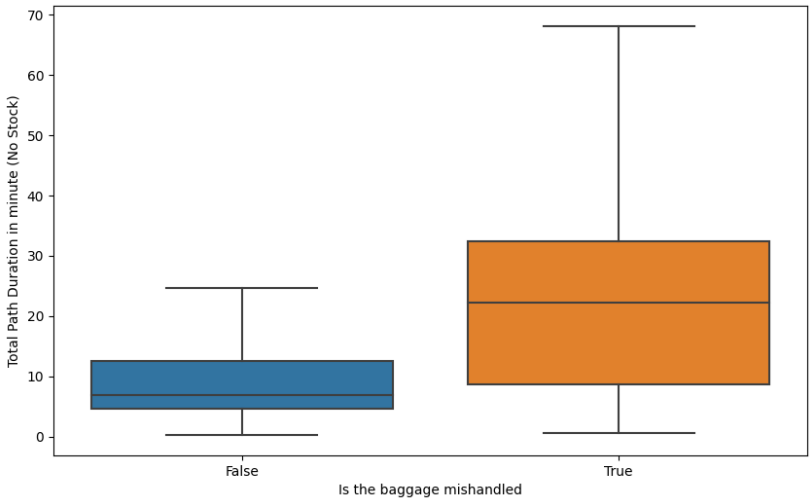


Figure 6: Distribution of paths duration for non-failed and failed bags (without outliers)

For these reasons, travel times can provide a great deal of information for the models, and including them could considerably improve predictions. However, as explained before, we aim to perform our classification at the moment the baggage enters the BHS. The travel time is therefore not yet known at this point. To overcome this problem, two methods have been developed.

Past duration time as an indicator First, the variable *past median duration time* has been created. This corresponds to the median path duration of those who finished their path in the 30 minutes before the baggage to be predicted entered the BHS. This variable could provide the model with information on the quality of baggage handling over the last half-hour.

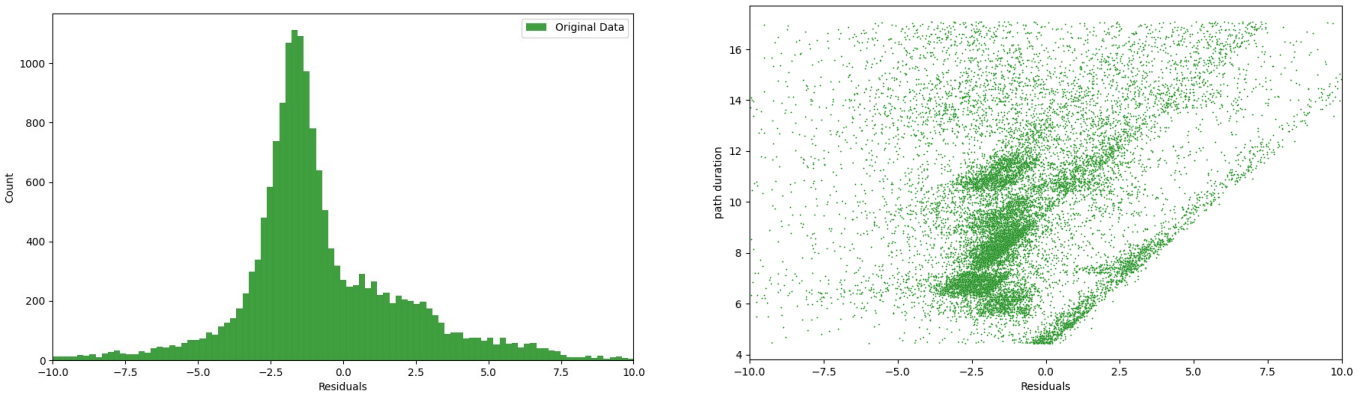
Using model residuals to generate noise Secondly, I drew on a modeling project carried out by my colleague Miracle VODOUMBO. He is developing a model for predicting path duration in the BHS. My idea was to use his predicted travel times to include them as a feature in my prediction model. As Mr. VODOUMBO's model is not yet finished, I decided to use his current residuals to generate noise in the actual travel times. With this method, I add bias to the actual travel times so that they constitute a variable similar to the one that could be predicted in the future.

In order to reproduce the prediction errors as closely as possible, we split the residuals into several groups of travel time quantiles. Indeed, depending on the journey time, the prediction will be more or less accurate. The Table 2 shows the quantile of the travel times used to generate the residuals.

Table 2: Path duration quantiles

Quantiles	**Path Duration (in minute)**
Q1	4.4
Q3	17
D9	28.4
P99	53.3

For each quantile, I have analyzed the residuals of the travel time predictions. The Figure 7a shows the distribution of residuals for runs with a time between Q1 and Q3. Finally, the Figure 7b shows these same residuals crossed with path duration. It can be seen that the residuals are highly dependent on travel times, and that it is important to generate the residuals correctly so as not to bias path duration too much.



(a) Distribution of residuals

(b) Scatter plot of residuals Vs. path duration

Figure 7: Distribution and scatter-plot of residuals for path duration between $Q1$ and $Q3$

For each different quantile group, I processed a kernel density estimation (see Parzen 1962) which is defined by the function :

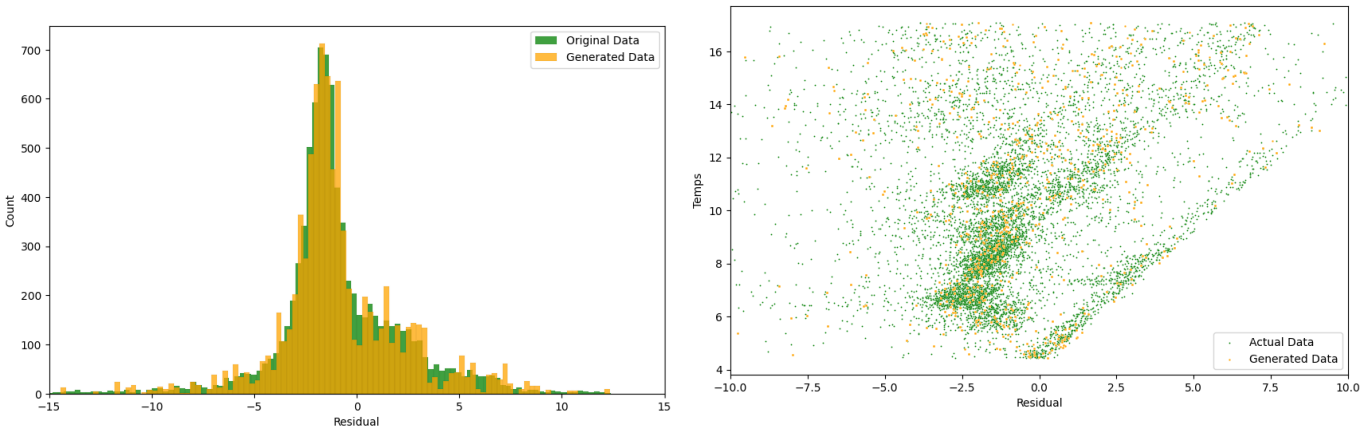
$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{3}$$

where :

- $f(x)$ is the estimated probability density function at point x.
- n is the number of data points.
- h is the bandwidth, a smoothing parameter that determines the width of the kernel function K. h controls the smoothness of the resulting density function. A small h can make the estimate too sensitive to small variations in the data (resulting in an estimate with strong noise), while a large h can over smooth the data, leading to small variance and large bias.
- K is the Gaussian kernel function, which is a symmetric, non-negative function that integrates to one. We can also swap this function with the Epanechnikov, uniform and many others.
- x_i are the data observed from the sample.

The summation $\sum_{i=1}^{n}$ runs over all the data points, and the function $K(.)$ gives the contribution of each data point to the estimation of the density at point x.

Once done, I generated data from this estimated kernel density. Then, for each of the actual path duration, I searched for the nearest generated path duration (with a binary search algorithm) and assigned its associated generated residual. Under these conditions, there are duplicates in the generated residuals attributed to real travel times. As we can see in the Figure 8b, we have more actual than generated data because the only generated data displayed are those assigned to a real travel time. However, the residuals generated are well dispersed and fit well to the residuals of the predictive model.



(a) Distribution of actual Vs. generated residuals

(b) Actual Vs. generated residuals

Figure 8: Comparison of actual and generated data for path duration between $Q1$ and $Q3$

Once the residuals have been generated, I had to add them to the actual path duration. However, some of our residuals are negative, so adding them together could result in negative path duration, which makes no sense. To avoid this, the residuals are added only if they are less than the path duration. If this is not the case, then the absolute value of the residual is added. This makes it possible to add residuals without creating negative running times. We can see the comparison of journey times with noise, compared with their actual times in the Figure 9. The blue curve represents actual path duration, ranked from shortest to longest. The red curve shows the same path duration when the residuals have been added. As can be seen, path duration and residuals are correlated, which was be expected since it's harder for the forecasting model to predict long times than short ones, and the margin of error is also higher.

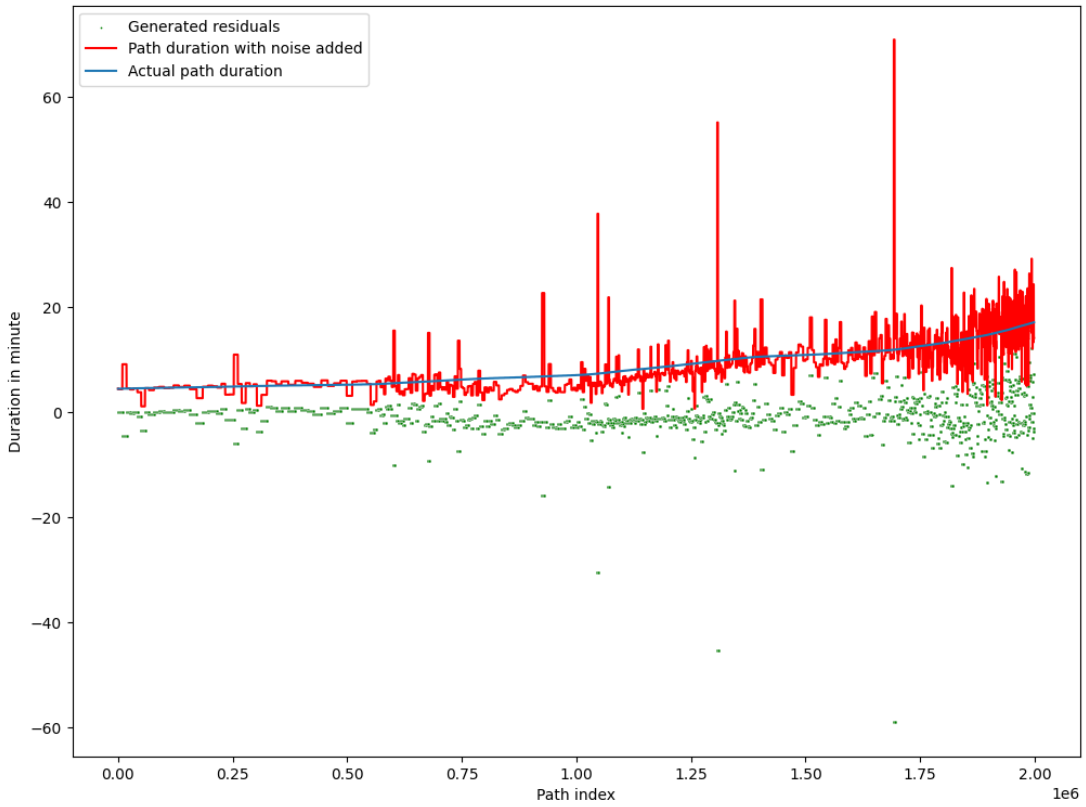


Figure 9: Comparison of actual path duration with path duration with noise added (Q1 < Path duration < Q3)

To conclude, I used the forecasting model of path duration to generate noise and add them up to the actual path duration. This strategy allows me to use the path duration, assuming our department will be able to forecast them before the baggage enters in the TBE.

3.2.4 Bags input status

When being injected onto the sorter, bags are classified according to their status. There exists a lot of status, those indicate how the bag should be sorted. Those status are important and can give information to models.

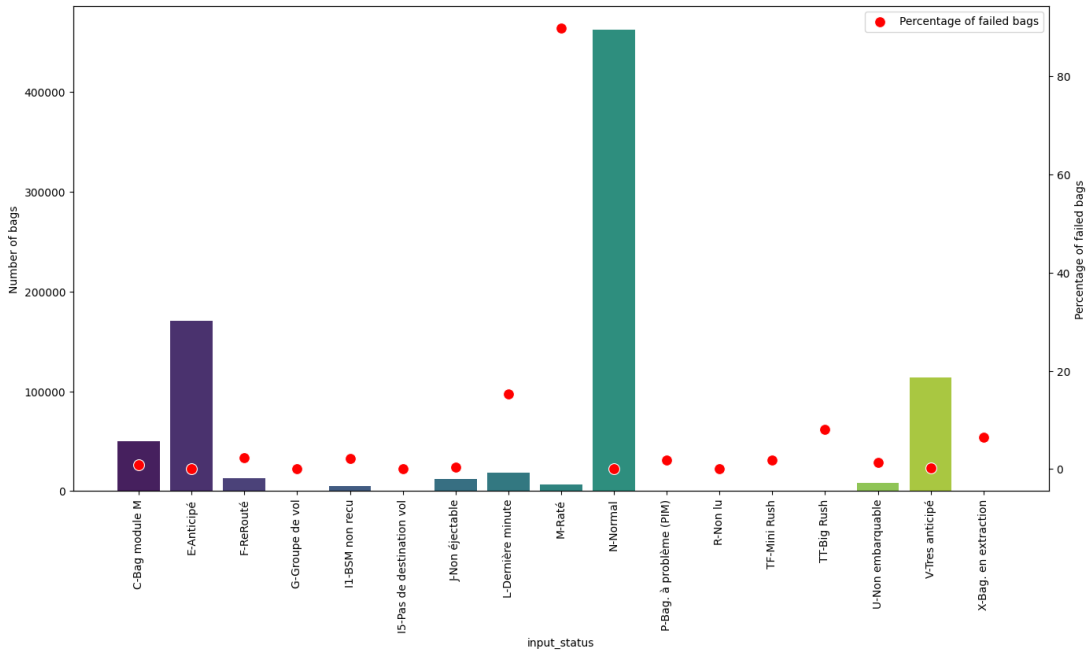


Figure 10: Number and percentage of failed bags per input status

All status having different meanings, they also have different frequency. The *normal* status is the most often used and has a low percentage of failed bags. Bags having this status are usually those injected on time in the sorter, when the outbound flight is open to load bags. When a bag is injected earlier and the outbound flight isn't open to load bags, they take the *anticipated* or *very anticipated* status. We could think that injecting a bag much before loading time would be correlated with a lower percentage rate of failed bags. But the Figure 10 shows a higher percentage of failed bags for *V-Très anticipé* (very anticipated) bags than *N-Normal* ones. This can be explained by the re-rooting of failed bags. If a flight ends its bag loading process while a bag is still in the sorter, this bag's flight will change (*has_flight_changed* will turn **True**) then its status will change to *V-Très anticipé* (as its unlikely to find another flight soon after missing the previous one) and the bag will be store in a stock to wait. The *L-Dernière minute* (last minute) status is given to a bag entering the BHS knowing there is not much time left. If it manages to exit the BHS on time, it will be loaded onto the flight with the last minute procedure (manually instead of being put into a container). This status is a very important information for the model as we can see it the category with the higher failed rate (out of *M-Raté*). In the aim to model bags output status (failed or not failed), we should carefully consider those *input status* because some of them could make the models to over-fit. For example, *M-Raté* means that the bag is failed when it enters in the BHS. We should get rid off those kind of bags.

Finally, here is the list of *input status* we will keep to train and test models : *C-Bag module M, L-Dernière minute, U-Non embarquable, V-Très anticipé, E-Anticipé, P-Bag. à problème (PIM), F-ReRouté, R-Non lu, J-Non éjectable.*

3.2.5 Minute until flight departure

Another very important data available is the number of minute before flight departure (*input_ICT_BHS*). Knowing the *SOBT* of a flight and the *input_date* of a bag, we computed the number of minute in between both. Looking at the Figure 11, it turns out in average, a bag has a bit less than 200 minutes to reach its flight. The distribution looks to be a gaussian distribution, with some extreme value higher than 600 minutes (*V-Très anticipé* bags) and some negative values. Having a negative number of minute until flight departure mostly mean it's almost impossible to reach its flight on time. However, looking at both distributions of minute until flight departure of mishandled and successful bags in the Figure 12, we can see a decreasing proportion of successful bags from approximately 0 to -100 minutes. It means some bags are still managed even when being pushed onto the BHS after the schedule outbound time. This is most likely explained by flights delays. Those visualisations give us a first important cause of mishandled bags, being injected too late in the BHS.

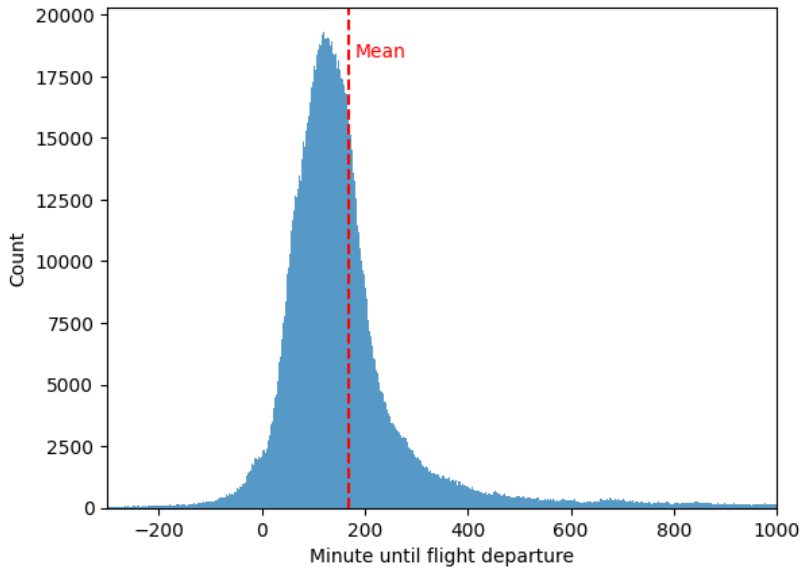
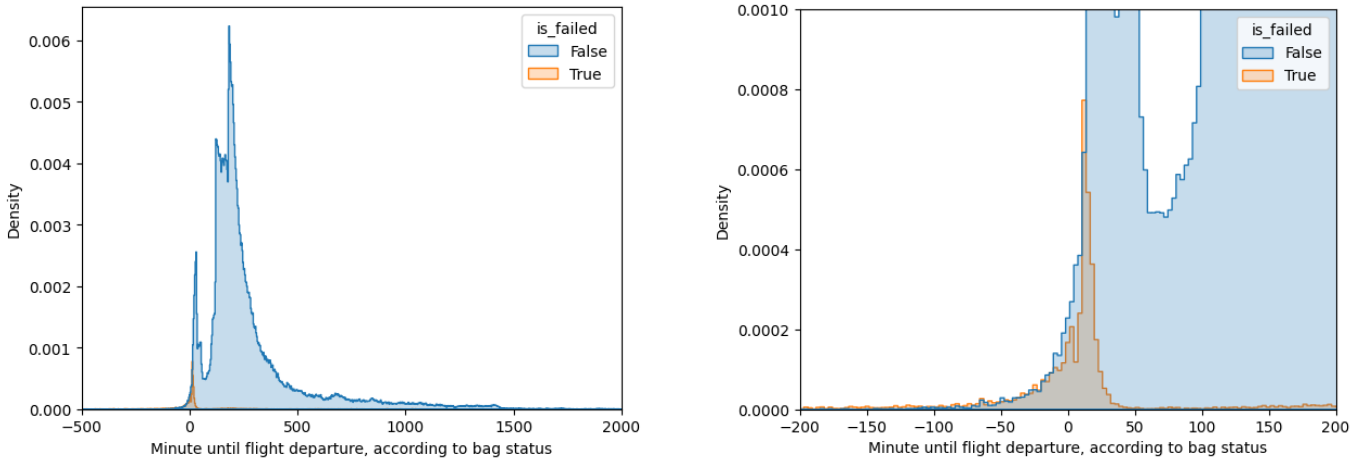


Figure 11: Minute until flight departure distribution



(a) Macro view

(b) Micro view

Figure 12: Minute until flight departure distribution according to bag status

3.2.6 Geographical links

Pretty late during the year, I accessed the flights informations of each bag paths data. This dataset concerns only transiting bags[7]. It gave us additional informations, and the most relevant were geographical origins and destinations. They are key information in the logistic process for two reasons.

- *Ball baggage* : One of the baggage hazards that can delay a bag journey in the BHS process is when bag falls off a conveyor belt. When this happens, a handler is called in and has to walk to the location to put the baggage back on the conveyor belt. The handlers have noted that the concerned type of baggage are very often ball baggage, which as the name suggests, are baggage in the shape of a ball (or rounded bags). Because of this shape, they are more likely to roll and fall off the facilities. Finally, although we don't have any statistics on this type of baggage, handlers have pointed out that more often, the origin or destination of the ball bags are African country. For these reasons, including the origin and destination of a baggage can make it possible to include an additional difficulty linked to the type of baggage.

- *Parking areas* : The second impact of origin and destination is the flight parking areas. As flight companies share terminal and check in doors, they also share parking areas. We don't have exact plans of it, but as the airport is extremely large, the destination and origins of flights could add geographical information within the airport in the model. For example, if a bag comes from an european country and is going to another european country, it is more likely that it uses the same company for both flights. Therefore, the path and process in between its arrival and departure would be shorter. On the other hand, a bag coming from Ireland and taking a flight for China will probably not use the same company. So the parking areas might be more far away from each other.

For the locations, even though we had precised data like name of airports and countries, I decided to use continent only to reduce the complexity of models and avoid unnecessary noise.
To encode those data, I grouped each possible origins and destinations links possible. As we are interested by the links effect (for example a bag from Africa and going to America), it would be more efficient for models to have one column for links (*from Africa to America*) instead of two separated columns for origin and destination. Finally, as those are categorical data I encoded them with the one-hot encoder strategy.

[7]Bags arriving at CDG airport by an inbound flight in the aim to be loaded in an outbound fight with another destination. Those bags doesn't exit at CDG and are pushed in the BHS at their arrival.

4 Imbalanced dataset issue

The question of unbalanced datasets is one of the interesting issues to deal with in machine learning. In a classification, we want to predict which of the different possible events will occur. To train an algorithm, it needs a certain amount of data in order to calibrate its coefficients with the training dataset, and test the best theoretical coefficients with the test dataset. During classification, the different classes may not be equally distributed.

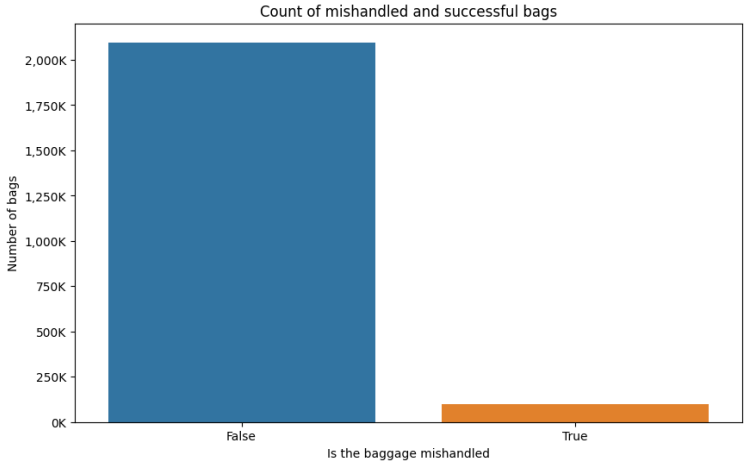


Figure 13: Proportion difference of mishandled and successful bag

In our case of mishandled baggage, the Figure 13 shows the difference between the number of successful and mishandled bags. Fortunately for passengers, the number of mishandled bags is far smaller than the number of successful bags. In our dataset, there are 2 mishandled bags for every 100 successful bags.

This large difference in proportions is problematic, because when training the models, there's a risk of over-training them to predict successful baggage (see Chen et al. 2024).Indeed, if 98% of baggage are not mishandled, the models will be able to get a good score by predicting 100% of successful baggage. However, there is a risk of not being able to predict missed baggage. To overcome this problem, we first take a look at the confusion matrix. Having a look at the Table 3, we can try to understand the confusion matrix concept. This matrix gives four different informations :

Table 3: Confusion Matrix

		Actual	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

- TP : The model predicted a *True* event while it was actually *True*.
- TN : The model predicted a *False* event while it was actually *False*.
- FP : The model predicted a *False* event while it was actually *True*.
- FN : The model predicted a *True* event while it was actually *False*.

Therefore in the context of mishandled baggage issue, we might manage to predict a high number of successful bags (FP will be high) but struggle to predict mishandled bags as it is the minority class (TP will be lower). Having a look at some metrics might help to understand the problem and

how to manage it. The accuracy compute the ratio of good predictions over all the population. It doesn't mind if one of the class got a bad prediction score. Using this metric only can lead to over estimate the capacity of the model. If the model predict 100% of the bags as successful, as it is the majority class, the accuracy might be very high even though the number of TP is null.

$$\text{Accuracy} = \frac{\sum TP + TN}{\sum TP + TN + FP + FN} \tag{4}$$

The recall compute the ratio of TP over the TP and the FN populations. Having a high recall rate means we didn't predict much actually positive status as negative one. In the bags issue, it means we would not predict mishandled bags as successful. Reversely, having a low recall means we predict too much mishandled bags as being successful. This ratio will be very important as it will give information on the ability of the model to predict mishandled bags.

$$\text{Recall} = \frac{\sum TP}{\sum TP + FN} \tag{5}$$

The precision is the ratio of TP over TP and FP populations. It measures how many of the predicted positive instances are actually positive. A high precision indicates that when the model predicts positive values, it is actually correct (So we have a lower number of FP).

$$\text{Precision} = \frac{\sum TP}{\sum TP + FP} \tag{6}$$

Both recall and precision are important to optimise a model, especially when to target classes are imbalanced. However, deciding to optimise the recall to predict the maximum of mishandled-bags would also increase the number of false positive (successful bags predicted as mishandled). Reversely, optimising the model by maximising the precision would increase the false negative, meaning that more mishandled bags wont be predicted. To find the best trade-off between precision and recall, the F1-score combines both metrics. A high F1-score coefficient indicates that both recall and precision are quite good. In such a case, we can consider the model as accurate for positive predictions. In the case of imbalanced dataset, the F1-score will give more balanced measured by considering both precision and recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

5 Models

Before turning to the methodology and results, this section will present the machine learning algorithms and methods used.

I decided to use three different algorithm. The Logisic regression is my benchmark model (as in Leeuwen et al. 2020) and will be use to verify if my other more complex models improve the performance of the predictions. Then, I trained and tested two ensemble models[8]. The Random Forest (RF) is often used for classification tasks as it combines the strengths of multiple individual models (decision trees) to improve the classification and avoids randomness of decision trees. It also reduces over fitting and increases generalization of the classification problem. On the other hand, Extreme Gradient Boosting (XGBoost), an evolution of the RF, enhances performance by sequentially building trees that correct errors from previous one. Both of those ensemble models car capture complex non-linear patterns and lead to more accurate predictions. Finally, in the aim to transform scores into probabilities, I used the Platt-Scaling and Isotonic regression methods, both for RF and XGBoost. Those methods are embedded models. Using a classifier, we can train a calibration model to calibrate its predicted probabilities, such that we can analyse them as probabilities instead of scores (which are more difficult to analyse).

5.1 Random Forest

Introduced by Breiman 2001, Random Forest operates by constructing a multitude of decision trees (Breiman, J. Friedman, and Stone 1984) during training and outputting the class that is the mode of the classes[9] or mean prediction[10] of the individual trees.

When building a decision tree, at each node a sub sample is selected randomly among the entire subset (see Louppe 2014). Therefore, using a decision tree as a predictive model is likely to be non-efficient and under-fit as the parameters will depends on the sub-sample drawn. The RF algorithm was developed as an extension of decision trees. By constructing a multitude of decision trees and averaging the predicted labels (or continuous predictions) avoids the main decision tree issue. This method is known as bagging[11] and makes RF be part of the ensemble models family.

The Random Forest algorithm can be summarized in the following steps:

1. Draw n bootstrap samples from the original dataset.
2. For each bootstrap sample, grow decision tree. At each node, randomly select m features from the total p features.
3. Choose the best feature among the m to split the node.
4. Repeat steps 1, 2 and 3 for a predefined number of trees.
5. Aggregate the predictions of the trees (by majority voting for classification or averaging for regression).

[8] Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone (definition from Wikipedia 2024)

[9] classification

[10] regression

[11] as well as bootstrap sampling

Algorithm 1 Random Forest Algorithm

Input: Training data (x_i, y_i), number of trees N, number of features m
Initialize: $f_0(x)$
for $i = 1$ to N **do**
  Draw a bootstrap sample D_i from the training data
  Grow a decision tree T_i on D_i:
  for each node in T_i **do**
    Randomly select m features from p
    Choose the best feature among the m to split the node
  end for
end for
Output: Aggregate the predictions of N trees

The RF model even if more interpretable than models like XGBoost, still can be seen as a black box[12] when considering the entire forest. However, I decided to use it for its robustness and effectiveness in various predictive tasks.

5.2 Extreme gradient boosting

The Extreme Gradient Boosting, introduced by T. Chen and Guestrin 2016, is a machine learning algorithm popular for its efficiency. It was designed as an extension[13] of the RF such that it can handle large and complicated dataset. In our complex mishandled bags issue, it essential to train and optimise such a model to compare its results to the others (especially to the Logistic regression and the Random Forest).

Unlike RF which is a bagging gradient method, the XGBoost uses gradient boosting approach where trees are built sequentially to correct and adjust the error of previous trees. It is this main important difference that enables XGBoost to achieve more accurate results than RF. The XGBoost algorithm is a pretty efficient and speed one for large dataset, as it utilize advanced optimization techniques such as parallel processing and cache awareness. Moreover, including both L1 (Lasso) and L2 (Ridge) regularization techniques, XGBoost also prevent over-fitting and increase the generalization[14] power.

Following and retracing the literature of the XGBoost 2022, the algorithm can be summarized in the following steps:

1. Initialize the model with a base prediction, usually the mean of the target values.
2. For a predefined number of iterations:
   (a) Compute the gradient of the loss function with respect to the current model's predictions.
   (b) Fit a decision tree to the residuals.
   (c) Update the model by adding the fitted tree, scaled by a learning rate.

[12] Model whose internal logic and decision-making process are not easily interpretable by humans. While the model inputs and outputs are known, the inner logic that transform the inputs onto outputs is opaque.

[13] Gradient boosting is a machine learning technique used for regression and classification tasks. It builds models in a sequential manner, where each new model corrects errors made by the previous ones. By combining these models, it creates a final strong model that improves prediction accuracy. (J. H. Friedman 2001)

[14] Adaptation ability of the model with new and unseen data

3. The final model is the sum of the base prediction and all the fitted trees.

Algorithm 2 XGBoost Algorithm

Input: Training data (x_i, y_i), number of iterations M, learning rate η
Initialize: $f_0(x) = \log(\frac{\bar{y}}{1-\bar{y}})$
for $m = 1$ to M **do**
  Compute the residuals: $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$
  Fit a decision tree $h_m(x)$ to the residuals r_{im}
  Update the model: $f_m(x) = f_{m-1}(x) + \eta h_m(x)$
end for
Output: Final model $f_M(x) = f_0(x) + \sum_{m=1}^{M} \eta h_m(x)$

Diving into the maths, XGBoost algorithm optimizes the following objective function:

$$\mathcal{L}(f) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{m=1}^{M} \Omega(h_m) \tag{8}$$

where:

- $L(y_i, \hat{y}_i)$ is the logistic loss for classification.
- $\Omega(h_m)$ is the regularization term for the m-th tree, defined as:

$$\Omega(h_m) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} ||w_j||^2 \tag{9}$$

  where we have two regularization parameters. γ controls the number of leaves T and w_j, the weight of leaf j, is penalized by λ when it turns too heavy. Both parameters are extremely precious in the aim of generalization.

The gradient boosting framework updates the model by adding the tree that minimizes the following objective:

$$\mathcal{L}^{(m)} = \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(m-1)}) + g_i h_m(x_i) + \frac{1}{2} h_i h_m(x_i)^2 \right] + \Omega(h_m) \tag{10}$$

where g_i and h_i are the first and second-order gradients of the loss function with respect to the predictions:

$$g_i = \frac{\partial L(y_i, \hat{y}_i^{(m-1)})}{\partial \hat{y}_i^{(m-1)}}, \quad h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(m-1)})}{\partial \hat{y}_i^{(m-1)2}} \tag{11}$$

The tree is then fitted to the negative gradients, and the weights are adjusted to minimize the objective function, incorporating both the loss and the regularization term.

As the XGBoost model is very complex, it is considered a black box in which it is virtually impossible to interpret the coefficients. However, we'll be using it for its accuracy, with the aim of predicting missed baggage as accurately as possible.

5.3 Probability calibration

Classifier models used classify data according to predicted scores. However, these scores cannot be interpreted as probabilities. Indeed, when a model learns and predicts unbalanced data, it is likely to underestimate the probability of events in the minority class. In our case, models are likely to assign too low a probability to missed baggage[15]. For this reason, classifier scores aren't considered as predicted probabilities but scores.

First introduced by Zadrovzny and Elkan 2002 (and pretty well explained by Niculescu-Mizil and Caruana 2005[16]) Platt Scaling and Isotonic regression are both most common method of probability calibrations.

5.3.1 Platt Scaling

Platt scaling involves fitting a logistic regression model to the classifier scores. The logistic regression model is trained using the classifier scores as input and the true binary labels as the target. In the case of baggage issue, the classifier score are the predicted probability that a baggage is mishandle, and the true binary label is the actual final status of the baggage, being successful or mishandled. The formula for platt scaling is given by :

$$P(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)} \tag{12}$$

where $f(x)$ is the predicted probability of the classifier for input x, and A and B are parameters learned from the training data. Both parameters are estimated using maximum likelihood method. And gradient descent is used to solve:

$$\arg\min_{A,B} \left\{ -\sum_i y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right\}, \tag{13}$$

where,

$$p_i = \frac{1}{1 + \exp(Af_i + B)} \tag{14}$$

5.3.2 Isotonic Regression

The isotonic regression is a non-parametric method where the only restriction is that the estimated function should be monotonically increasing (isotonic). The isotonic regression can be mathematically written as :

$$y_i = m(f_i) + \epsilon_i \tag{15}$$

where f_i is the classifier score and y_i the true target. Then, we find the isotonic function $\hat{m}$ such that :

$$\hat{m} = \arg\min_z \sum_i (y_i - z(f_i))^2 \tag{16}$$

For both method, using the same training set for the first classifier and the calibration would lead to over-fit. Thus, we need to have a second hand calibration dataset used to train the platt scaling and isotonic regression after having trained the classifier with the main train dataset.

[15] A phenomenon also known as distortion

[16] Articles Brownlee 2020 and DataLab 2022 were also used to understand these concepts

6 Methodology

All the machine learning models used were trained using the same methodology, otherwise their results would not be comparable. This section details the methodology used to train and test the algorithms.

Data normalization Continuous data normalization is a crucial step in the modeling process. It consists in transforming the scale of the data so that all data are comparable. Without this transformation, algorithms could over-estimate the importance of a feature due to its much larger scale than another feature. Under these conditions, data normalization increases the predictive performance of models, or in other words, reduces its degradation. As my data contains outliers (which I haven't removed), Robust Scaler is particularly effective as this method is based on the median and interquartile range.[17], which makes it less sensitive to outliers[18]. This allows the data to be centered and reduced so that extreme values do not disproportionately influence the results, providing a more reliable representation of data characteristics for machine learning algorithms. Bellow is the function used for the robust scaling.

$$x_i = \frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{17}$$

where Q_1 is the 1st quartile, and Q_3 is the third quartile.

Data splitting In most training cases, the data is split between a train set and a test set. In my case, I split my data between a train, a test and calibration datasets. The calibration dataset was used separately to calibrate the data, using the platt scaling and isotonic regression methods (explained in subsection 5.3). The Figure 14 shows how data has been split in proportion.

Figure 14: Data split

This split method was applied for training, calibration and testing by setting the random state. In this way, all algorithms had the same data for each modeling step. Finally, each dataset contained the same proportion of failed baggage. By using the function *stratify* of sklearn[19], we can make sure that the minority class will always have the same proportion. If we don't do this, we could end up with, for example, too many failed test sets and too many failed training sets. Under these conditions, our test results would be biased.

[17] Library : ScikitLearn 2007 - 2024a

[18] In the contrary, Standard Scaler method is based on the mean and standard deviation. As a result, having outliers can bias the scaling and make it less efficient.

[19] Library : ScikitLearn 2007 - 2024b

Cross validation Cross-validation is an essential technique in training, as it enables the performance of a model to be assessed more robustly and reliably. Unlike a simple separation of data into training and test sets, cross-validation divides the data into several subsets or "folds", then trains and tests the model on different combinations of these subsets. This reduces the risk of bias due to a specific partition of the data, and provides a more accurate estimate of the model's ability to generalize to unseen data. In short, cross-validation helps to avoid over-fitting and ensures that the model is well adapted to predict varied data, making its predictions more reliable in production. Cross-validation is used with the train dataset, which is split into four folds.

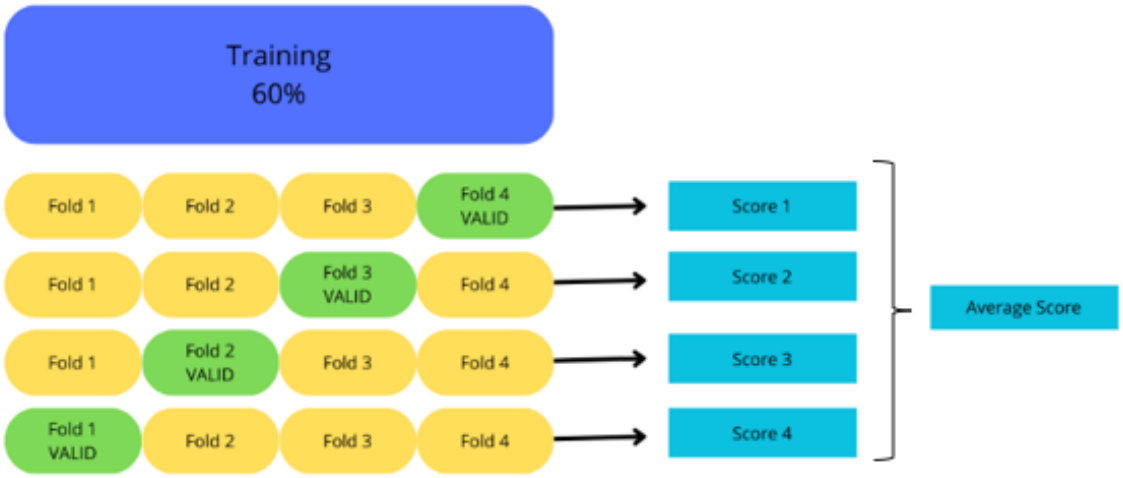


Figure 15: KFold cross-validation

The Figure 15 shows how the training set has been split for cross-validation. We have four different folds. The models have therefore been trained separately but with the same parameters four times and tested on four different dataframes. Their four scores are then calculated and aggregated into an average, which is the final model score. By using this method, we avoid a model performing better randomly depending on the training or test dataset.

Hyper parameter optimization Machine learning algorithms can contain a large number of parameters, specifying how the learning tasks should process (such as the number or depth of trees). Those are called hate, hyper-parameters. A way to optimize an algorithm is to optimize those parameters. For this purpose, there exists two popular methods. The grid search consists in setting a couple of possible values for each parameters, train the algorithm for each possible combination of them, and keep the combination with the highest score. The random search consists in setting a certain distribution function for each parameters, and draw each combination randomly. With such a method, we can find certain region of parameters where they are more or less efficient.
I firstly used the grid search method to find a first set of parameters, and then tried to improve the score with a random search.

The Table 4 display the final sets of parameters for both RF and XGBoost. Those were find with the random search method.

Table 4: Final sets of hyper-parameters

Parameter	**Model**	
	RF	**XGBoost**
criterion	Entropy	
min samples leaf	1	
max depth	35	31
nb. estimators	278	188
column sample by tree		0.63
eta		0.06
reg lambda		0.88
sub sample		0.87
min child weight		1

Concept of class weights Class weights are coefficients assigned to classes in order to handle imbalanced datasets. When data is imbalanced, the class with the majority of instances can dominate the learning process, causing the model to perform poorly on the minority class. By assigning higher weights to minority classes and lower weights to majority classes, the model can be made more sensitive to the minority class, improving overall performance. For imbalanced datasets, class weights adjust the learning algorithm to pay more attention and give more power to the minority class, helping to balance the model's performance across all classes.

In a Logisic regression, class weights are incorporated into the loss function (see Müller 2020). The weighted loss function is then given by :

$$L(\theta) = -\sum_{i=1}^{n} w_i \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right] \tag{18}$$

where w_i are the class weights, y_i are the true labels, and p_i are the predicted probabilities.

In Random Forest, class weights can be applied to the criterion used to split the nodes. The Gini impurity or entropy can be weighted by the class weights. For instance, the weighted Gini impurity is:

$$G_w = \sum_{k=1}^{K} w_k p_k (1 - p_k) \tag{19}$$

where w_k is the weight for class k, and p_k is the proportion of class k at a given node.

For XGBoost, class weights can be used to scale the gradients and Hessians during the boosting process. The weight-adjusted loss function for XGBoost algorithm can be represented as:

$$L(\theta) = \sum_{i=1}^{n} w_i \ell(y_i, f(x_i)) \tag{20}$$

where ℓ is the loss function, w_i are the class weights, y_i are the true labels, and $f(x_i)$ are the model predictions.

Classification threshold choice In our binary classification models, the aim is to predict whether a bag will be mishandled. We therefore have a binary output. Instead of directly giving a class as output, the model produces scores. These scores indicate the model's confidence in the bag being successful or mishandled.

For example, if the model gives a score of 0.7 and classifies the bag as negative, we can say that it predicts that there is a greater chance that the bag will fail than succeed. By default, the models used classify baggage as mishandled if the score is greater than 0.5 and as successful if the score is less than 0.5. However, this arbitrary classification is not always the one that optimizes the score, especially when dealing with unbalanced data. To improve classification, a different threshold can be defined. Optimizing the F1-score involves finding the threshold that maximizes this metric. Thus, I have calculated the F1-Score for all existing thresholds from 0 to 1. Finally, the classification threshold kept is the one for which the F1-Score is the highest.

Providing actual probabilities As explained in the subsection 5.3, predicted score are not always probabilities. In order to give probabilities instead of predicted scores, we can perform probability calibration. However, this step must be carried out rigorously to avoid over fitting.
Probability calibration isn't done with training data. Firstly, we need to train the models with the train data. Then, the predicted scores are given to the probability calibration models (platt scaling and isotonic regression) to train them.[20]. The Figure 16 gives a overview of the entire set-up methodology.

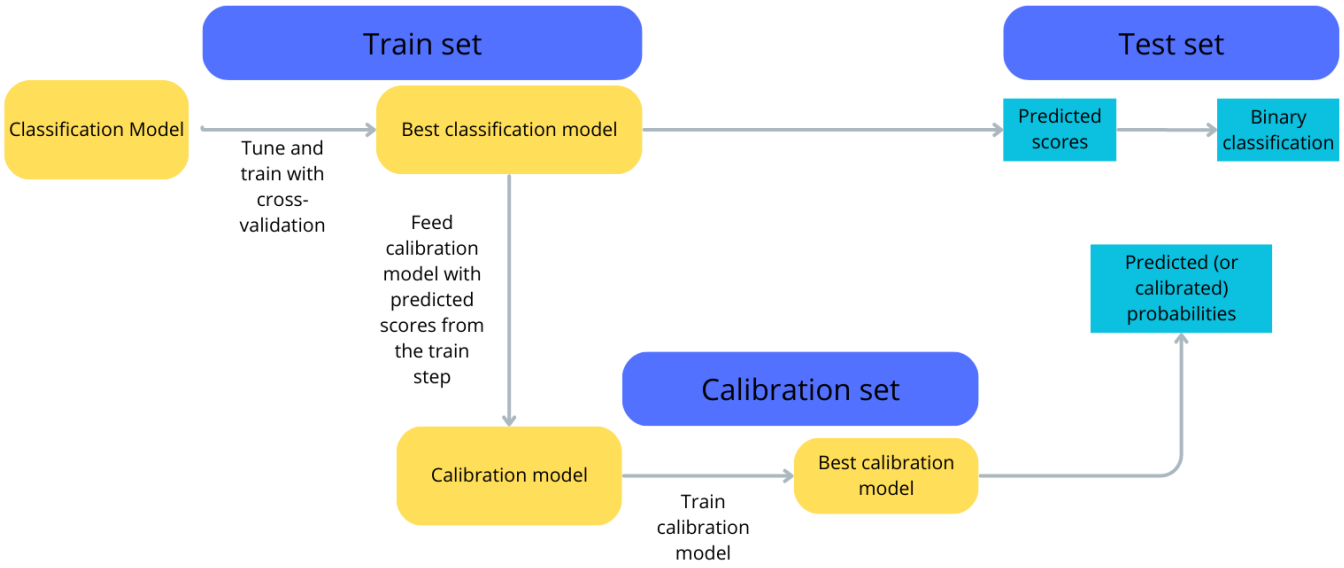


Figure 16: Training, calibration and testing methodology

[20]This type of method is known as an "embedded model": a model that contains at least one variable predicted by another model.

7 Results

The Table 5 lists the results of the models. For each of them, the F1-Score, recall and precision have been calculated with their optimal classification threshold.

Classification model Model 1, the logistic regression used as a benchmark, is the model with the lowest classification performance. This shows that the classification is quite complex and that it is therefore useful to use more advanced machine learning models.
Both ensemble models without path duration are quite efficient with a recall of 0,74 (model 2) and 0,73 (model 3), which means they didn't predict much mishandled bags as successful and F1-Score and Precision between 0,7 and 0,75. Moreover, we can see that adding path duration (with residuals) allows models to improve their metrics by about 0,1 point. Optimization by random search of hyper-parameters, on the other hand, managed to improve the F1-score of the RF by 0,03 points and of the XGBoost by 0,01, which is very low.
Both probability calibration methods do not improve model scores. This was to be expected as because the aim of probability calibration is to transform predicted scores into probabilities rather than to better classify the data.

The Figure 17 shows the distribution of predicted probabilities for logistic regression, which is the least efficient model. On the other hand, the Figure 18 shows the predicted probability distribution for model 10. This is the model I consider to be the most successful and that I recommend for use in the section 7. These graphs have been made with sample data, to see how the predicted probabilities are distributed, and where the thresholds are located. We can see in blue the successful bags and in red the mishandlded luggage. On the ordinate, we have the probabilities predicted by the model. The higher the dots, the higher the predicted probability of a mishandled. Ideally, blue dots should be very low, red dots very high. As can be seen, the logistic regression does not classify failed and successful bags well. However, our best model shows a much better separated distribution between mishandled and successful baggage. The predicted probability distributions of the other models are all available in the appendix

To better analyze model results, we can also analyze confusion matrices. The Figure 19 and Figure 20 shows the confusion matrix of the benchmark model and the more efficient one (RF with isotonic regression). Once again, we can see that logistic regression doesn't classify mishanlded luggage very well. In fact, for every bag it predicts to be mishanlded, 1,756 actually fail, while 2,918 succeed. This represents too much confusion. With such a model, too many bags that should have been successful could be predicted as misses. However, for model 10, for all the baggage it predicts to mishandled, 2,616 actually miss, while only 223 successful. Clearly, our best-performing model is able to classify mishandled baggage much better than the reference model, which classifies with too many errors. In the appendix, Figure 43 to Figure 53 shows the density of predicted mishandled bags versus the density of predicted successful bags. These plots have redundant information with predicted probability distribution, but show the estimated density of all points (and not only samples).

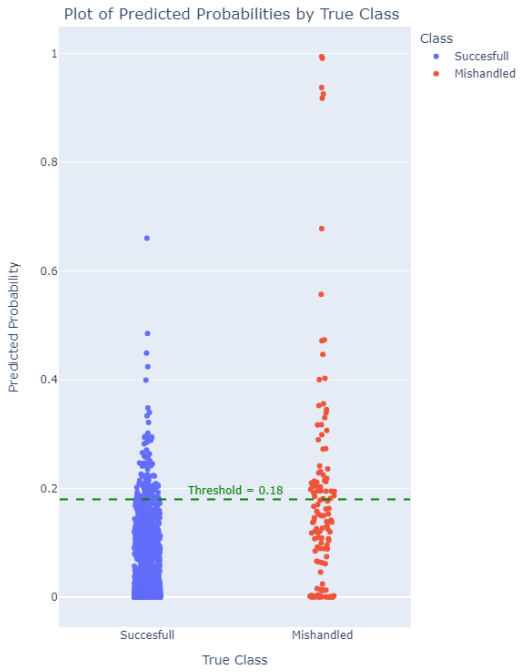


Figure 17: Probability distribution - Model 1

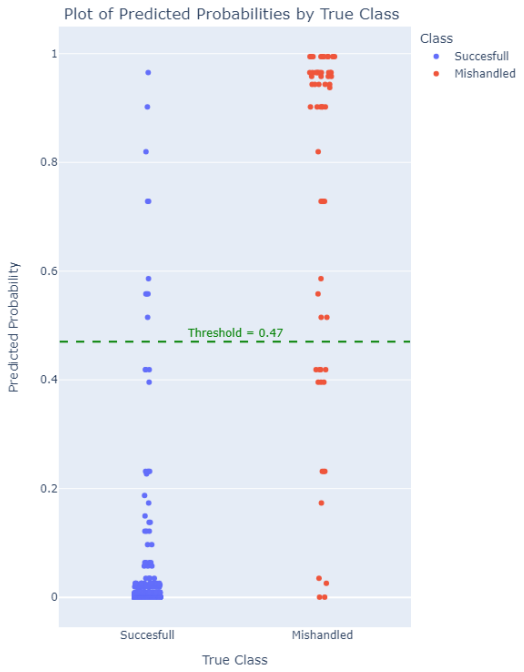


Figure 18: Probability distribution - Model 10

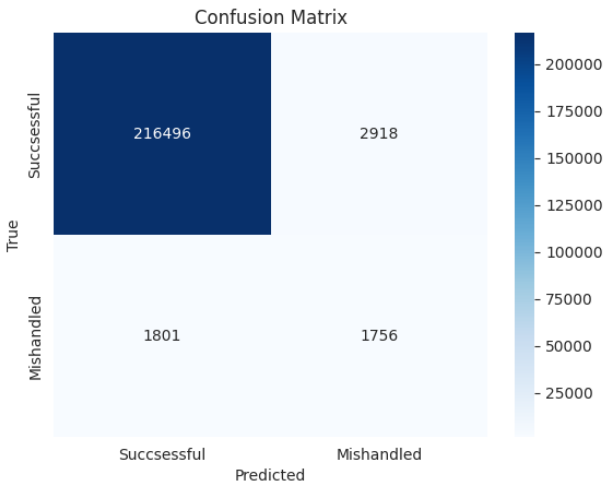


Figure 19: Confusion matrix - Model 1

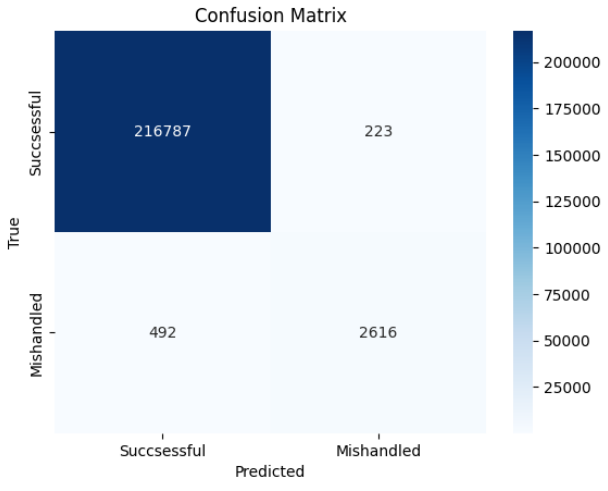


Figure 20: Confusion matrix - Model 10

Probability calibration We can see that the classification thresholds change after calibration, due to the fact that scores are transformed into probabilities, and the two are not equally distributed. To analyse and define the best probability calibration, we need to have a look at Figure 21 and Figure 22. These two graphs compare the distribution of predicted scores (or probabilities) with the empirical proportions of mishandled baggage. Perfectly calibrated probabilities should follow the diagonal of the graph. In this case, we could say that on the bunch of bags where the predicted probability of mishandling is 70%, the same proportion are actually mishandled. This would actually correspond to a probability, not a score. We can see that for XGBoost (blue line in the Figure 21), the model tends to over-predict baggage with a score of 0,8. However, when the probabilities are calibrated by isotonic regression, they are under-predicted. Finally, platt-scaling does not improve but degrades score quality. The Figure 22 shows that the probability calibration performed better for the RF than the XGBoost. Indeed, the RF without calibration tends to under-estimate scores between 0,4 and 0,95. However, the isotonic regression fits almost perfectly with the diagonal. With such a calibration, we can conclude that the RF calibrated with an isotonic regression is the more accurate embedded model to calibrate and provide interpretable probabilities.

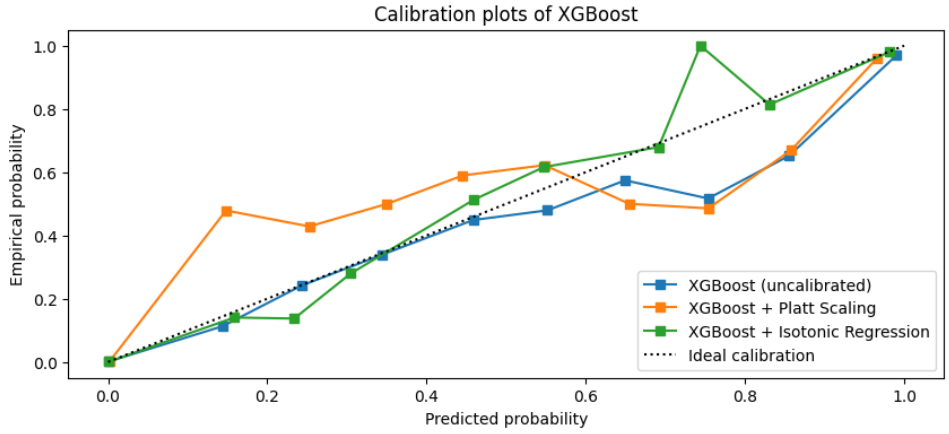


Figure 21: Probability calibration for XGBoost

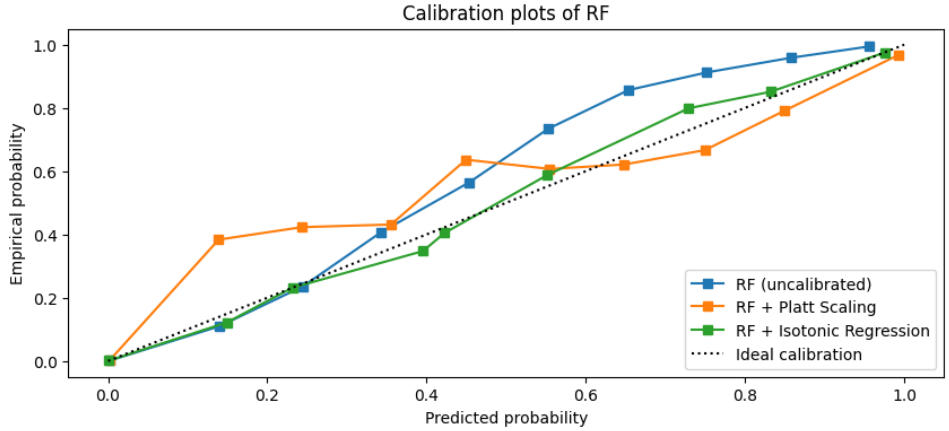


Figure 22: Probability calibration for Random Forest (RF)

Feature importance in predictions Having predicted classifications and probabilities is pretty important and a good step forward in the aim to reduce mishandled baggage. However, under-

standing how models make their predictions could give us more insight to understand the reason baggage can be mishandled. As they are considered as black boxes, machine learning models are not easily interpretable. However, computing their shapley values (see Molnar 2024) allows to quantify the impact of features on model predictions. The Figure 23 shows the shap values of the model 10. We can see that the most important variable is the number of minute before flight departure (*input_ICT_BHS*). All of the red dots are on the left side of the figure, which can be interpreted as : when the baggage enters onto the BHS with a large number of minute before flight departure, it is estimated more likely to be mishandled. In the other hand, if a baggage enters onto the BHS while its flight departure is planned in 5 minutes, it is more likely to be mishandled. This make sense and doesn't give much information to the handlers. However, this data is the most important in prediction.

The predicted path duration (or path duration with noise added) is the second most important data. This one is pretty important because if we actually manage to do an embedded model and predict path duration before the bag enters in the sorter, this shap value could warn handlers that there might have a problem in the sorter such that the path duration will be higher than planned.

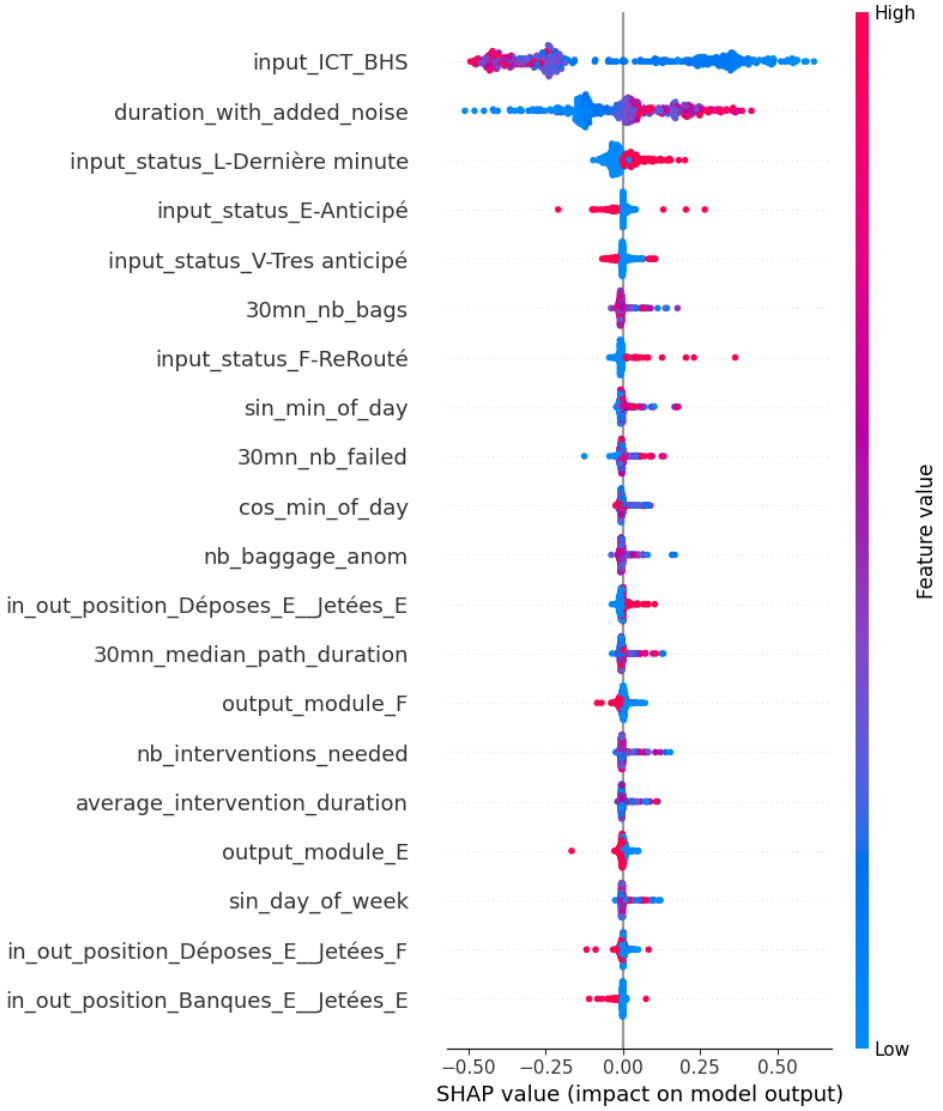


Figure 23: Shap values of the model 10 (RF + Isotonic regression)

Moreover, this information can also be used in post operation analysis. If there was a day with a lot of mishandled baggage, ADP could use this model as to compute shap values and check out why and quantify the cause of mishandling. This could help in the allocation of responsibility between ADP and Air France[21] for mishandled baggage. If a baggage is predicted mishandled due to a longer path duration predicted, this would mean ADP is responsible of the mishandling. On the other hand, if a baggage is missed because it has a short amount of time before flight departure, it means Air France injected it too late and they might be responsible. About the other features, we see that they all have a lower impact. Still, those impact are not insignificant.

Finally, I computed the individual shap values for three mishandled baggage. The Figure 56 is pretty interesting, as wee see that he model predicted the baggage to be mishandled, not only because of its path duration or the number of minute before flight departure. The probability of being mishandled increased thanks to a lot of variables. In this case, it would be interesting to analyse the baggage path and see what was the reason of its mishandling.

The Figure 57 shows that the model predicted a successful baggage aleven thought it was mishandled. The number of minute before flight departure have reduced the predicted probability to be mishandled, this most likely means that the baggage wasn't injected onto the BHS too late. In contrast, the predicted path duration slightly increased the predicted probability. According to those values, we could conclude that something went wrong during the journey in the BHS.

Finally, the Figure 58 shows that the model clearly predicted the baggage to be mishandled because it had not enough time before flight departure. With such values, we could conclude that it arrived too late and the BHS did not mishandled it.

Conclusion

Based on the results, I would suggest to use the model 10 (RF calibrated by isotonic regression), for both classifying and predicting probabilities. Computing the shap values in real time could also help handlers to take decisions. Moreover, those values can also be used in post operations analysis to help the analyst to understand the reasons of a mishandling.

Furthermore, all of my models were trained and tested for three months. Adding the variable "*date of month*" and training the model on a longer period could make it learn more seasonal variations and cyclical patterns.

Testing the models under conditions of high logistical disruption (or high rate of mishandled bags) could be a good experimentation. This could provide rigorous stress-test, allowing to observe how the model behaves under complicated circumstances. If the model performance remains stable, it would mean it is confident even when there are a lot of perturbations in the logistic process.

Further investigations should be made on the use of path duration. If a predictive model on this variable is finished, I would suggest to compare features used in both models and to make sure they don't have the same. If embedding of both models ends-up to be impossible, the models 2 and 3 (models not using predicted path duration) could be calibrated. They have a lower performance as they don't use the path duration, but still can be improved and used.

The use of over or under sampling could be an idea to increase the models performances. However, I didn't use those methods as my scores were higher than expected without sampling. By using such methods, we would be more likely to over fit our models, so they should be applied rigorously.

[21] As Air France is in charge of baggage logistic outside the BHS

Appendix

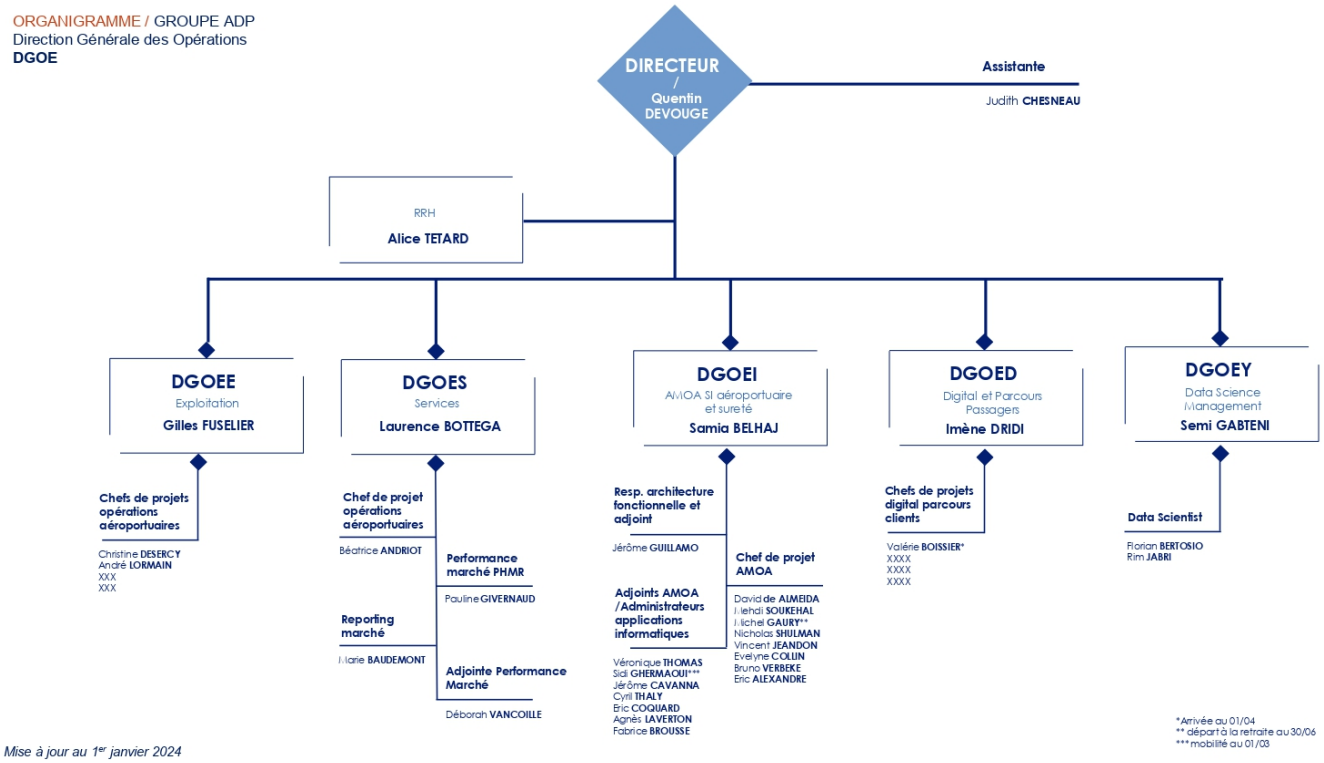


Figure 24: DGO organization chart

Table 5: List of models

ID	Name	Description	Threshold	F1-Score	Recall	Precision
1	Logistic regression	Main features	0.18	0.43	0.49	0.38
2	RF	Main features	0.33	0.76	0.74	0.78
3	XGBoost	Main features	0.7	0.74	0.73	0.75
4	RF	Path duration with noise added	0.34	0.85	0.85	0.9
5	XGBoost	Path duration with noise added	0.61	0.88	0.87	0.9
6	RF	Random search optimization	0.37	0.88	0.85	0.9
7	XGBoost	Random search optimization	0.45	0.89	0.87	0.9
8	RF + Platt scaling	Platt scaling for probability calibration	0.39	0.88	0.84	0.93
9	XGBoost + Platt scaling	Platt scaling for probability calibration	0.13	0.88	0.87	0.9
10	RF + Isotonic regression	Isotonic regression for probability calibration	0.47	0.88	0.84	0.92
11	XGBoost + Isotonic regression	Isotonic regression for probability calibration	0.35	0.89	0.88	0.9

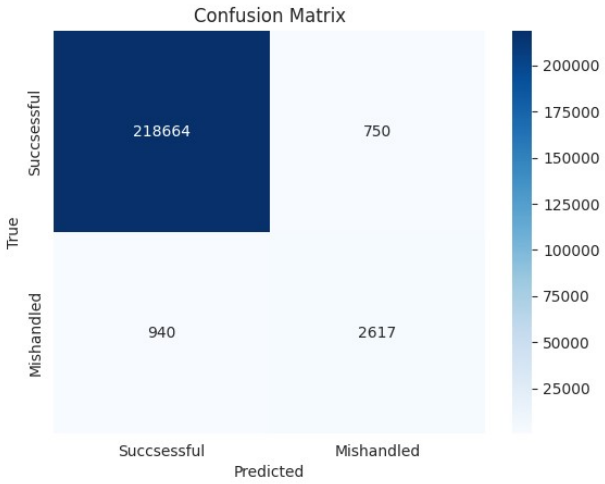


Figure 25: Confusion matrix - Model 2

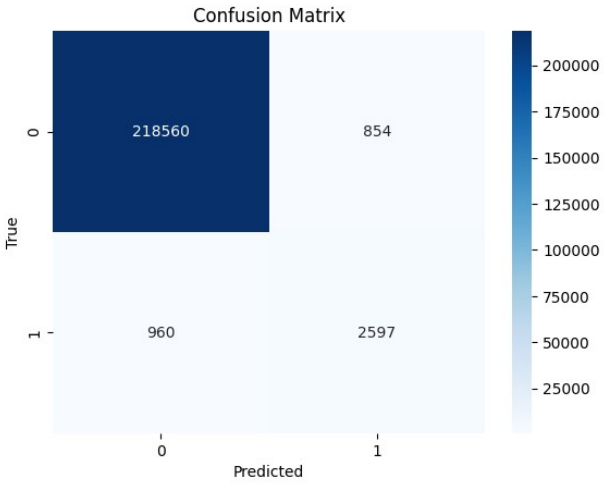


Figure 26: Confusion matrix - Model 3

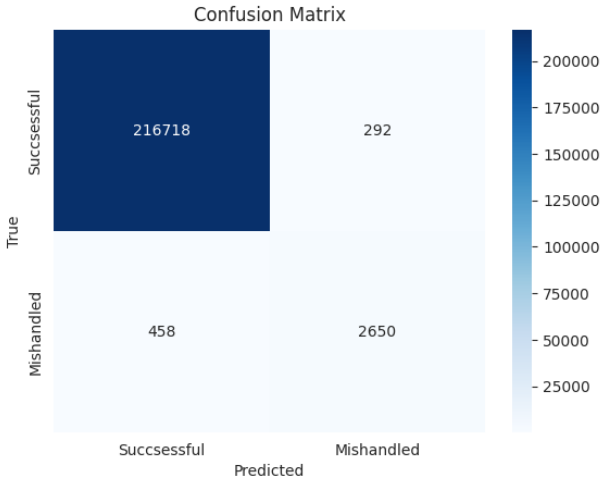


Figure 27: Confusion matrix - Model 4

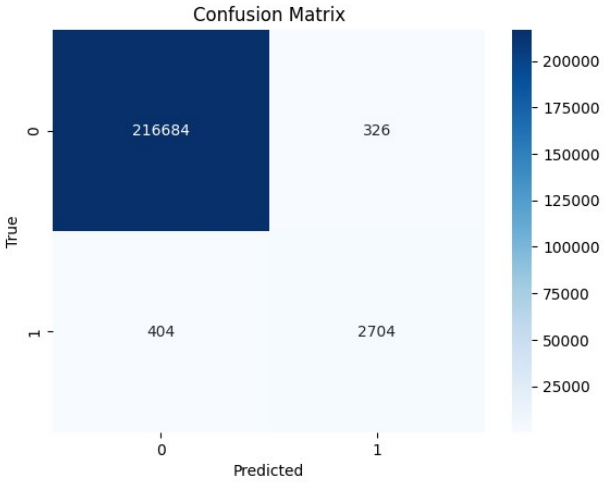


Figure 28: Confusion matrix - Model 5

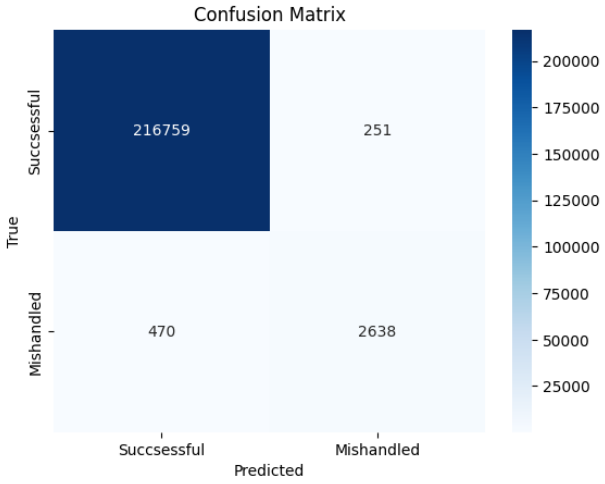


Figure 29: Confusion matrix - Model 6

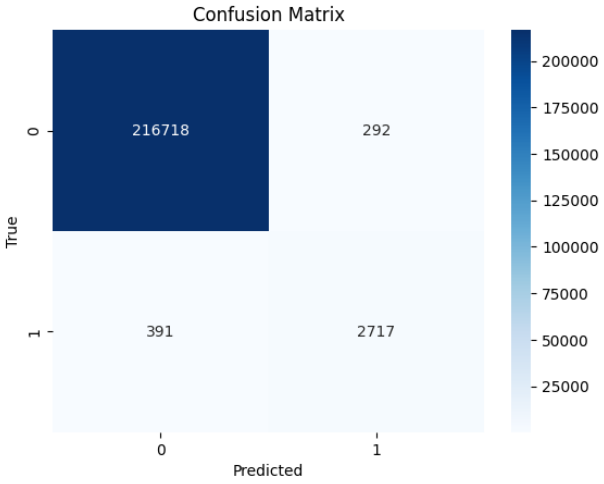


Figure 30: Confusion matrix - Model 7

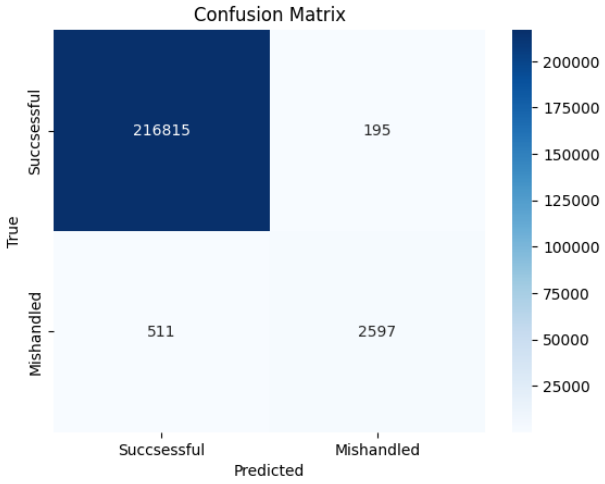


Figure 31: Confusion matrix - Model 8

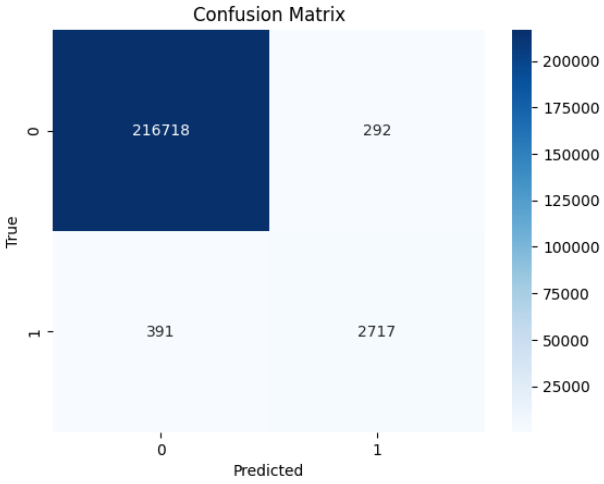


Figure 32: Confusion matrix - Model 9

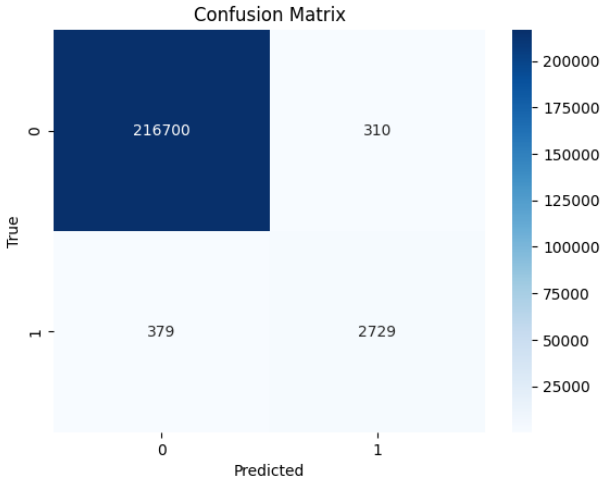


Figure 33: Confusion matrix - Model 11

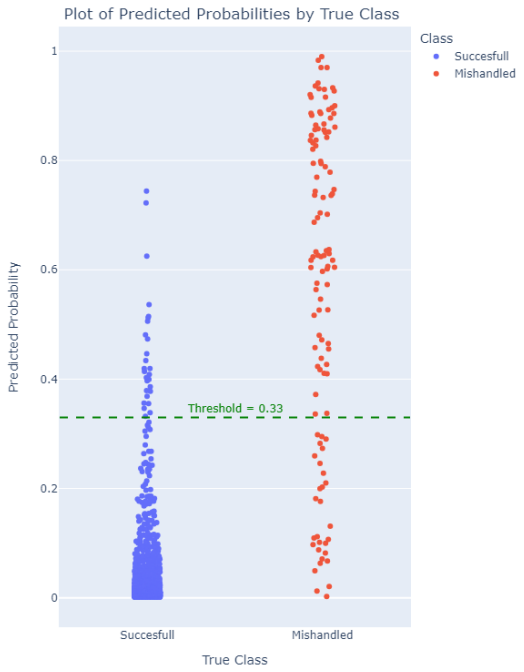


Figure 34: Probability distribution - Model 2

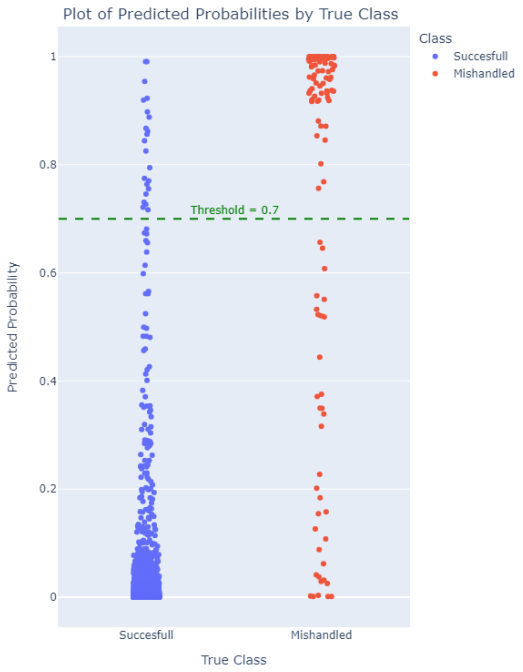


Figure 35: Probability distribution - Model 3

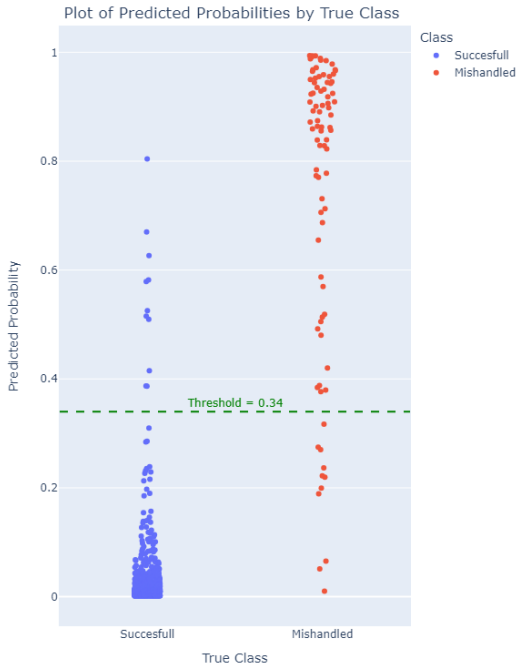


Figure 36: Probability distribution - Model 4

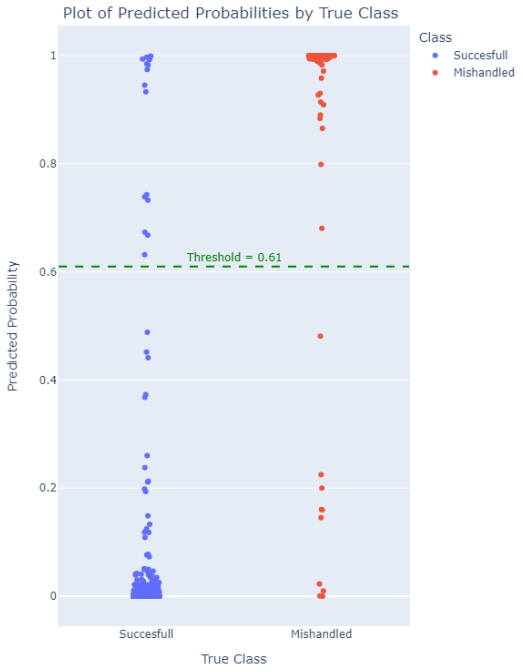


Figure 37: Probability distribution - Model 5

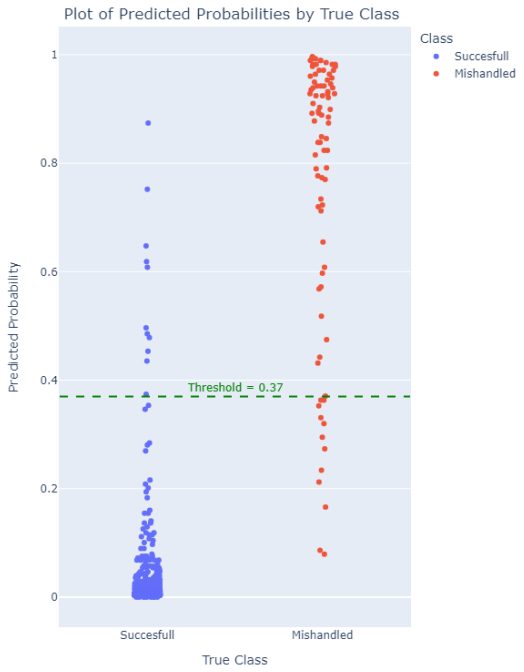


Figure 38: Probability distribution - Model 6

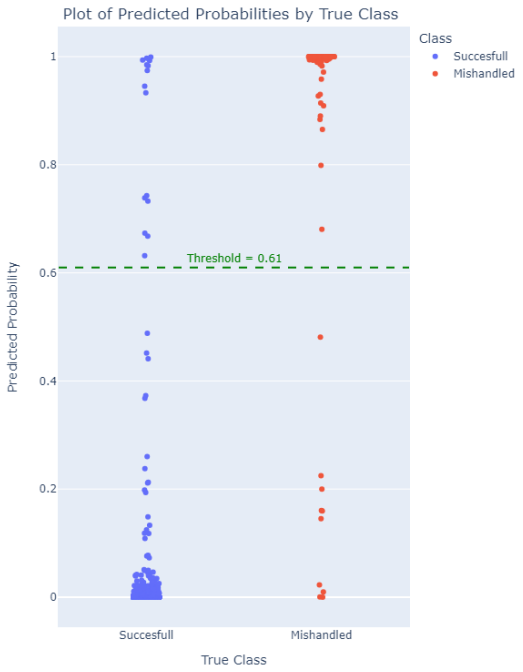


Figure 39: Probability distribution - Model 7

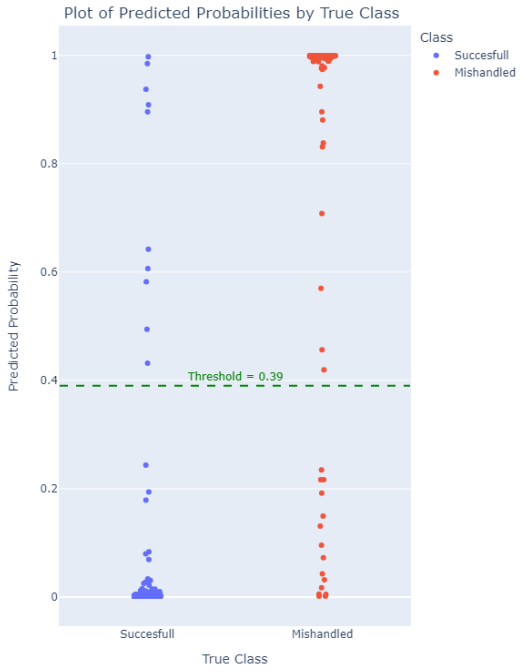


Figure 40: Probability distribution - Model 8

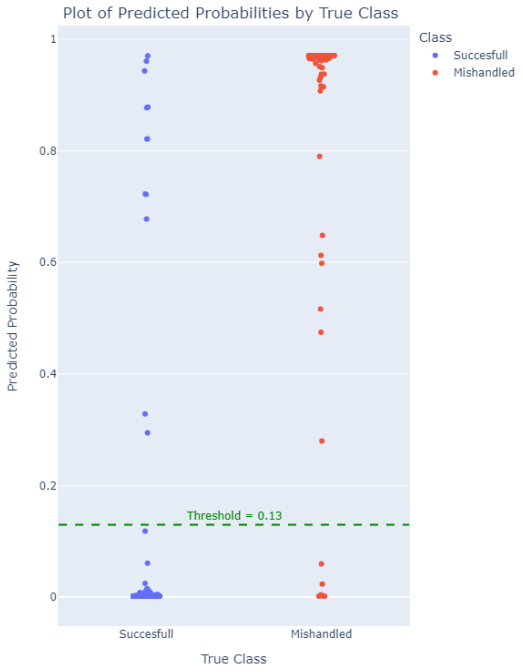


Figure 41: Probability distribution - Model 9

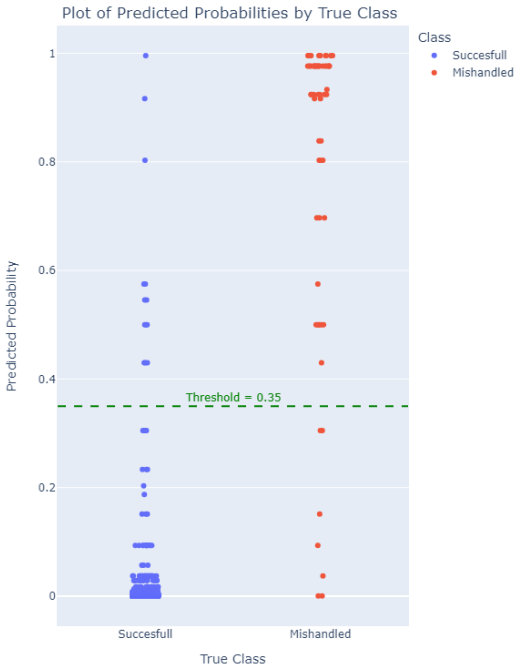


Figure 42: Probability distribution - Model 11

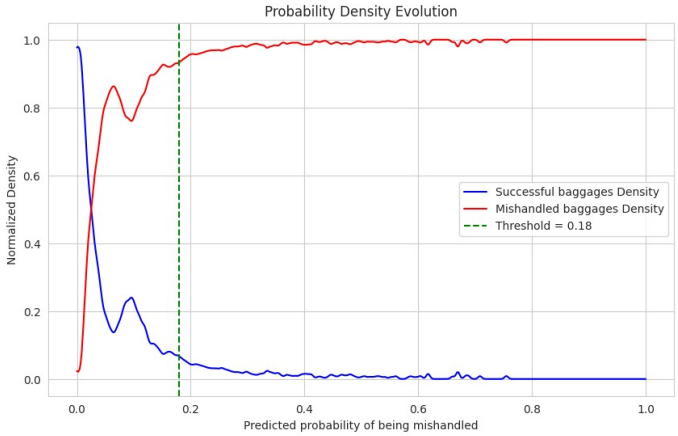


Figure 43: Probability density - Model 1

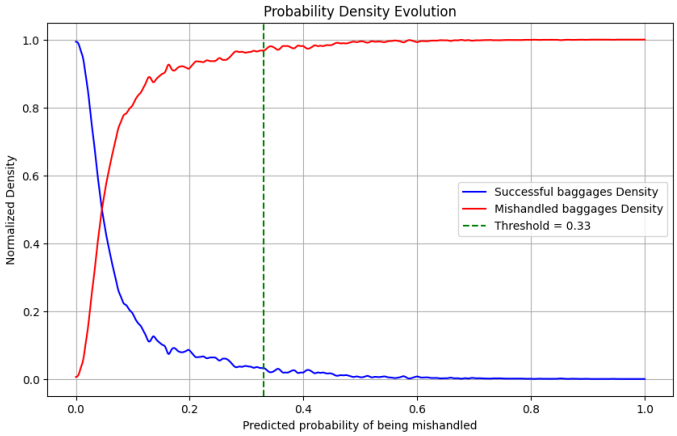


Figure 44: Probability density - Model 2

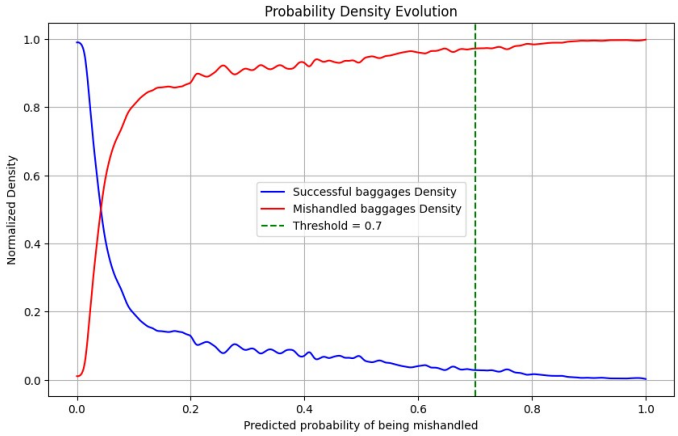


Figure 45: Probability density - Model 3

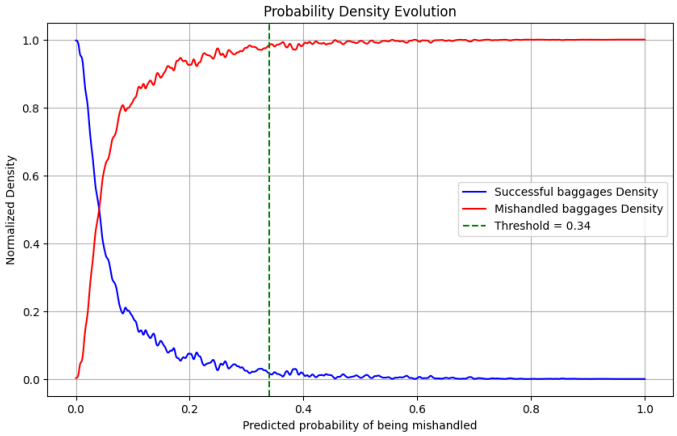


Figure 46: Probability density - Model 4

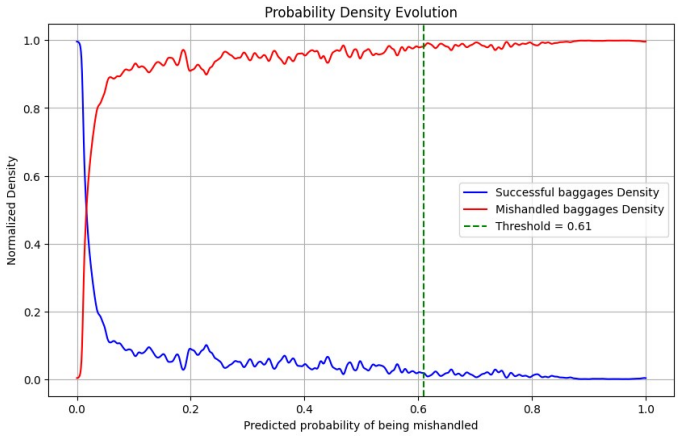


Figure 47: Probability density - Model 5

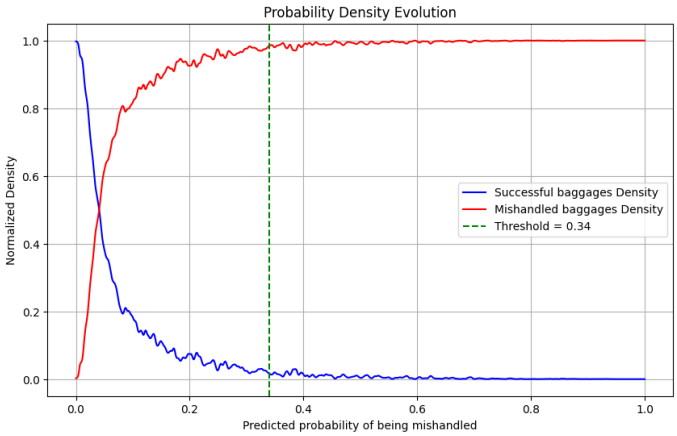


Figure 48: Probability density - Model 6

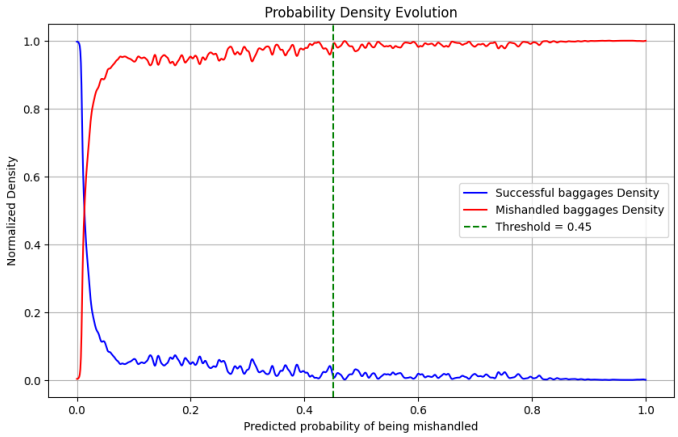


Figure 49: Probability density - Model 7

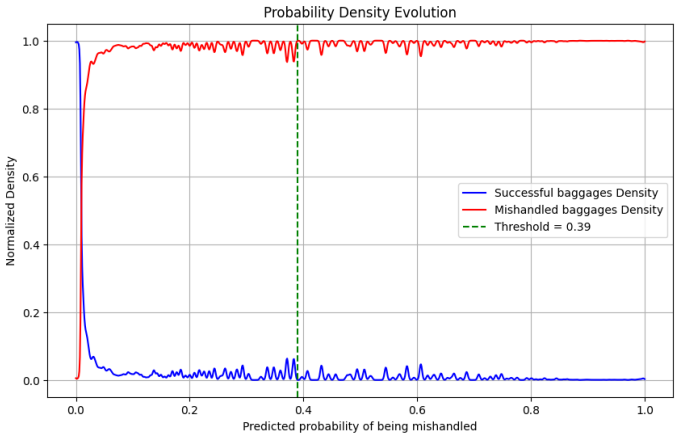


Figure 50: Probability density - Model 8

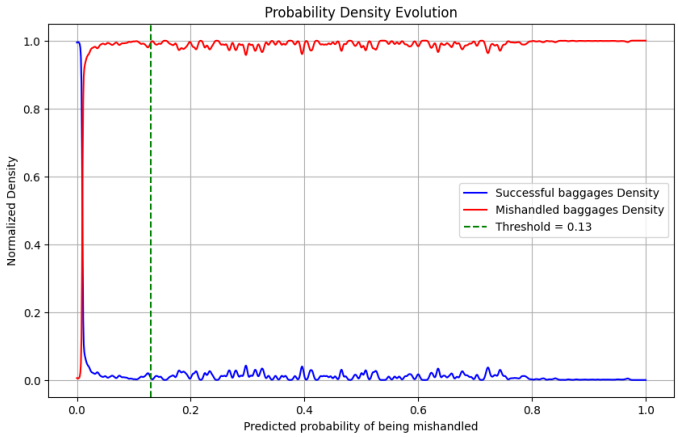


Figure 51: Probability density - Model 9

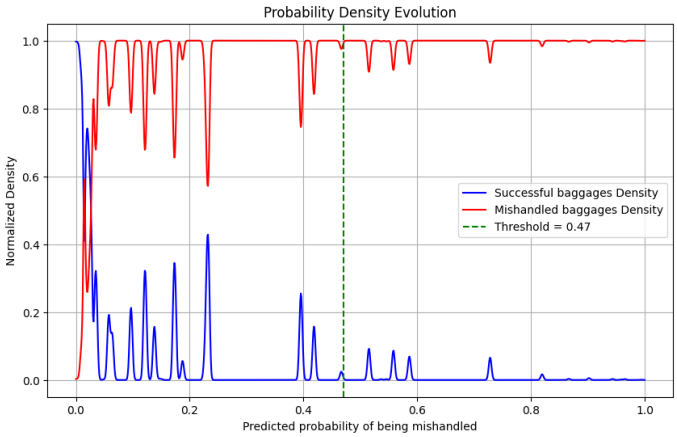


Figure 52: Probability density - Model 10

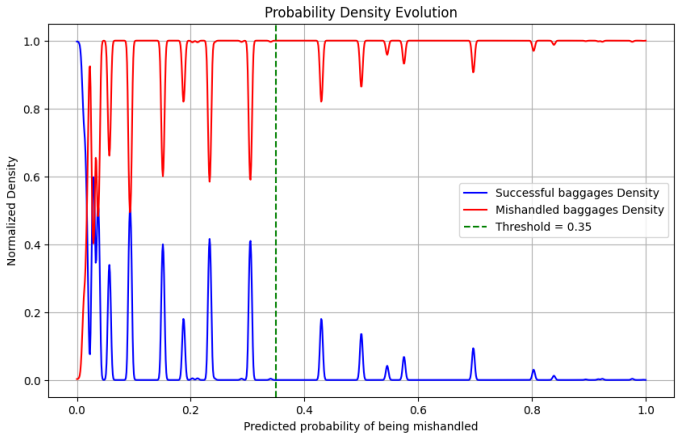


Figure 53: Probability density - Model 11

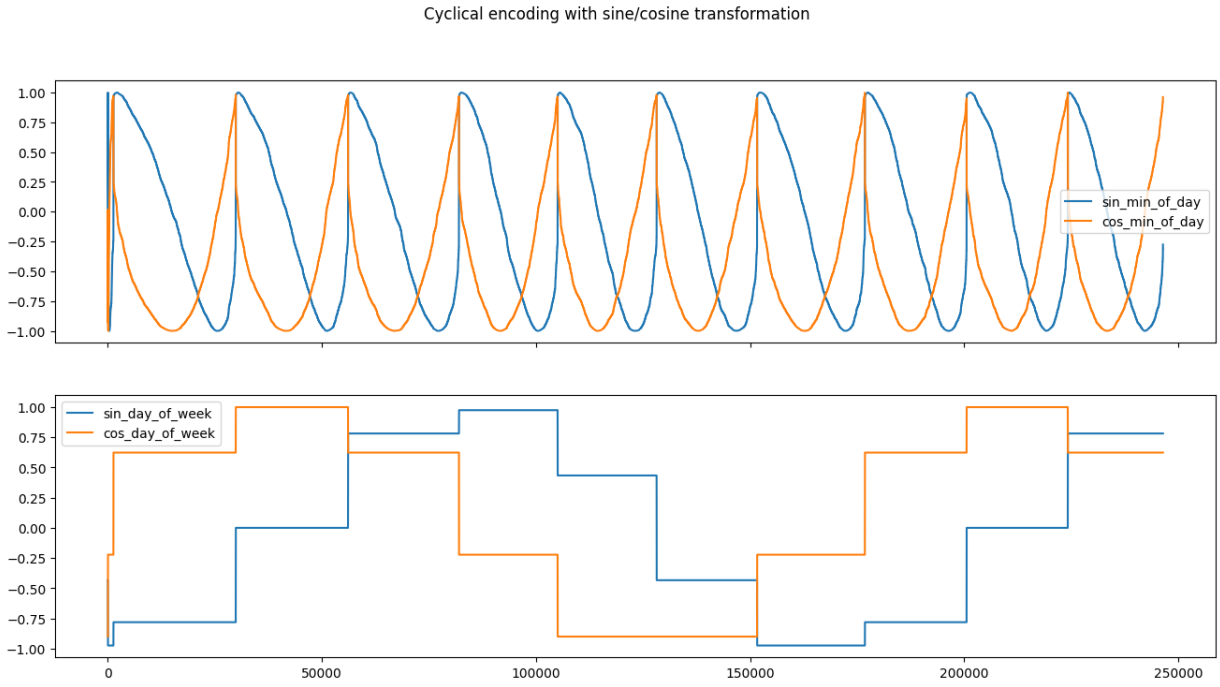
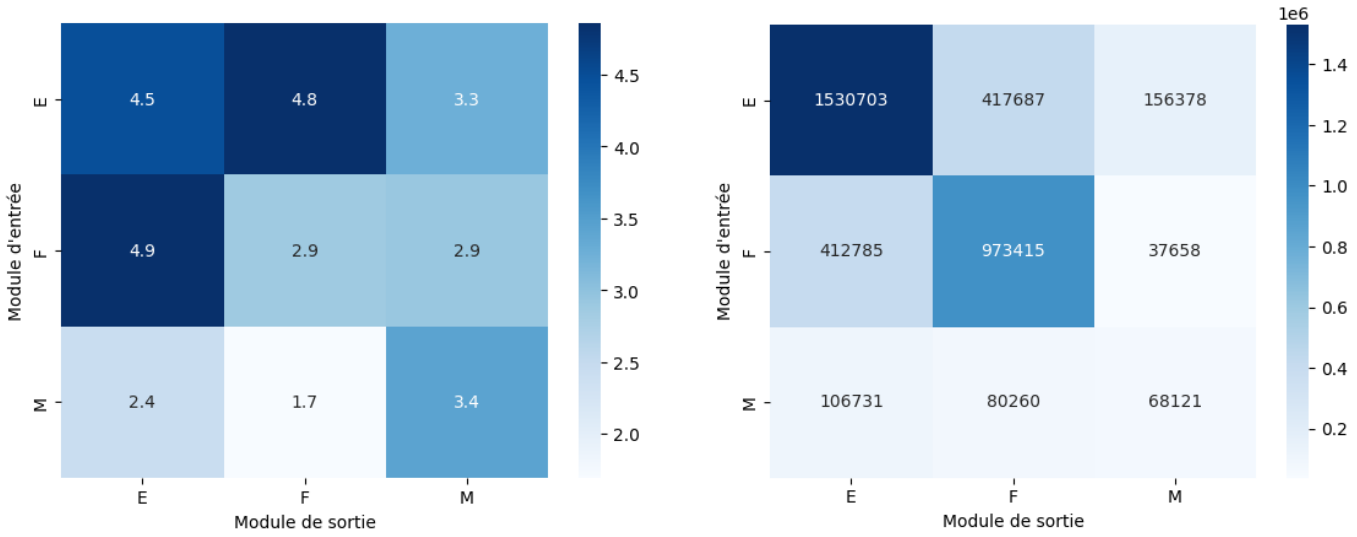


Figure 54: Cyclical encoding of days and minutes



(a) Percentage of mishandled bags according to input and output modules

(b) Number of bags according to input and output modules

Figure 55: Mishandled bags according to modules

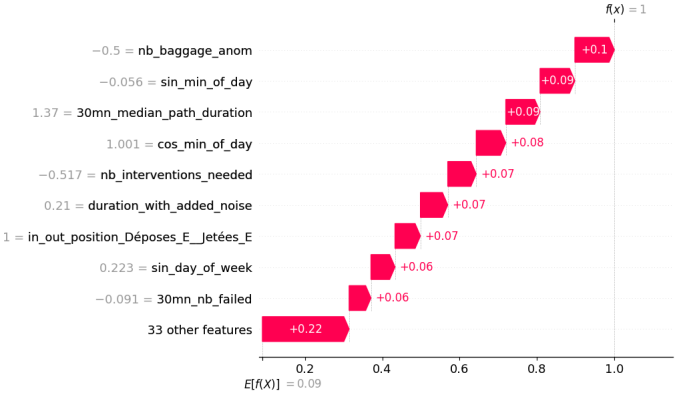


Figure 56: Shap values of a random mishandled baggage (1)

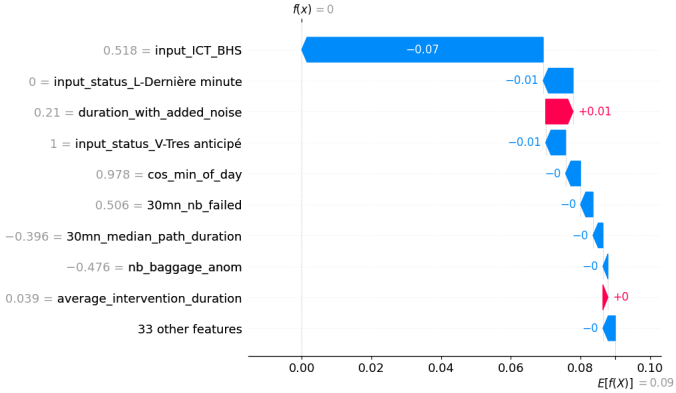


Figure 57: Shap values of a random mishandled baggage (2)

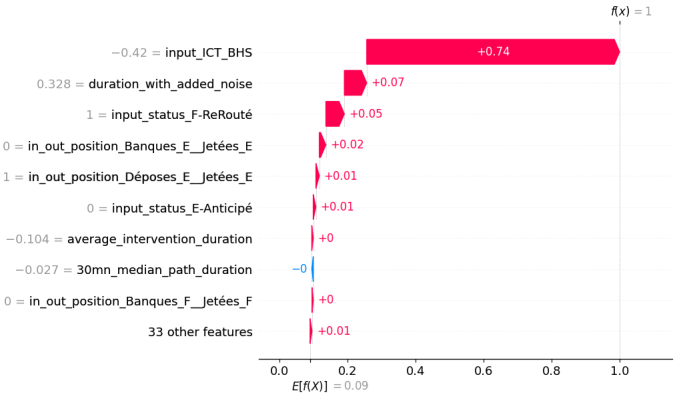


Figure 58: Shap values of a random mishandled baggage (3)

Acronyms

ADP Groupe Aéroports de Paris. 9–11, 16, 39

BHS Baggage Handling System. 9–13, 16–18, 22–24, 38, 39

CDGB Charles de Gaulle Baggage department. 10, 12

DGO Direction Générale des Opérations. 6, 10, 40

EB1 registration banks. 16

EBS bagage stocker. 16

EDC correspondence deposits. 16, 17

EJT piers. 17

ETP primary sorting. 16

FN false negative. 25

FP false positive. 25

LGBM Light Gradient Boosting Machine. 10

LR Logisic regression. 27, 33

Mbis Stocker M. 16

RF Random Forest. 6, 27, 28, 32, 33, 35, 37, 39

TBE Bagage handling system E. 12, 16, 21

TBF Bagage handling system F. 16

TN true negative. 25

TP true positive. 25

XGBoost Extreme Gradient Boosting. 6, 27, 28, 32, 33, 35, 37

References

[1] Abdelghanya, Abdelghanyb, and Narasimhanc. "Scheduling baggage-handling facilities in congested airports". In: *Journal of Air Transport Management* (2006). URL: https://www.sciencedirect.com/journal/journal-of-air-transport-management.

[2] Leo Breiman. "Random Forests". In: (Oct. 2001). URL: https://doi.org/10.1023/A:1010933404324.

[3] Leo Breiman, Jerome Friedman, and Charles Stone. "Classification and Regression Trees". In: *Classification and Regression Trees* (Jan. 1984). URL: https://books.google.fr/books/about/Classification_and_Regression_Trees.html?id=JwQx-WOmSyQC&redir_esc=y.

[4] Jason Brownlee. *How to Calibrate Probabilities for Imbalanced Classification*. 2020. URL: https://machinelearningmastery.com/probability-calibration-for-imbalanced-classification/.

[5] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: (June 2016). DOI: 1603.02754. URL: https://arxiv.org/pdf/1603.02754.

[6] Chen, Yang, Yu, Shi, and Chen. "A survey on imbalanced learning: latest research, applications and future directions". In: *Artificial Intelligence Review* (2024).

[7] Experian LatAm DataLab. *Imbalanced Binary Classification - A survey with code*. 2022. URL: https://pibieta.github.io/imbalanced_learning/notebooks/Calibration.html.

[8] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232.

[9] Herbert Van Leeuwen, Yingqian Zhang, Kalliopi Zevanou, Shantanu Mullick, Uzay Kaymak, and Tom De Ruijter. "Lost and Found: Predicting Airline Baggage At-risk of Being Mishandled". In: (Feb. 2020). URL: https://pure.tue.nl/ws/portalfiles/portal/146339725/ICAART_2020_141_CR_2.pdf.

[10] Gilles Louppe. "Understanding Random Forests". In: *University of Liège, Faculty of applied Science* (July 2014). URL: https://arxiv.org/pdf/1407.7502.

[11] Christoph Molnar. "Interpretable machine learning". In: (2024). URL: https://christophm.github.io/interpretable-ml-book/.

[12] Andreas Müller. *Calibration, Imbalanced Data*. 2020. URL: https://amueller.github.io/COMS4995-s20/slides/aml-10-calibration-imbalanced-data/#40.

[13] Alexandru Niculescu-Mizil and Rich Caruana. "Predicting good probabilities with supervised learning". In: (2005). URL: 10.1145/1102351.1102430.

[14] Emanuel Parzen. "On Estimation of a Probability Density Function and Mode". In: *Project Euclid* (Sept. 1962). URL: https://doi.org/10.1214/aoms/1177704472.

[15] Harrison Pim. *The best way to encode dates, times, and other cyclical features*. 2023. URL: https://harrisonpim.com/blog/the-best-way-to-encode-dates-times-and-other-cyclical-features.

[16] Van Der Sanden. "A Forecasting Framework for Recirculation in Baggage Handling Systems". In: (2020). URL: https://research.tue.nl/en/studentTheses/a-forecasting-framework-for-recirculation-in-baggage-handling-sys.

[17] ScikitLearn. *Robust Scaler documentation*. 2007 - 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.robust_scale.html#sklearn.preprocessing.robust_scale.

[18] ScikitLearn. *Train test split documentation*. 2007 - 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[19] Matthijs Van Den Hoven. "Utilization Improvement of a Sortation System for the Parcel Industry". In: (Sept. 2019). URL: https://research.tue.nl/en/studentTheses/utilization-improvement-of-a-sortation-system-for-the-parcel-indu.

[20] Wikipedia. *Ensemble Learning*. 2024. URL: https://en.wikipedia.org/wiki/Ensemble_learning.

[21] Developers XGBoost. *Introduction to Boosted Trees*. 2022. URL: https://xgboost.readthedocs.io/en/stable/tutorials/model.html.

[22] Bianca Zadrovzny and Charles Elkan. "Transforming Classifier Scores into Accurate Multiclass Probability Estimates". In: *Department of Computer Science and Engineering* (2002). URL: https://doi.org/10.1145/775047.775151.