Mining Risk Factors in RFID Baggage Tracking Data

Ahmed, Tanvir; Calders, Toon; Pedersen, Torben Bach

Published in:
IEEE 16th International Conference on Mobile Data Management

DOI (link to publication from Publisher):
10.1109/MDM.2015.31

Publication date:
2015

Document Version
Peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Ahmed, T., Calders, T., & Pedersen, T. B. (2015). Mining Risk Factors in RFID Baggage Tracking Data. In IEEE 16th International Conference on Mobile Data Management (Vol. 1, pp. 235-242). IEEE Computer Society Press. DOI: 10.1109/MDM.2015.31

Mining Risk Factors in RFID Baggage Tracking Data

Tanvir Ahmed[§],[†] Toon Calders[†]
[†]Université Libre de Bruxelles, Brussels, Belgium
Email: toon.calders@ulb.ac.be

Torben Bach Pedersen[§]
[§]Aalborg University, Aalborg, Denmark
Email: {tanvir, tbp}@cs.aau.dk

Abstract—**Airport baggage management is a significant part of the aviation industry. However, for several reasons every year a vast number of bags are mishandled (e.g., left behind, send to wrong flights, gets lost, etc.,) which costs a lot of money to the aviation industry as well as creates inconvenience and frustration to the passengers. To remedy these problems we propose a detailed methodology for mining risk factors from Radio Frequency Identification (RFID) baggage tracking data. The factors should identify potential issues in the baggage management. However, the baggage tracking data are low level and not directly accessible for finding such factors. Moreover, baggage tracking data are highly imbalanced, for example, our experimental data, which is a large real-world data set from the Scandinavian countries, contains only 0.8% mishandled bags. This imbalance presents difficulties to most data mining techniques. The paper presents detailed steps for pre-processing the unprocessed raw tracking data for higher-level analysis and handling the imbalance problem. We fragment the data set based on a number of relevant factors and find the best classifier for each of them. The paper reports on a comprehensive experimental study with real RFID baggage tracking data and it shows that the proposed methodology results in a strong classifier, and can find interesting concrete patterns and reveal useful insights of the data.**

I. Introduction

Aviation industry suffers from enormous loss due to baggage mishandling. A recent report [1] shows that in 2013, 3.13 billion passengers traveled by airlines and among them around 21 M passengers and 21.8 M bags were affected by baggage mishandling that costs 2.09 billion USD to the airline industry. It also creates frustration to the passengers during their vacation or business trips. Common baggage mishandlings are: left behind at the origin airport, missed connecting flight, bag loss, wrong bag destination, etc. A bag has to follow several steps while traveling from the origin airport to the final destination. The steps include check-in, screening, sorting, loading, transfer at the transit airport, arrival, etc. Mismanagement at any of these stages can be the reason for the bag being mishandled. The use of Radio Frequency Identification (RFID) in the baggage management system enables to track a bag while passing through different stages within an airport as well as across the airports. The massive baggage tracking can be very useful for analyzing and finding interesting patterns. Combining the tracking data with other dimensions like route information, flights and punctuality, day hours, week day, transit duration, etc., can reveal risk factors that are responsible for baggage mishandling.

Data mining is the process of extracting interesting patterns and knowledge from large data sets, and the acquired knowledge can be used for predicting unknown labels of new instances [10]. For example, from the baggage tracking data, we may be interested in knowing, *what is the probability that a bag will be mishandled if it has 35 minutes transit time at Copenhagen airport on Sunday morning*? However, before performing any data mining the data has to be well prepared such that the gained knowledge is useful. In the baggage tracking scenario, the generated huge volume tracking data are very low level and not directly suitable for further analysis. Therefore, relevant and important features need to be extracted from the unprocessed raw tracking data. Furthermore, the percentage of mishandled bags is very low (e.g., only 0.8% in our experimental data set) as compared to the percentage of correctly handled bags. It makes the data set highly imbalanced. This imbalance problem in the data set makes the mining process biased towards predicting that any bag will be handled correctly by default and makes it difficult to learn rules related to incorrectly handled bags. Thus, we need to take special care of this issue to get the mining techniques to work properly and to get patterns of higher quality.

In this paper, we propose a step by step methodology for performing data mining tasks to find interesting patterns and risk factors that are highly correlated with baggage mishandling. We present the essential steps for extracting a set of high-level features called *FlightLeg Records* for mining from the unprocessed raw RFID baggage tracking data. We have applied various classification techniques on the data set and deal with the imbalance problem by applying several different re-balancing techniques for finding the best predictive model. We also fragment the data set based on some important factors and learn specialized classification models for each fragment. We have conducted a comprehensive experimental study with a large amount of real-world RFID baggage tracking data from a major industry initiative called the BagTrack project (www.daisy.aau.dk/bagtrack). The data set has been collected from several airports in the Scandinavian countries. The experiment shows that fragmenting the data set helps to achieve better models. We also analyze and report some interesting patterns and risk factors that are discovered from the data set. The proposed methodology and techniques can help the aviation industry for examining baggage management problems and ultimately improving the baggage handling quality.

The remainder of the paper is organized as follows. Section 2 presents the preliminaries including the problem statement. Section 3 discusses the steps of the solution. Section 4 reports the experimental results. Section 5 reviews related work. Section 6 concludes and points to future work.

II. Preliminaries

RFID-Based Baggage Handling In airport baggage management a bag has to go through different steps to go from origin to final destination. Suppose that Nadia needs to travel from Aalborg Airport (AAL) to Arlanda Airport (ARN) via Copenhagen Airport (CPH). First, Nadia has to check-in and handover her bag to the check-in desk staff. Then the staff puts the bag on the conveyor belt for the automatic baggage sortation system. After passing all the steps inside AAL, the bag is loaded into the aircraft using belt loader for the targeted flight. As the bag has to be transferred to ARN, upon arrival at CPH it is shifted to the transfer system. After all the required stages at CPH, the bag is loaded to the aircraft for its next flight to ARN. After arriving at ARN, the bag is shifted to arrival belt and finally Nadia collects the bag from the arrival belt. During this journey, the bag has to go through up to 11 stages and there can be many baggage handlers handling the bag at the different stages.

Fig. 1 shows an example of RFID reader deployment at different locations of a baggage management system. An RFID reader is deployed in a fixed location and the position of the reader is recorded in the database. For example, *reader1* in Fig. 1 corresponds to *check-in1*, *reader6* corresponds to *Gateway-1* etc. The circles represent the RFID readers and their activation ranges.

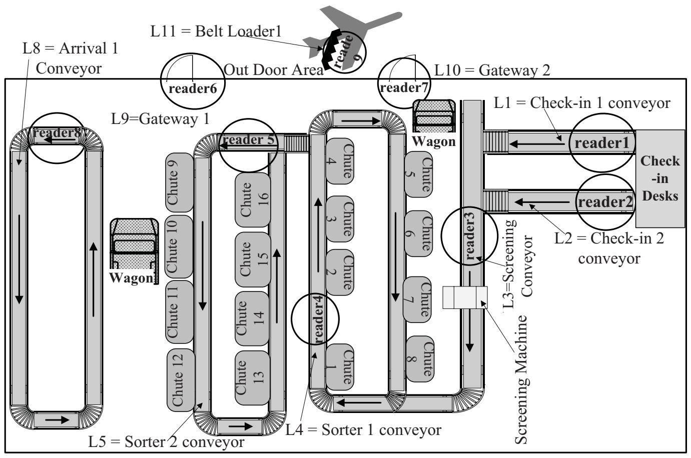
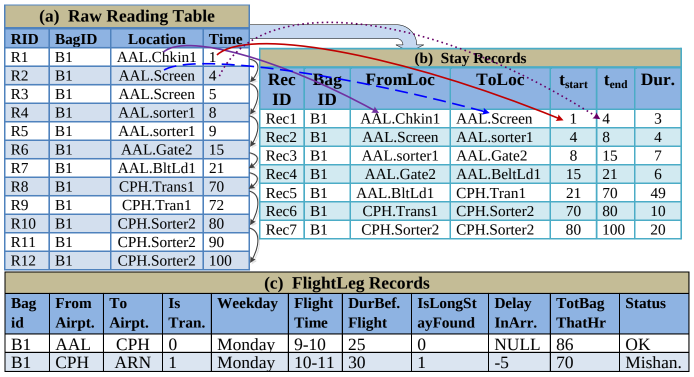


Fig. 1: RFID reader deployment in airport baggage tracking [3]

At check-in, an RFID tag is attached to the bag. The tag contains a small built in memory that stores bag information including the bag identifier, flights, route legs, date of departure, etc. While passing different stages, whenever a bag enters into a reader's activation range, it is continuously detected by the reader with a sampling rate which generates raw reading records of the form: ⟨*BagID, Location, Time, {info}*⟩, meaning that a reader at location *Location* detects a bag with ID *BagID* at timestamp *Time* and the tag stores the information *info*. Considering only location and time related information, some examples of raw reading records are shown in Fig. 2a. In the table, *RID* represents the reading identifier. As seen a bag can have several readings at the same location and on the basis of a single record, it is also not directly possible to compute how long an object spent in a particular location. To overcome these problems and prepare the data for further analysis we convert the raw reading records into stay records [3].


(a) Raw Reading Table

RID	BagID	Location	Time
R1	B1	AAL.Chkin1	1
R2	B1	AAL.Screen	4
R3	B1	AAL.Screen	5
R4	B1	AAL.sorter1	8
R5	B1	AAL.sorter1	9
R6	B1	AAL.Gate2	15
R7	B1	AAL.BltLd1	21
R8	B1	CPH.Trans1	70
R9	B1	CPH.Tran1	72
R10	B1	CPH.Sorter2	80
R11	B1	CPH.Sorter2	90
R12	B1	CPH.Sorter2	100

(b) Stay Records

Rec ID	Bag ID	FromLoc	ToLoc	t_{start}	t_{end}	Dur.
Rec1	B1	AAL.Chkin1	AAL.Screen	1	4	3
Rec2	B1	AAL.Screen	AAL.sorter1	4	8	4
Rec3	B1	AAL.sorter1	AAL.Gate2	8	15	7
Rec4	B1	AAL.Gate2	AAL.BeltLd1	15	21	6
Rec5	B1	AAL.BltLd1	CPH.Tran1	21	70	49
Rec6	B1	CPH.Trans1	CPH.Sorter2	70	80	10
Rec7	B1	CPH.Sorter2	CPH.Sorter2	80	100	20

(c) FlightLeg Records

Bag id	From Airpt.	To Airpt.	Is Tran.	Weekday	Flight Time	DurBef. Flight	IsLongStayFound	Delay InArr.	TotBag ThatHr	Status
B1	AAL	CPH	0	Monday	9-10	25	0	NULL	86	OK
B1	CPH	ARN	1	Monday	10-11	30	1	-5	70	Mishan.

Fig. 2: Example of getting stay records from raw records and example of FlightLeg records

Stay Records A stay record is of the form: *StayRecord*⟨*BagID, FromLocation, ToLocation,* t_{start}, t_{end}, *Duration, {StayInfo}*⟩ which represents that a bag with *BagID* first appeared at *FromLocation* at time t_{start} and then first appeared at the next location *ToLocation* at time t_{end}. It took *Duration* time to go from the reader at *FromLocation* to the reader at *ToLocation*. The *{StayInfo}* represents a set of other dimensional information related to the bag and to the transition (e.g., bag status, next flight schedule, origin and destination airports, and other flight-related information). For the final location of a bag, a special stay record is stored where the *FromLocation* and the *ToLocation* are same and the t_{start} and t_{end} represent the first and last times the bag appeared at that location. The stay records compress the huge data volume of raw readings and also enable to find abnormally long time spans between locations that may lead to baggage mishandling. Fig. 2b shows the stay records for the raw records of Fig. 2a. In the table, *RecID* represents the stay record identifier. In Fig. 2b, *Rec1* represents that bag *B1* had a transition from *AAL.Checkin-1* to *AAL.Screening* and it took 3 time units for the transition. The bag first appeared at *Checkin-1* at time 1 and then appeared at *Screening* at time 4. The stay records mainly introduce a very important feature which is the duration information. Nevertheless, it is still at a lower level for further higher level analysis. We are more interested in a higher level analysis like the baggage management performance at airport level, weekday, hour of day, and other relevant factors. We take the duration feature including some other dimensions from the stay records and create a table called *FlightLeg Records* which is described below.

FlightLeg Records The attributes and their descriptions

TABLE I: Description of the attributes of *FlightLeg records*

No.	Name (Data Type)	Description
1	*FromAirport (Symbolic)*	Departure airport of the corresponding flight
2	*ToAirport (Symbolic)*	Destination airport of the corresponding flight
3	*IsTransit (Boolean)*	A Boolean value representing whether it is a transit bag or not for the corresponding flight
4	*Weekday (Symbolic)*	Weekday of the corresponding flight
5	*FlightTimeHour (Symbolic)*	Departure hour of the corresponding flight
6	*DurationBeforeFlight (Integer)*	Available time (in minutes) for the bag to catch the flight
7	*IsLongerStayFound (Boolean)*	If any stay duration between readers at the *FromAirport* is longer than expected then it is *true*, otherwise it is *false*
8	*DelayInArrival (Integer)*	Delay in arrival (in minutes) of the arrival flight for the transit bag
9	*TotalBagInThatHour (Integer)*	Total number of bags read during the departure hour of the flight at *FromAirport*
10	*Status (Symbolic)*	Status of the bag i.e., 'OK' or 'Mishandled'

of the *FlightLegRecords* table are shown in Table I. For each flight of a bag we have one instance in the *FlightLegRecords* table. It captures some important features extracted from the *Stay Records* like {*IsTransit, DurationBeforeFlight, IsLongerStayFound, TotalBagInThatHour, Status*}. The way of calculating the value of *DurationBeforeFlight* varies based on whether the record belongs to transit or not (see description of *IsTransit* column). For a non-transit record, it is calculated from the first reading time of the bag at check-in and the actual departure time of the flight. Conversely, for a transit record, it is calculated from the actual flight arrival time at the *FromAirport* and actual departure time of the next flight to the *ToAirport*. The *DurationBeforeFlight* attribute is useful to see the effect on baggage mishandling due to the operating duration before departure. The value of *IsLongerStayFound* is determined by comparing the movement of baggage between readers at *FromAirport*. For each distinct transition between a pair of locations in the *Stay Records*, the bags that followed the top 5% longest durations are considered as *longer than expected*. The value of *DelayInArrival* is calculated from the actual and scheduled arrival times of the flight in the transit airport (i.e., *FromAirport* is a transit airport). For the non-transit records *DelayInArrival* is *NULL*. The status of the bag indicates whether the bag was mishandled or not in the *FromAirport*. The status of a bag is extracted from the reading records of the bag at the readers at *FromAirport*, flight timing, route information, etc. If a bag has any reading in the *FromAirport* after the corresponding flight departure time, then the bag is considered as left behind. Conversely, if a bag has any reading from an airport which is not in its planned route, then it is considered as wrong destination. Fig. 2c shows an example of the content of *FlightLegRecords*. We use the *FlightLegRecords* table for our further analysis.

Problem Statement Our primary goal is to find interesting patterns and identify risk factors that are correlated to baggage mishandling and ideally indicate appropriate corrective actions. We want to find bags with higher probability of being mishandled.

Definition 1 (Risk Score). Given a set of *FlightLeg records* F and an instance $f \in F$, the risk score $r \in \mathbb{R}$ is the probability estimate (PE) score for the instance f being *Mishandled*.

Definition 2 (Rank). Given a set of *FlightLeg records* F with assigned risk scores, the rank of a record $f \in F$ is the position of f in the list of F sorted by risk in descending order, i.e., the record with the highest risk score is ranked first.

We are interested in finding the best predictive model that can produce correct risk scores and the most accurate ranking of our data set.

III. Solution

For producing a risk score as well as a ranking we take the help of a classification algorithm. In order to find a risk score, the system has to learn from a set of training records and assign a risk score to each test example. We use the *Status* attribute of the *FlightLegRecords* as the class column. However, in our data set around only ***0.8%*** of the records belongs to *'Mishandled (MH)'* and the remaining ***99.2%*** of the records belongs to *'OK'*. With such an imbalanced distribution between the classes, the learning process gets biased towards predicting *OK* for almost all the instances by default and frequently misclassifies the *MH* instances. This is because a classifier tries to make the classification rules more general and considers the *MH* records as noise and discards the *MH* records. As a result, this *imbalance problem* should be handled wisely to overcome poor quality results otherwise produced by the classifier. It is also essential to choose an appropriate classification algorithm which will provide a good quality result for the given data set. To achieve the intended quality results from the raw baggage tracking data, we follow some essential steps. The steps of our solution are given in Fig. 3 and discussed in the following.

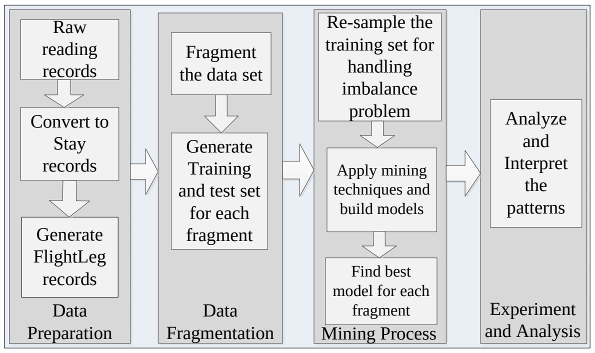

Raw reading records
Convert to Stay records
Generate FlightLeg records
Data Preparation
Fragment the data set
Generate Training and test set for each fragment
Data Fragmentation
Re-sample the training set for handling imbalance problem
Apply mining techniques and build models
Find best model for each fragment
Mining Process
Analyze and Interpret the patterns
Experiment and Analysis

Fig. 3: Outline of the steps

A. Data Preparation

Before applying any data mining technique and finding patterns, we need to pre-process the source data and select and construct relevant features. In this step, the raw baggage tracking records are converted into *Stay records* and then into the *FlightLegRecords*. The structure of the tables and steps of preparing such tables were already described in section II.

B. Data Fragmentation

Fragments The baggage management problem varies based on different important factors like whether the bag is in the transit airport or not, the duration of the transit, etc. Based on some important factors we have divided the data set into 5 fragments and applied data mining algorithms on each of the fragments for finding patterns specific to each fragment. Moreover, as the data set is imbalanced, the fragmentation allows examining the imbalance problems for specialized cases. Fig. 4 shows the different fragments of our data set, and the numbers inside the square bracket show the number of records and mishandling rate for the corresponding fragment in our experimental data. The combined records (CR) contain all the records of *FlightLegRecords* table. *CR* is divided into *transit records (TR)* and *non-transit records (NTR)*. To see the effect of transit duration on the baggage mishandling rate, we have drawn Fig. 5. It shows that almost all the bags (80-100 %) are mishandled when the transit duration is ≤ 31 (minutes). Based on transit duration, we have divided the *TR* into two different fragments. Transit records containing *DurationBeforeFlight*$\leq$*31* belong to fragment *Shorter Transit Records (STR)* and other transit records belong to fragment *Longer Transit Records (LTR)*. Both of these fragments help to analyze the shorter and longer transit baggage separately.

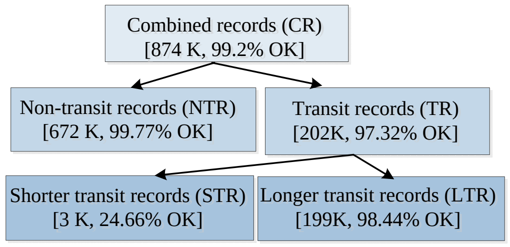


Fig. 4: Fragments of the data set

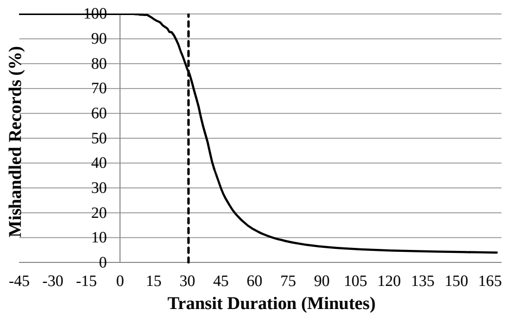


Fig. 5: Mishandling rate with change in transit duration

Training and test set Before applying data mining techniques, we have to prepare the training and test data sets. For each of the discussed fragments, we have one partition for training or learning (P1), and another partition for testing (P2). Strategies for obtaining the training and test sets are explained further in the experimental evaluation section.

C. Mining Process

Handling imbalanced data by re-sampling As discussed earlier, that the data set is highly imbalanced and learning directly from the data set will produce very poor quality patterns. To remedy these imbalance problems, we use 2 different kinds of sampling for the training data set (P1):
Undersampling technique **(US)**: in this technique a subset of *P1* is created by randomly deleting *OK* records until we reach equal number of records with class *OK* and class *MH*.
Oversampling technique **(OS)**: in this technique a superset of *P1* is created by copying some instances or generating new instances of *MH* records until we obtain an equal number of records for class *OK* and *MH*. We use Synthetic Minority Over-sampling Technique (SMOTE) [6] for getting *OS* data.

Mining Techniques We apply *Decision Tree (DT), Naive Bayes classifier (NB), KNN classifier (KNN), Linear regression (LIR), Logistics regression (LOR)*, and *Support vector machine (SVM)* on the training set *P1* of the combined records *CR* with the sampling strategies discussed above. We also do the same directly to P1 without re-sampling *(WS)*. Then we use different types of measures (discussed in the next paragraph) for finding the classification and sampling techniques that provide the best model for our data set. Then the chosen techniques are used for generating models for the remaining fragments. Note that we have deliberately chosen not to consider association rules mining technique to find out rules based on confidence and support scores, since this is an unsupervised technique, while our problem is supervised. We are only interested in modeling our target variable w.r.t. the other variables. In some sense, decision tree induction can be considered as a form of rule induction; every path from the root to a leaf actually represents an association rule.

Finding the best technique In general, a confusion matrix as shown in Table II is used for assessing the performance of a classifier. The confusion matrix shows how many test records are correctly and incorrectly classified for both positive and negative classes.

TABLE II: Confusion Matrix

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

Typically the performance of a classifier is evaluated by its predictive accuracy defined by, *Accuracy = (TP+TN)/(TP+FP+TN+FN)*. However, for an imbalanced data set the predictive accuracy is not an appropriate measure. For example, in our case an accuracy of 99% does not make sense, as it may misclassify all the examples as *OK (negative)* regardless of whether a record belongs to *Mishandled (positive)* or not. Here actually the classifier is 99% accurate in predicting the negative instances and 0% accurate in predicting the positive instances.

In an information retrieval system, *precision* is the measure which represents the relevance of the retrieved results, whereas *recall* represents the coverage of the relevant instances in the result. Precision is calculated as, *Precision = TP/(TP+FP)*, and recall (also known as TP rate) is calculated as, *Recall = TP/(TP + FN)*. A precision-recall (PR) curve is a good way to visualize the precision for a given recall. The points of a PR curve are calculated based on the generated scores of the classifier. The scores are sorted in descending order and each score is considered as a threshold for calculating the value of the precision and recall to draw a point in the PR curve.

The receiver operating characteristic (ROC) curve [14] is a well-known visualization of the performance of a ranker. The X-axis of an ROC curve represents false positive rate (FP rate = FP/(FP + TN)) and the Y-axis represents true positive rate (TP rate). So, it shows the trade-off between the TP rate and FP rate. The *Area Under the ROC Curve (AUC)* is a popular measure for evaluating the quality of ranking produced by a classifier [15]. The maximum value of an *AUC* can be 1 and it means that all the positive examples are placed in the top in the ranking. A classifier with AUC = 0.5 represents a random classifier that randomly guesses the classes.

In our cases, we consider two classes *'OK'* and *'Mishandled'* for classification and consider *'Mishandled'* as the positive class. In our scenario, misclassifying a *Mishandled* bag as *OK* (false negative) is more severe than misclassifying an OK bag as Mishandled. As such we are specifically interested in algorithms with a high recall on the *Mishandled* bags rather than in merely optimizing the accuracy of the classifier. In our case, we use the AUC as the main measure for choosing the model that provides the best ranking. We use precision-recall curves for finding which threshold provides higher precision for a good amount of recall.

IV. Experimental Evaluation

We use KNIME V2.9.2 (www.knime.org) for modeling our experimental work flows, applying data mining algorithms, producing and visualizing the results obtained from our data set. For preparing the data set from the source data we use different kinds of SQL queries and C# programs.

We use real RFID-based baggage tracking data, collected from 13 different airports with a total of 124 RFID readers deployed. There are 196 M raw reading records for 1.4 M bags collected for the period from *January 1, 2012* until *December 2, 2013*. In the original data set there are a lot of incomplete and erroneous records, e.g., missing flight and route information, unusual reading time, reading from unknown readers, etc. It creates problems while extracting different information about a bag like transit information, status at different stages, delay in departure and arrival, flight time hour, etc. As a relatively clean data set is an essential part for data mining, the problematic bags with the above mentioned incomplete information are filtered out during conversion into *stay records*, leaving us with 728K bags with 2.68 M *stay records*. Among these bags, some bags have stay records only in the arrival airports, which do not create any instances in the *FlightLegRecords* table. Finally, after converting *stay records* into *FlightLeg records* we have 671,712 bags with 874,347 *flight leg records* for mining. Among them only 0.8% of the records belongs to the class *Mishandled* (MH), the remaining 99.2 % belongs to the class *OK*. For each fragment, the total number of records and percentages of *OK* are shown in Fig. 4. Among the fragments, only *STR* contains a higher number of *MH* records than *OK* records. In the rest of this section, we will show how different classification algorithms (discussed in Section III-C) perform on the combined records with different kinds of re-sampling techniques. Then we will identify the best classification and sampling technique and discuss the obtained patterns and analysis results from the data.

For the Decision Tree Induction, the C4.5 algorithm is used with the *Gini index* quality measure and the *MDL* pruning method. To reduce the number of branches, the minimum number of records per node is set to 100. For the KNN classifier, we tried with K=5 and K=7. As in both of the cases the results were similar, we finally report for K=7. Before applying KNN, linear regression, logistics regression, and SVM the structure of the input data table is changed as these algorithms do not work with categorical attributes. In these cases, we convert each value of a categorical attribute into a separate column and put Boolean 0 or 1 accordingly. An example of such conversion for Fig. 2c is shown in Table III. For the linear regression and the logistics regression, the attributes with continuous values are normalized into the [0;1] interval. In our data set all the *FromAirport*s are within the *Schengen territory* (http://en.wikipedia.org/wiki/Schengen_Area) and a person traveling within this territory does not require any special passport control. Unlike *FromAirport*s we have too many values in the *ToAirport* column which creates many branches in the decision tree and for other classification algorithms this column become useless. To make the *ToAirport* column useful and make the learned pattern interesting we categorized the *ToAirport*s into three types: *Domestic, Schengen, and Others*.

TABLE III: Converting string values into columns of Fig. 2c

AAL	CPH	IsTransit	Monday	9-10	10-11	....
1	0	0	1	1	0	...
0	1	1	1	0	1	...

In our experiments, we split the available data into a training and a test set based on the date of the record. All records before *15- May-2013* were included in the training set *(P1)* [Total: 615K, OK: 99.2%] and all records from that date or later were added to the test set *(P2)* [Total: 259K, OK: 99.18%]. The reason we did not rely on a standard cross-validation approach is because there exist dependencies between the bags. Bags that were on the same plane are more likely to have similar properties, as well as a similar class label. Therefore spreading bags of the same flight over both the training and test set may cause a biased estimation of the performance due to overfitting. By dividing the data based on date, we can guarantee that the training set and the test set are independent, and we get an unbiased estimate of the performance of the mined models.

Overall from Table IV in all the cases we can see that the AUCs are better than a random classifier predicting by default class *OK* for every bag. It shows that the re-balancing technique (i.e., WS, US, and OS) has a high impact on the AUCs of some classifiers, whereas it has almost no effect for the *Naive Bayes classifier*. It shows that the decision tree

TABLE IV: The table below lists the AUCs with the different types of classification algorithms and re-balancing/re-sampling techniques for the combined records (CR)

Re-sampling	**DT**	**NB**	**KNN**	**LOR**	**LIR**
WS	0.88	0.83	0.71	0.82	0.78
US	0.87	0.83	0.79	0.85	0.79
OS	0.81	0.83	0.78	0.83	0.74

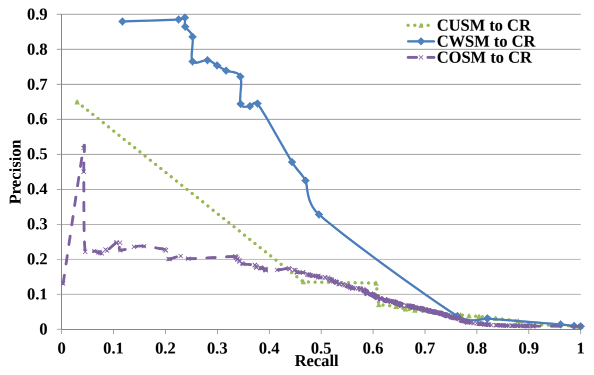


Fig. 6: Precision-recall curves for CWSM, CUSM, and COSM tested on the test set of the combined records (CR)

produces the highest AUC compared to all the other classifiers. The AUCs produced by the over-sampling (OS) technique shows that it is not helping to produce a better ranking in our data set. The performance of over-sampling is unpredictable given that without knowing the precise process that generated the data, it is hard to generate good synthetic examples. During our experiments, we found that the SVM learning process did not produce any results within a reasonable time (several days). So, we do not report any AUC with SVM. As the decision tree produces the highest AUC (i.e., the best ranking), we consider it as the best classifier for our scenario. For our further experiments, we will only use the decision tree *without re-sampling (WS)* and with *US* for producing other models. We call the model generated by the decision tree without re-sampling (*WS*) the Combined Without-sampling Model (CWSM), with *US* the Combined Under-sampling Model (CUSM), and with *OS* the Combined Over-sampling Model (COSM). Fig. 6 shows the precision-recall (PR) curves for *CWSM*, *CUSM*, and *COSM* when applied to the *CR*. It shows that *CWSM* produces the best PR curve that always gives a higher precision for the different values of recall compared to the other two models. It shows that for 50% recall, we can get 34% precision. It means that the ranking produced by the decision tree contains 35% of the actual *MH* records among the top 50% predicted *MH* records.

After finding the best classification algorithm and short listing the re-balancing techniques we conduct further experiments on the different fragments. We learn decision tree models for *NTR, TR, STR,* and *LTR* without re-balancing (*WS*) and respectively they are called:

- *Non-transit Without-sampling Model (NTWSM)*
- *Transit Without-sampling Model (TWSM)*
- *Shorter Transit Without-sampling Model (STWSM)*
- *Longer Transit Without-sampling Model (LTWSM)*

TABLE V: AUCs for models built from different fragments and testing on relevant fragments

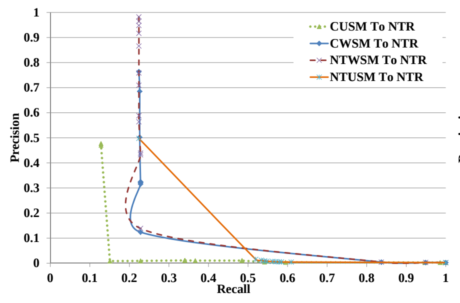
Sampling	**Model**	**CR**	**NTR**	**TR**	**STR**	**LTR**
WS	**CWSM**	**0.88**	**0.74**	0.79	**0.79**	0.66
	NTWSM	-	**0.74**	-	-	-
	TWSM	-	-	0.67	0.72	0.5
	STWSM	-	-	-	0.73	-
	LTWSM	-	-	-	-	0.61
US	**CUSM**	0.87	0.67	0.82	0.54	0.77
	NTUSM	-	0.73	-	-	-
	TUSM	-	-	**0.85**	0.5	**0.78**
	STUSM	-	-	-	0.77	-
	LTUSM	-	-	-	-	**0.78**

We also learn decision tree models for the fragments with *US* and respectively they are called:

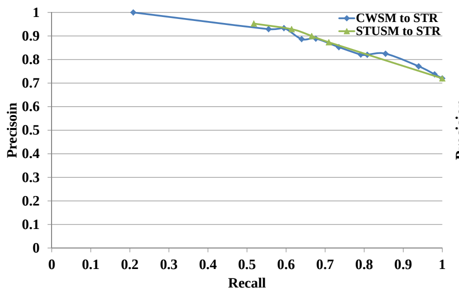
- *Non-transit Under-sampling Model (NTUSM)*
- *Transit Under-sampling Model (TUSM)*
- *Shorter Transit Under-sampling Model (STUSM)*
- *Longer Transit Under-sampling Model (LTUSM)*

For all the cases, the training and test sets are taken by filtering the data from *P1* and *P2* of the *CR*. Then we apply the models *CWSM* and *CUSM* to the test sets of all these fragments. We also apply all the other models to the relevant fragments for finding the best models for each of the fragments. Table V shows the AUCs for the models and cross checking with the different fragments. It shows that the individual models give a reasonable AUC with their own fragment. The AUCs with the *TR* shows that both fragmenting and re-balancing helps to achieve better models for the transit cases. For the *NTR*, models without re-balancing (i.e, *CWSM* and *NTWSM*) produce better ranking. Fig. 7a shows the PR curves of different models when applied to the *NTR*. It shows that both *CWSM* and *NTWSM* gives very similar precision for different values of recall. So, from the AUCs and PR curves we can conclude that both *CWSM* and *NTWSM* can produce better ranking and patterns for the *NTR* compared to the other models. Table V shows that *CWSM* produces the highest AUC for the *STR*, and the next closer AUC is produced by *STUSM*. Fig. 7b shows the PR- curves of these two models with the *STR* and both of them produces very similar curves and for the higher value of recall at some points *CWSM* produces higher precision. So, we can conclude from the AUCs and PR-curves that *CWSM* is the best model for the *STR*. For the *LTR*, it is clear that the data must be re-balanced before learning for this type of cases. Fig. 7c shows the PR curves for *CUSM, TUSM,* and *LTUSM* when applied to the *LTR*. It shows that *TUSM* produces the best PR curve. So, from the AUCs and PR curves we can conclude that *TUSM* is the most appropriate model for the *LTR*.

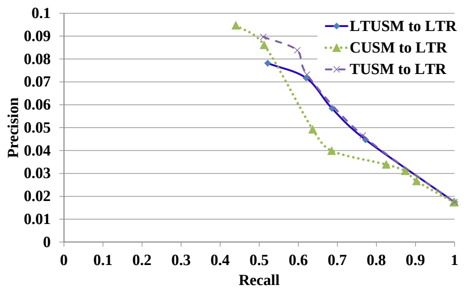
The fragmentation helps to build specialized models; however, it also reduces the training data size. To see the effect of training data size on the AUC, we learned decision tree models with the *CR* (without re-balancing) with different data size and the results are reported in Fig. 8. It shows that for lower numbers of training data set like 20K and 40K the AUCs are low and with increase in the number of training examples it becomes stable at 0.88.



(a) Models tested on NTR



(b) Models tested on STR



(c) Models tested on LTR

Fig. 7: Precision-recall curves for different models tested on NTR, STR, and LTR

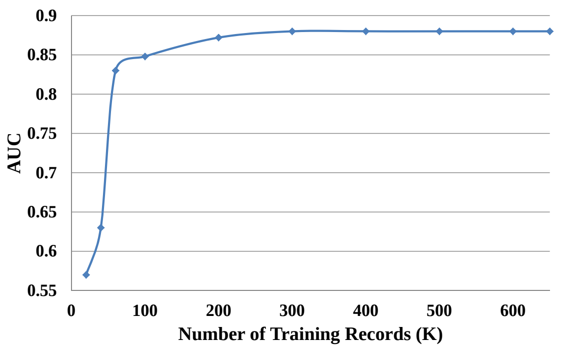


Fig. 8: AUC with different size of training data

From the comprehensive experiments, we can conclude that for achieving an unbiased and good ranking the training data may need to be re-balanced before applying the data mining tasks. In our data set the decision tree C4.5 algorithm is the most appropriate choice for classification and ranking. In our case, re-balancing with under-sampling helps us to achieve a better model for the transit bags. We also learn that for a better ranking, it may require learning specialized models for different groups of data in the whole data set like we did for non-transit, transit, shorter and longer transit data. In our data set for the non-transit records, longer transit records, and shorter transit records the chosen models are *CWSM, TUSM,* and *CWSM* respectively. Moreover, it is also learned that a larger number of training records also important for a better and stable ranking.

Extracted Rules and Analysis: We now explore the patterns found for the different fragments by their chosen classifier and the assigned probability estimate scores (risk scores) of each pattern. For each fragment, we report the top 5 rules with the highest risk scores followed by analysis. For reasons of confidentiality, the airport names in a pattern have been changed to A1, A2, ..., A6.

***For NTR by CWSM*:**

Rule1: *If DurationBeforeFlight≤2min ⇒ MH* [Score: 0.98]

Rule2: *If DurationBeforeFlight>2min AND FromAirport=A8 AND IsLongerStayFound=1 AND TotalBagInThatHour> 192 ⇒ MH* [Score: 0.88]

Rule3: *If DurationBeforeFlight>2min AND FromAirport=A8 AND IsLongerStayFound=1 AND TotalBagInThatHour≤192 AND DurationBeforeFlight≤65min ⇒ MH* [Score: 0.66]

Rule4: *If DurationBeforeFlight>2min AND FromAirport=A8 AND IsLongerStayFound=1 AND TotalBagInThatHour≤192 AND DurationBeforeFlight>65min ⇒ OK* [Score: 0.27]

Rule5: *If DurationBeforeFlight>2min AND FromAirport=A2 AND DurationBeforeFlightFlight >40min ⇒ OK* [Score: 0.12]

***For STR by CWSM*:**

Rule1: *If DurationBeforeFlight≤2min ⇒ MH* [Score: 0.98]

Rule2: *If DurationBeforeFlight>2min AND FromAirport=A2 AND DurationBeforeFlight ≤25min ⇒ MH* [Score: 0.93]

Rule3: *If DurationBeforeFlight>2min AND FromAirport=A3 AND DurationBeforeFlight ≤ 9min ⇒ MH* [Score: 0.91]

Rule4: *If DurationBeforeFlight>25min AND FromAirport=A2 AND TotalBagInThatHour>145 ⇒ MH* [Score: 0.71]

Rule5: *If DurationBeforeFlight > 29min AND FromAirport=A2 AND IsLongerStayFound=1 AND TotalBagInThatHour>115 AND DurationBeforeFlight ≤ 35min ⇒ MH* [Score: 0.63]

***For LTR by TUSM*:**

Rule1: *If DurationBeforeFlight≤54min ⇒ MH* [Score: 0.88]

Rule2: *If DurationBeforeFlight>54min AND IsLongerStayFound=1 AND DurationBeforeFlight≤75min ⇒ MH* [Score: 0.64]

Rule3: *If DurationBeforeFlight>75min AND IsLongerStayFound=1 AND DurationBeforeFlight ≤ 95min ⇒ OK* [Score: 0.45]

Rule4: *If DurationBeforeFlight>95min AND IsLongerStayFound=1 ⇒ OK* [Score: 0.28]

Rule5: *If DurationBeforeFlight>54min AND IsLongerStayFound=0 ⇒ OK* [Score: 0.18]

The above rules show that available baggage handling time before the flight departure is always an important issue regardless of the category of the bag. For the non-transit bags the departure airport is a very important factor and based on the *FromAirport* the other risk factors like check-in time of the bag before the flight, longer stay between locations, and total number of bags during the flight hour have high influence on the baggage management problem. Rules 4 and 5 of the *NTR* can be discarded as they have very low risk scores. For the *STR*, it is considered to be mishandled by default. The risk factors and the effect of transition duration for the *STR* also vary based the transit airport. The rules show that when the transit duration increases, the other factors like a longer stay between baggage handling locations and number of bags during the flight hour take influence on the baggage management problem. In case of longer transit records, a record with *DurationBeforeFlight ≤ 54min* is directly classified as *MH* regardless of any other condition. This condition also reflects

Fig. 5 discussed earlier, where the mishandling rate suddenly started increasing fast around this point, and it is almost 100% when the duration ≤ 31min. The rules of *LTR* also show that if *DurationBeforeFlight* $> 54min$ then a longer stay at a location is highly responsible for baggage mishandling. The rules 3, 4, and 5 of the *LTR* can be discarded due to their low risk scores.

V. Related Work

Related work falls into two main categories. One is to pre-process unstructured RFID-based tracking data, and another one is to perform data mining task on imbalanced data set. Warehousing and mining techniques of RFID data from supply chain systems have been proposed in [9]. They convert the raw RFID records into cleansed record containing the first and last reading times of an object under the readers activation range. They took the advantage of bulky movement of objects for compressing the huge volume of RFID data. A data warehouse for analyzing RFID-based baggage tracking data is proposed in [3], where the raw tracking records are converted into *StayRecords* along with other dimensions. In [2], [4] the raw reading records are converted into mapping records containing the entry and exit times of an object at a constrained (e.g., conveyor belts of airport baggage management) and semi-constrained indoor symbolic locations (e.g., large hall, rooms, etc.,). In the present paper, we further refine the *stay records* into *FlightLeg records* that capture different aggregate information from the stay records including other dimensional information for a higher level analysis.

Several papers address the problems of mining with imbalanced data set [6], [13]. The main approaches of dealing with imbalanced data are re-sampling (includes under-sampling [12] and over-sampling [6]) and cost-sensitive learning [8]. Measuring the performance of classifiers and comparing models specially for imbalanced data set scenario have been discussed widely [6], [13]. We use AUC as the main measure [5], [11], [12] for comparing the models as well as present precision-recall curve as there is a deep relation between ROC and PR space [7]. In the present paper, we apply several data mining techniques with different re-balancing techniques for finding best classifier and re-balancing techniques that can provide a good ranking in our data set.

VI. Conclusion and Future work

In this paper, we proposed a detailed methodology for finding risk factors from the imbalanced RFID airport baggage tracking data. We presented the pre-processing steps for preparing the raw RFID tracking data into *FlightLeg records*. We estimated the risk score of a bag being mishandled. In order to compute the risk scores, we learned classifiers that assigned scores and then evaluated the quality of the scores with the AUC measure. We dealt with the imbalance problem, applied different data mining techniques, and based on AUCs and Precision-Recall curves we found that the decision tree is the best classifier for our data set. We fragmented the data set into transit, non-transit, shorter and longer transit and obtained the appropriate models for the different fragments. We also found that re-balancing the data set by under-sampling helps to achieve a better predictive model for the longer transit bags. We conducted comprehensive experiments with real baggage tracking data, and it showed that fragmenting and mining each of the fragments separately was a right choice. The extracted patterns show that overall available handling time for a bag is a critical factor and; more specifically, a bag is considered to be a high risk if it has less than 54 minutes in the transit airport. For non-transit bags, the factors depend on the departure airport. It was also found that a longer stay between baggage handling locations and the total number of bags during the flight hour are important factors to predict mishandling as well. The proposed methodology can help the aviation industry with examining baggage management problems for further improvement in the system.

Several directions for future work exist. First, a more thorough study of the root causes for mishandling, which is non-trivial, given the low probability of Mishandled events. Second, analyzing baggage handling sequences for finding problems in the system. Third, finding spatio-temporal outliers from the RFID baggage tracking data. Fourth, developing native support from the data mining tools like automatic methods for finding the most appropriate models.

Acknowledgment

This work is supported by the BagTrack project funded by the Danish National Advanced Technology Foundation under grant no. 010-2011-1.

References

[1] Sita baggage report 2014. www.sita.aero/content/baggage-report-2014.

[2] T. Ahmed, T. B. Pedersen, and H. Lu. Capturing hotspots for constrained indoor movement. In *SIGSPATIAL/GIS*, pages 462–465, 2013.

[3] T. Ahmed, T. B. Pedersen, and H. Lu. A data warehouse solution for analyzing rfid-based baggage tracking data. In *MDM (1)*, pages 283–292, 2013.

[4] T. Ahmed, T. B. Pedersen, and H. Lu. Finding dense locations in indoor tracking data. In *MDM (1)*, pages 189–194, 2014.

[5] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, 1997.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *JAIR*, 16:321–357, 2002.

[7] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *ICML*, pages 233–240, 2006.

[8] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *SIGKDD*, pages 155–164, 1999.

[9] J. Han, H. Gonzalez, X. Li, and D. Klabjan. Warehousing and mining massive RFID data sets. In *ADMA*, pages 1–18, 2006.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[11] J. Huang and C. X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 17(3):299–310, 2005.

[12] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 39(2):539–550, 2009.

[13] V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Inf. Sci.*, 250:113–141, 2013.

[14] F. J. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *KDD*, pages 43–48, 1997.

[15] H. Zhang and J. Su. Naive bayesian classifiers for ranking. In *ECML*, pages 501–512, 2004.