Mining Risk Factors in RFID Baggage Tracking Data

Tanvir Ahmed§, † Toon Calders‡
1Université 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. Mismangement 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 flight, the bag is loaded into the aircraft using belt loader for the targeted flight. The bag first appeared at AAL.Checkin-1 to Screening Machine and then appeared at AAL.Gate2. After screening, Nadia puts the bag on the conveyor belt for the automatic baggage handover her bag to the check-in desk staff. Then the staff checks the bag and then appeared at AAL.Screen.

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.

![RFID reader deployment](image)

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: $\langle BagID, Location, Time, Info\rangle$, 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].

### Example of getting stay records from raw records and example of FlightLeg records

**Stay Records** A stay record is of the form:

StayRecord$\langle BagID, FromLocation, Location, Time, FromTime, ToTime, Duration, \{StayInfo\}\rangle$ which represents that a bag with BagID first appeared at FromLocation at time FromTime and then first appeared at the next location Location at time ToTime. 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 FromTime and ToTime 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
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.

**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.

---

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

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

---

**Fig. 3: Outline of the steps**
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≤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](image)

![Fig. 5: Mishandling rate with change in transit duration](image)

**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>
<thead>
<tr>
<th>Actual Positive</th>
<th>Predicted Positive</th>
<th>Predicted Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
<td></td>
</tr>
<tr>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
<td></td>
</tr>
</tbody>
</table>

Table II: Confusion Matrix

Typically the performance of a classifier is evaluated by its predictive accuracy defined by, \( Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \). 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, \( Precision = \frac{TP}{TP+FP} \), and recall (also known as TP rate) is calculated as, \( Recall = \frac{TP}{TP + FN} \). 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 case, 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 (false negative) 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, 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 FromAirports 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 FromAirports 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 ToAirports into three types: Domestic, Schengen, and Others.

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

<table>
<thead>
<tr>
<th>AAL</th>
<th>CPH</th>
<th>EctTransit</th>
<th>Monday</th>
<th>9-10</th>
<th>10-11</th>
<th>....</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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 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)

<table>
<thead>
<tr>
<th>Re-sampling</th>
<th>DT</th>
<th>NB</th>
<th>KNN</th>
<th>LOR</th>
<th>LIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS</td>
<td>0.88</td>
<td>0.85</td>
<td>0.71</td>
<td>0.82</td>
<td>0.78</td>
</tr>
<tr>
<td>US</td>
<td>0.87</td>
<td>0.83</td>
<td>0.79</td>
<td>0.85</td>
<td>0.79</td>
</tr>
<tr>
<td>OS</td>
<td>0.81</td>
<td>0.83</td>
<td>0.78</td>
<td>0.83</td>
<td>0.74</td>
</tr>
</tbody>
</table>

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 Under-sampling Model (STWSM)
- Longer Transit Under-sampling Model (LTWSM)

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

<table>
<thead>
<tr>
<th>Sampling</th>
<th>Model</th>
<th>CR</th>
<th>NTR</th>
<th>TR</th>
<th>STR</th>
<th>LTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS</td>
<td>CWSM</td>
<td>0.88</td>
<td>0.74</td>
<td>0.79</td>
<td>0.79</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>NTWSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.67</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>TWSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>STWSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>LTWSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>US</td>
<td>CUSM</td>
<td>0.87</td>
<td>0.67</td>
<td>0.82</td>
<td>0.54</td>
<td>0.77</td>
</tr>
<tr>
<td></td>
<td>NTUSM</td>
<td>-</td>
<td>0.73</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TUSM</td>
<td>-</td>
<td>-</td>
<td>0.85</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>STUSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.77</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>LTUSM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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 TUSM. 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.
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\(\leq 2\text{min}\) \(\Rightarrow\) MH [Score: 0.98]

**Rule2:** If DurationBeforeFlight\(>2\text{min}\) AND FromAirport=A8 AND IsLongerStayFound=1 AND TotalBagInThatHour\(\leq 192\) \(\Rightarrow\) MH [Score: 0.88]

**Rule3:** If DurationBeforeFlight\(>2\text{min}\) AND FromAirport=A8 AND IsLongerStayFound=1 AND TotalBagInThatHour\(\leq 192\) AND DurationBeforeFlight\(\leq 65\text{min}\) \(\Rightarrow\) OK [Score: 0.66]

**Rule4:** If DurationBeforeFlight\(>2\text{min}\) AND FromAirport=A8 AND IsLongerStayFound=1 AND TotalBagInThatHour\(\leq 192\) AND DurationBeforeFlight\(>65\text{min}\) \(\Rightarrow\) OK [Score: 0.27]

**Rule5:** If DurationBeforeFlight\(>2\text{min}\) AND FromAirport=A2 AND DurationBeforeFlightFlight\(>40\text{min}\) \(\Rightarrow\) OK [Score: 0.12]

**For STR by CWSM:**

**Rule1:** If DurationBeforeFlight\(\leq 2\text{min}\) \(\Rightarrow\) MH [Score: 0.98]

**Rule2:** If DurationBeforeFlight\(>2\text{min}\) AND FromAirport=A2 AND DurationBeforeFlight\(\leq 25\text{min}\) \(\Rightarrow\) MH [Score: 0.93]

**Rule3:** If DurationBeforeFlight\(>2\text{min}\) AND FromAirport=A3 AND DurationBeforeFlight\(\leq 9\text{min}\) \(\Rightarrow\) MH [Score: 0.91]

**Rule4:** If DurationBeforeFlight\(>25\text{min}\) AND FromAirport=A2 AND TotalBagInThatHour\(>145\) \(\Rightarrow\) MH [Score: 0.71]

**Rule5:** If DurationBeforeFlight\(>29\text{min}\) AND FromAirport=A2 AND IsLongerStayFound=1 AND TotalBagInThatHour\(>115\) AND DurationBeforeFlight\(\leq 35\text{min}\) \(\Rightarrow\) MH [Score: 0.63]

**For LTR by TUSM:**

**Rule1:** If DurationBeforeFlight\(\leq 54\text{min}\) \(\Rightarrow\) MH [Score: 0.88]

**Rule2:** If DurationBeforeFlight\(>54\text{min}\) AND IsLongerStayFound=1 AND DurationBeforeFlight\(\leq 75\text{min}\) \(\Rightarrow\) MH [Score: 0.64]

**Rule3:** If DurationBeforeFlight\(>75\text{min}\) AND IsLongerStayFound=1 AND DurationBeforeFlight\(\leq 95\text{min}\) \(\Rightarrow\) OK [Score: 0.45]

**Rule4:** If DurationBeforeFlight\(>95\text{min}\) AND IsLongerStayFound=1 \(\Rightarrow\) OK [Score: 0.28]

**Rule5:** If DurationBeforeFlight\(>54\text{min}\) AND IsLongerStayFound=0 \(\Rightarrow\) 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\(\leq 54\text{min}\) 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 ≤ 31 min. The rules of LTR also show that if DurationBeforeFlight > 54 min 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 preprocess 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