SIGNAL PROCESSING AND MACHINE LEARNING FOR AIR TRAFFIC DELAY PREDICTION

Víctor M. Tenorio
Department of Signal Theory and Communications, King Juan Carlos University, Spain
Antonio G. Marques
Department of Signal Theory and Communications, King Juan Carlos University, Spain
Luis Cadarso
Aerospace Systems and Transport Research Group, European Institute for Aviation Training and Accreditation (EIATA), King Juan Carlos University, Spain

ABSTRACT

As data quality and quantity increase, the prediction of future events using machine learning (ML) techniques across engineering disciplines grows by the day. Air transportation cannot be an exception. Delay prediction is paramount in the aerospace industry, since air traffic delays are responsible for millions of dollars in losses to airlines and passengers, along with negative impacts on the environment. In this contribution, we leverage recent signal processing and ML advances to put forth a processing-and-learning pipeline for the prediction of air traffic delays. The proposed approach is executed in several steps. Firstly, we apply signal processing and data science techniques to filter and denoise the original information. Secondly, we run a descriptive analysis of the data and design new features tailored to the prediction problem. Thirdly, we implement a scheme to select the most informative of those features, contributing to a better generalization performance, and offering useful insights. Two algorithms are used to that end: one based on random forests and one employing a sparse logistic regression approach. Finally, once the features are selected, we implement, analyse, and compare several ML architectures (from classical classifiers to deep learning) to predict the delay. While the focus of the comparison is prediction accuracy, metrics such as sample and computational complexity are also discussed. Numerical experiments are drawn from the US domestic market for the year 2018, when more than 7 million flights between 358 airports were flown. The designed processing/learning pipeline reveals interesting insights and achieves better prediction results than the state of the art. The results confirm that air traffic delay prediction is a challenging problem, mainly because the delay is extremely airport-dependent and the data is highly unbalanced (i.e., only a small percentage of flights are noticeable delayed), and identify worth-pursuing future lines of work.

1. INTRODUCTION

During the year 2018, 46.1 million flights were operated in the world, with more than 4.4 billion passengers being transported (The International Air Transport Association (IATA), 2020), which represents 3.5% more flights and 6.4% more passengers than during the previous year. More importantly, traffic predictions for the next decade foresee that the growth of traffic will persist, further stressing the air traffic system and challenging its efficient operation. With these numbers in mind, and given the role that the aerospace system plays in our lives, initiatives targeted at a more efficient operation of the system can generate remarkable social and economic benefits. A particularly challenging aspect is that of air traffic delays. To illustrate the importance of this point, let us focus on the US domestic market. During 2018, the 7 million domestic flights connecting US cities experienced an average delay of 9.9 minutes, yielding an aggregated delay of more than 133 years. According to Eurocontrol, each minute of delay costs more than 100€ to the system (Mulligan, 2019), including all the stakeholders. The Federal Aviation Administration (FAA) and the Nextor consortium in the US estimated the delay cost to be greater than 28 billion dollars in 2018 (Airlines for America, 2019). From this point of view, being able to predict air traffic delays before they take place would provide substantial benefits in the context of daily operations. Although the costs associated with delays would not be fully eliminated, airlines could probably mitigate them by taking proactive actions, such as rescheduling or cancelling some flights in advance.

Using past and current information to predict future flight delays is, however, not an easy task. Precise mathematical models describing the delay dynamics are to be had, the available information is noisy and oftentimes incomplete, and the historic delays seem to depend heavily on the particular airport and time of the year. Hence, rather than pursuing a model-based approach, this paper puts forth a data-based approach that combines recent advances in signal processing (SP) and machine learning (ML) to address the problem of air traffic prediction using historical information. In recent years, ML and artificial intelligence algorithms have excelled in a number of engineering problems (from image recognition to language processing) beating the state of the art in areas where well-established model-based approaches had been used for decades, and providing a feasible alternative in areas where model-based designs were inexistent or poorly accurate.

The approach implemented in this paper proceeds in three steps. In the first step, we design SP and ML algorithms to identify, aggregate and extract relevant features from a database containing past information about air traffic delays. In the second step, we use the output of step 1, together with information of actual flight delays, to train different supervised ML architectures. The third step consists in using the trained ML architectures to predict current delays. The final goal is threefold: 1) providing a delay prediction accuracy that beats the state of the art, 2) testing and comparing a broad range of ML architectures that exploit

different levels of information structure, and 3) gaining insights on the key factors and trade-offs that govern the problem of air-traffic delay prediction.

The number of works dealing with the prediction of air traffic delays is limited, which is likely a testament to the complexity of the problem. To the best of our knowledge, Rebollo (2012) was the first publication that addressed the problem of flight delay prediction over origin-destination (OD) pairs using ML algorithms, a problem further explored in Rebollo & Balakrishnan (2014), Hanley (2015), and Gopalakrishnan & Balakrishnan (2017). These works recast the delay prediction as a binary classification task, so that the problem reduces to estimating if the delay is going to be below or over a given threshold in a specific OD pair. Addressing the delay prediction as a decision task, which is also our approach, brings a number of advantages, but it also generates some challenges, being the so-called "unbalance" between the positive class (i.e., flights where delay exists) and the negative class (i.e., flights where delay does not occur) the most challenging one (see Section 4 for details). Regarding the ML architecture to predict the delay, those works rely mostly on decision trees (DT) and Random Forest (RF) classification algorithms, with Gopalakrishnan & Balakrishnan (2017) exploring other architectures and concluding that the best results were achieved by Neural Networks (NNs).

Relative to those works, our contributions are the following. Firstly, we predict not only delays at OD pairs but also at airports. Secondly, we implement several SP and data-science techniques for pre-processing the raw data, extracting relevant knowledge, and designing new features to serve as input to the classification problem. Thirdly, we design a problem-tailored feature-selection scheme to identify the most relevant features. Fourthly, we employ a range of (classical and modern) ML algorithms to address the classification, comparing pros and cons, and assessing their accuracy and computational complexity. Finally, we consider rebalancing training techniques to counteract the severe lack of balance present in the dataset. Regarding the ML algorithms implemented, it is also important to remark that some of the proposed architectures account for the specific structure of the data at hand, namely its temporal variation by using Long-Short Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997), and the topology of the air-traffic network (specifically, the graph connecting the different airports) by using Graph Convolutional Neural Networks (GCNN) (Gama et al., 2019). Our numerical tests, which are based on the US domestic market for the year 2018, reveal that the proposed processing and learning pipeline yields a better prediction performance -namely better precision, recall and f1-score (Geron, 2017)-, than that achieved by the state-of-the-art air-traffic delay prediction algorithms.

Section 2 details the problem to solve and introduces notation. Section 3 designs the schemes to select the most relevant variables for the prediction. Section 4 describes the ML architectures and assesses their delay prediction performance.

2. PROBLEM DESCRIPTION, DATASET, AND DELAY-SIGNAL GENERATION

Given a planned (departure or arrival) time, the delay of a flight is defined as the difference between the originally planned time and the actual time when the event happens, namely the departure or the arrival. Air traffic delay can be due to different causes that may be grouped in five categories (US Department of Transportation, 2020) corresponding to delays caused by: a) the airline carrier, or circumstances under the control of the airline; b) elements in the National Aerospace System (NAS) (airport operations, high traffic, etc.); c) other flights arriving late and thus making the plane that must operate the flight unavailable; d) severe weather conditions; and e) security reasons. Clearly, the effects of a particular delay have impacts beyond the time and element for which the delay first takes place and, hence, the entire network must be considered.

The air transportation network may be regarded as a system composed of different elements, including airports and OD pairs. We will use E to denote the set of elements of interest (i.e., either the set of airports or the set of OD pairs), e to denote a generic element of E, and G to denote a graph capturing the connections between the elements in E. Rather than predicting the delay of a particular flight, our focus is on predicting the aggregated (average) delay experienced by a particular network element e (airport or OD pair) and a particular time t'. To be more specific, let $d(e,t)$ denote the delay that element e is experiencing at time t. Then, we aim at designing ML architectures that, after properly trained, at time t use as input global network information and the values of $d(e,\tau)$ for $\tau \leq t$ to predict if $d(e,t')$, the delay of element e at time $t'=t+h$, will be greater than a prespecified threshold δ. Hence, $h>0$ can be viewed as the horizon of the prediction (typically between 2 and 6 hours) and $d(e,t')\geq\delta$ (with the delay threshold δ typically set to 60 minutes) as the condition to trigger the delay alarm.

Note that $d(e,t)$, which constitutes the essential piece of information of our model, can be viewed as a spatiotemporal signal that varies both across time and across the nodes of the network graph G. Regarding the construction of G, when dealing with the "airports graph", we consider two nodes to be connected by an edge if there is any flight between them. When the nodes of G represent OD pairs, we use as graph G the line graph associated with the graph of airports; see Harary (1972) for details on the definition of a line graph. From this "delay-signal" point of view, one can reasonably expect $d(e,t_1)$ and $d(e,t_2)$ to be close if the difference between t_1 and t_2 is small, and $d(e_1,t)$ and $d(e_2,t)$ to be related if e_1 and e_2 are neighbouring nodes. Last but not least, we will handle (and predict) arrival and departure delays separately, since their causes can be different. However, to simplify the discussion and notation, in the following sections we will generically refer to a single type of delay.

Dataset: To train and test our architectures we use a database provided by the Bureau of Transportation Statistics from the US Department of Transportation, whose website publishes data for different means of transport in the US. Our focus is on the database for domestic flights, whose entries provide information for individual flights, including

scheduled and actual departure and arrival times, origin and destination airports, delays and their categories, deviations, and cancellations. For 2018 only, this database features more than 6,000 OD pairs, 358 airports, and more than 7 million flights.

Pre-processing: One-third of the OD pairs show a frequency lower than 1 flight per day. These samples add computational complexity, without providing meaningful (generalizable) knowledge for a data-based delay-prediction approach, hence we only consider airports and OD pairs featuring more than 10 flights per day on average. Also, we remove cancelled flights, as we consider that they do not exhibit a causal relationship with the considered variables (departure and arrival times, delay, etc.).

Definition of the state signal *d*(*e*,*t*): Time is slotted into one-hour intervals, with *t* denoting the slot index. Inspired by Rebollo (2012), the process to obtain *d(e,t)* is as follows. For each hour *t* (say 3PM of Today), we consider a 120-minutes window that considers the previous and current hour (i.e., from 2:00PM to 3:59PM). Furthermore, let $F_{(e,t)}$ denote the subset of flights that took place within the 120-minutes window around *t* and involved element *e*. Then, the value of *d*(*e*,*t*) is simply defined as the average of the delay experienced by the flights in $F_{(e,t)}$. Following this procedure, we obtain a first version of the signal *d*(*e*,*t*) for all *e* and *t*. However, there are instances of (*e*,*t*) for which there was no flight within the considered two-hour interval (i.e., $F_{(e,t)}$ was empty) and, hence, the value of *d*(*e*,*t*) is not defined. To deal with those, we implemented the following approach: if the number of consecutive hours where *d*(*e*,*t*) is not defined is less than six, then we set *d*(*e*,*t*) using linear interpolation based on the values of *d*(*e*,*t'*) for $|t-t'| < 6$. If the number of consecutive empty windows is larger than six, we do not consider *d*(*e*,*t'*) with $|t-t'| \geq 6$ to be informative and set *d*(*e*,*t*) to zero, which is considered as the default delay value.

3. DESIGNING THE INPUT VARIABLES

This section details the design and selection of variables that will be used as input for our classification algorithms. This is an important issue oftentimes overlooked. For a data-based approach to succeed in the context of air-traffic delay prediction, a balance between the number of data points (samples) in the dataset and the dimensionality and complexity of the input variables needs to be reached. ML architectures that use the raw data as input will require many more training samples than those that use a reduced dimensionality (possibly quantized) representation of the data where relevant knowledge has been extracted. We split this design into two steps. Firstly, we extract (create) new variables from the dataset. Secondly, we implement a feature selection scheme to identify the subset of the most relevant variables for the prediction problem.

The input variables are split into two groups: time variables (hour, day, month, and year) and network delay-state variables (aggregating/encapsulating delays of one or several *(e,t)* pairs). When running a descriptive analysis of the data, we observe a seasonality pattern on

the delays. For that reason, we generated a new categorical variable ***"season"***, which takes three values: low (September to November), medium (January to May) and high (June to August, and December). Similarly, to account for the congestion of the overall network we define two new variables: one accounting for the network delay in the last hour and another one accounting for the network delay in the last 24 hours. Rather than using an average across elements of the network, we rely on a (K-Means) clustering algorithm to split the delay levels into (six) different groups and use those to define a ***"network delay congestion index"*** for each hour and day. Lastly, when modelling the delay in an element *e*, it is reasonable to assume that the delay experienced by the neighbours of *e* is relevant. Since considering the delay of each of the neighbours can lead to a very high dimensional signal, we define a continuous variable that, for every (*e,t*) pair, aggregates (sums) the value of the delays *d*(*e',t*) for all elements *e'* that, according to the graph *G*, are neighbours of *e*. Moreover, since the relation between delays of neighbouring nodes seems to depend strongly on the particular cause generating the delay, we define 5 different ***"neighbour-aggregated delay variables"***, each of them associated with one of the 5 causes of delay listed at the beginning of Section 2.

The information of the dataset with the addition of the variables described above yields more than a thousand input features (variables) for our classifier. Since this large number challenges learning and incurs a high computational-complexity cost, we put forth a feature-selection mechanism to identify the most informative variables. Two options are considered: one based on RF techniques and one based on lasso-regularized logistic regression (LR); see (Hastie et al., 2009). While not implemented in the context of air-traffic delay prediction, these two feature-selection techniques have been recently used in a number of ML approaches. The variable selection is carried out based on a "level of information" that depends on the effect/usefulness of the variable on the delay to be predicted. This guarantees that the fact of the goal being delay prediction is accounted for when running the proposed feature-selection mechanism.

The selection is run only for network-delay state variables (time variables are deemed informative, and their number is small, hence, they are preserved); it is carried out separately for each element of the network (delays seem to depend heavily on the airport, so we do not want to force the selected variables to be the same across the entire network); and it proceeds in two steps. Firstly, we run a single-shot selection step, where we reduce the number of features to 100. Secondly, we further reduce those 100 features to 10. That is achieved by running *I=10* different instances of the feature selection algorithm. Each of those instances identifies the 10 most important variables, and the ones we finally select correspond to the 10 most important ones obtained after averaging the results across the *I=10* instances.

As an illustration of the feature selection results, Figure 1 shows the selected variables by the two algorithms for the "O'Hare (ORD) to LaGuardia (LGA)" OD pair, which is the route with the highest number of flights. As we can see, the two methods assign the highest

importance value to the NAS delay in the neighbours. They also assign a high importance to the current departure delay of the ORD-LGA pair (the one under analysis) and LGA-ORD. The arrival delay variable in ORD airport is also selected by both algorithms, with the remaining ones being different. Analysing these differences is certainly meaningful but beyond the scope and page limits of this paper.

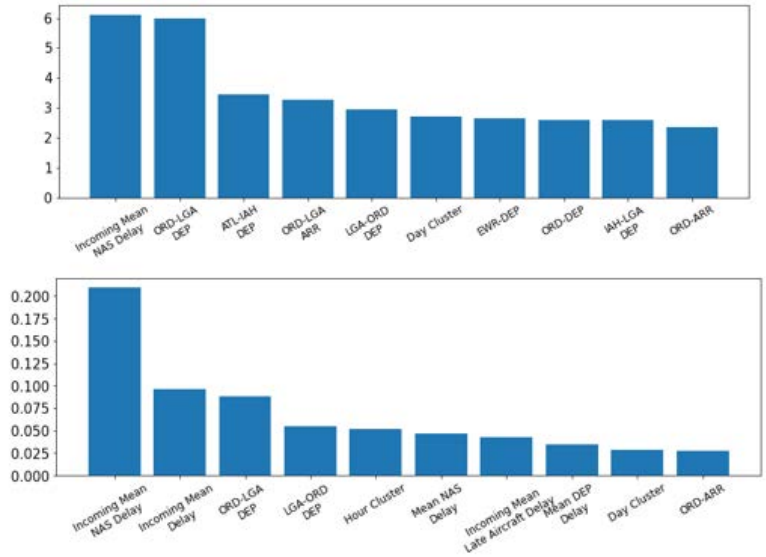

Figure 1: Features selected by the RF (top) and LR (bottom) schemes. The x-axis represents the selected features and the y-axis the "information level" given by the scheme.

4. ML ARCHITECTURES, TRAINING, AND TESTING RESULTS

Let x and y denote, respectively, the input and output variables of our classifier. The input x is a vector with as many dimensions as input features and y is a binary $\{0,1\}$ number. Suppose that: a) $\mathcal{T} = \{x^s, y^s\}_{s=1}^{S}$, where s denotes a sample index, is the set of training samples and b) $f(x|\theta) \rightarrow y$ is the input-output mapping implemented by a (nonlinear) parametric ML architecture whose parameters are collected in θ. The approach in a supervised ML setting is to use $\mathcal{T}$ to estimate $\hat{\theta}$, the value of θ, and then use the (learned) function $f(\cdot|\hat{\theta})$ to estimate the output associated with inputs x not included in $\mathcal{T}$ as $\hat{y} = f(x|\hat{\theta})$.

Recall that, at time t, our goal is to predict if $d(e,t+h)$, the delay of element e at time $t'=t+h$, will be greater than a prespecified threshold δ. The approach that we put forth to achieve that is to: i) train a different classifier per element e (so that the parameters can depend on the airport or OD-pair); ii) use as input the features described in Section 3; and iii) try a range of architectures defined in more detail soon. For the results presented in this paper, we set the prediction horizon to $h=2$ hours (so that time correlation is likely to exist, and companies have sufficient time to take mitigating actions) and the delay threshold to $\delta=60$ minutes (so that the incurred delay is meaningful). By recasting the prediction of a continuous variable

as the prediction of a (quantized) binary version of it, the complexity of the output is reduced, rendering data-based learning more likely. The main drawback associated with this approach is that, for most (e,t) pairs, the associated output label is zero. That is, for the vast majority of the samples the network is not saturated and the continuous delay is less than 60 minutes.

The next step is to describe how the training of the ML architectures is carried out. The two key aspects are i) the definition of the training set and ii) how to deal with the unbalancing of the labelled data. Regarding the first aspect, we use the data of the US domestic market from the year 2018 to train, and test the results using the data from January 2019. More specifically, if the focus is on predicting the delay for the element e, we: i) select the samples of the dataset related to e; ii) compute the features introduced in Section 3; iii) associate each input sample with the corresponding value $d(e,t+h)$; iv) remove all the samples for which $d(e,t+h)$ is exactly 0 minutes; and v) compute the binary label y associated with the input by checking if $d(e,t+h)\geq\delta$. The reason for iv) is threefold: 1) many of those samples correspond to cases where $d(e,t+h)$ was not defined and set to zero, 2) if $d(e,t+h)$ was an actual measurement, an exactly zero value represents a stress-free scenario were predictable 1-hour delays are very unlikely to take place, and 3) it contributes to balance the dataset, reducing the number of samples whose label is $y=0$. Indeed, the second aspect to be handled is the lack of balance in the data. A descriptive analysis of the dataset reveals that 97% of the samples satisfy $d(e,t)<\delta$, so that the associated label y is zero. This is a significant challenge because it means that a (useless) classifier that always predicts zero and misses all the delays would have an accuracy of 97%, likely beating all other classifiers that try to find a balance between false alarm and miss detection errors. Indeed, if accuracy is the metric chosen to train the ML architectures and the original data is used as the training set, then the optimal ML classifiers always yield a zero prediction. To bypass this problem, we use an *undersampling and oversampling balancing approach* during the training phase (Chawla, 2010). More specifically, we train the ML architectures using 3,000 samples but, rather than choosing them uniformly at random from $\mathcal{T}$, which is the standard approach in balanced setups and in our case would yield 90 samples with $y=1$ and 2,910 with $y=0$ on average, we pick the samples as follows. We split $\mathcal{T}$ into two subsets, from the subset with $y=0$ we pick 1,500 samples uniformly at random and from the subset with $y=1$ (which contains less than 1,500 samples), we build a larger set with synthetic copies of the original samples (to obtain a set with more than 1,500 samples) and then pick 1,500 of those "original and synthetic" samples with $y=1$ uniformly at random. This *undersampling and oversampling* approach balances the data and exposes the classifiers to a large number of samples where delay took place, so that they are able to learn the patterns in the input data that are related to those events. To render the learning process more robust and less sensitive to the subset design, we construct 10 instances of $\mathcal{T}$, train the classifiers for each instance, and average the results.

ML architectures: To tackle the classification task, we consider eight different ML architectures. Six of them correspond to "traditional" classification algorithms with an increasing level of complexity: LR, DT, RF, K-Nearest Neighbours (KNN), Gradient

Boosting Trees (GBT), and Multi-Layer Perceptrons (MLP) (Hastie et al., 2009). The other two are more advanced classification architectures that account explicitly for the domain structure of the delay signal $d(e,t)$, either its time structure (using an LSTM-based classifier) or its graph structure (GCNN-based classifier).

Prediction results over the testing dataset: Once the architectures have been trained and the corresponding (element-and-architecture dependent) parameters $\hat{\theta}$ have been learned, we assess the classification performance using the data on the test set (January 2019). Let N_{ab} be the number of samples where the predicted output is $\hat{y} = a$ and the actual label was $y = b$, and note that, for binary classification, the cardinality of $\{N_{ab}\}$ is 4. Based on those values, we report the classification performance of each of the trained architectures using the following metrics: accuracy $Acc=(N_{00}+N_{11})/(N_{00}+N_{01}+N_{10}+N_{11})$, precision $Prec=N_{11}/(N_{11}+N_{10})$, recall $Rec=N_{11}/(N_{11}+N_{01})$, and f1-score $F1=2\cdot Prec\cdot Rec/(Prec+Rec)$.

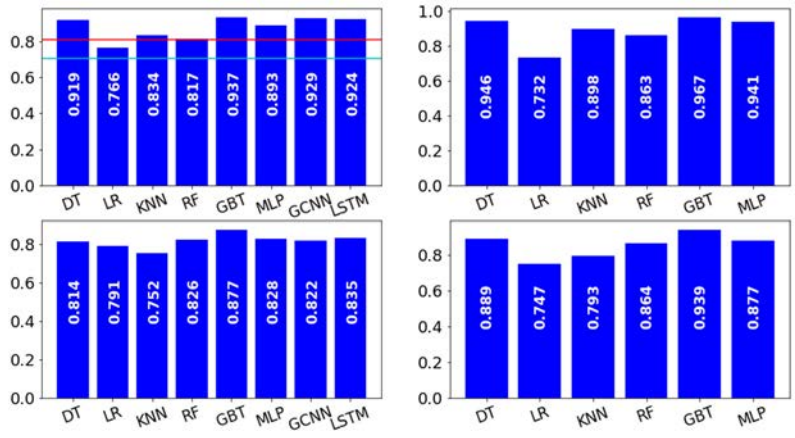

Figure 2: Comparison of the accuracy for OD Pairs (on the left) and airports (on the right), using balanced (on the top) and unbalanced (on the bottom) data.

Figure 2 shows the classification accuracy of our ML architectures both for the prediction of delays on OD-pairs (two panels on the left) and airport delays (two on the right), using balanced (top) and unbalanced (bottom) testing data. To facilitate comparisons with the state of the art, the results in Figure 2 only consider the 100 most delayed airports and the figures correspond to the averages across all the elements in the network. In addition to the results of our architecture, the top-left panel shows two horizontal lines: the red line corresponds to the accuracy of the approach in Rebollo & Balakrishnan (2014) and the blue one to that in Gopalakrishnan & Balakrishnan (2017). To keep the comparison fair, these two lines are not shown in the other panels because the state of the art addressed neither predicting airport delays nor testing on unbalanced data. The main observations are as follows: 1) The accuracy levels are relatively high but, as expected, below the 97% level achieved by the dummy all-zero classifier. 2) Our architectures (7 out of 8) beat the state of the art, in some cases by more than 10%. Since some of our classifiers are fairly simple, this outperformance confirms the importance of the pre-processing steps. 3) Predicting delays at airports is easier than at

OD pairs. 4) The results deteriorate when (actual) unbalanced testing sets are used. 5) Classifiers with a higher degree of sophistication seem to give rise to better results at the price of a higher computational and training complexity (which is the reason for not reporting the results of the GCNN and LSTM algorithms in plots on the right).

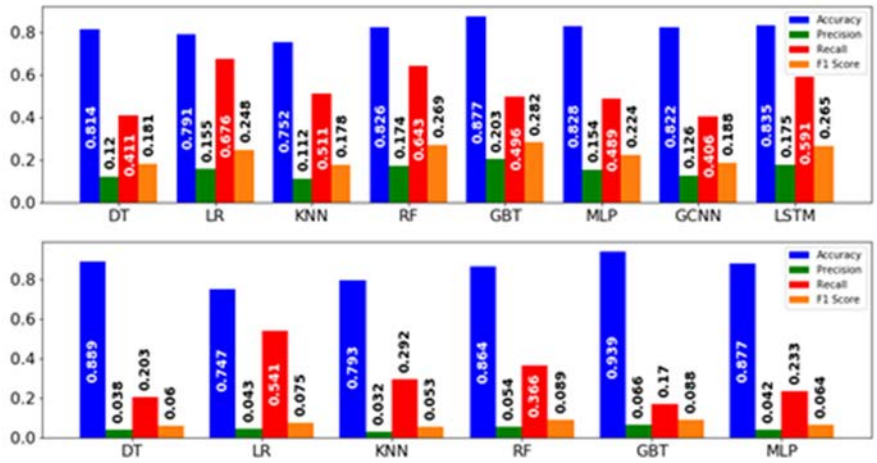

Figure 3: Accuracy, precision, recall, and f1-score for OD pairs (top) and airports (bottom) using unbalanced test data.

The reason for reporting accuracy was to establish a fair comparison with the state of the art. However, as already discussed, in unbalanced classification setups, accuracy must be complemented with precision, recall and f1-score metrics, with the last one being typically used as the main figure of merit to analyse. Figure 3 reports the four metrics for OD pairs (top) and airports (bottom) for our architectures using unbalanced test data. We observe that the results are noticeably worse, confirming that: a) most of the accuracy observed was due to the "trivial" prediction of non-delay scenarios and b) as expected, early prediction of a significant percentage of future delays based on present and past data is a challenging task. Interestingly, the results also show that better precision/recall/f1-score are obtained for OD pairs, suggesting that the higher accuracy in predicting delays at the airport level was associated with a larger number of zero labels. Regarding the ML architectures, the highest recall is achieved by the simple LR algorithm. However, the highest f1-score (which is typically viewed as the most important classification performance indicator) is achieved by the GBT, with RF and LSTM also showing a good performance.

To gain further insights, Figure 4 reports classification results in two additional scenarios dealing with OD-pairs and unbalanced test data. In the first scenario (top panel) we constrain the prediction to the 25 busiest OD-pairs (largest number of flights). The motivation here is to focus on elements with more data and, also, a higher likelihood of a delay, to check if under those conditions, learning from past and previous data is more feasible. The second one corresponds to a classifier that proceeds in two steps. In the first step if $d(e,t)=0$ the "deterministic" prediction is that $d(e,t')<\delta$ and, hence, that $y=0$. If $d(e,t)>0$, we then use the output of the ML classifier as the predicted y. Note that, under this approach, $\mathcal{T}$ needs to be

modified accordingly, so that all the training samples whose input contains $d(e,t)=0$ are removed. The idea behind this scenario is to identify the "trivial cases" first and then let the classifier focus on the non-trivial ones. The results corroborate that in both cases, the accuracy and, most importantly, the f1-score improve, suggesting that our intuition was correct and motivating further research of "hierarchical" classifiers that focus on the elements where delay is more likely and more data exists.

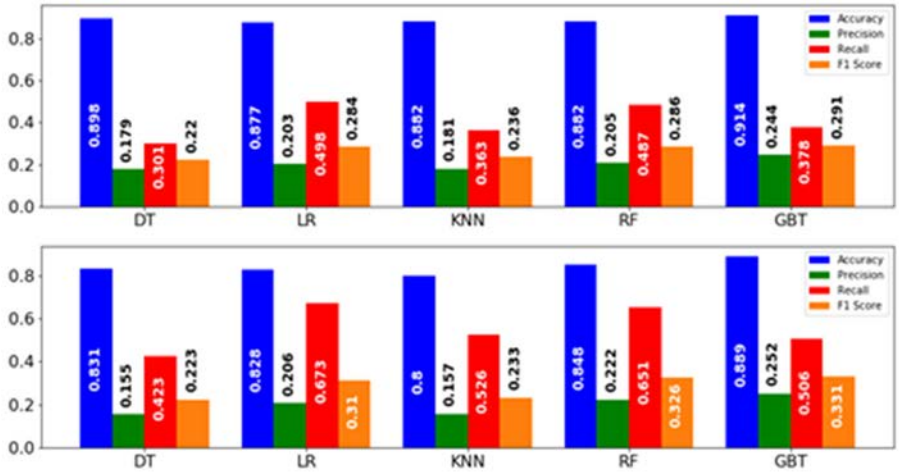

Figure 4: Accuracy, precision, recall, and f1-score for OD pairs using unbalanced test data for two additional scenarios (top and bottom). See the main text for additional details.

5. CONCLUSIONS

We put forth a data-based processing and learning pipeline to predict the delays in an air transportation network. Key novelties of our approach included the prediction of the delay not only across OD pairs but also across airports, the consideration of feature selection schemes, accounting for the (graph) structure of the network and the correlation of the delays across time, and the incorporation of rebalancing schemes to foster recall and precision performance. Our schemes were trained and tested using real data from the US domestic market in 2018 and 2019. The results outperformed the state of the art by more than 10% and confirmed that air-traffic delay is an inherently complex problem. Future research includes the prediction of delays for individual flights along with the implementation of network-segmentation schemes that enable tailoring the prediction to segments of the air transportation network that exhibit related delay dynamics.

ACKNOWLEDGEMENTS

The authors want to thank Sergio Rozada for his help in designing and coding the algorithms used in this publication and the Spanish "Agencia Estatal de Investigación" for the support via Project Grants TRA2016-76914-C3-3-P, PID2019-105032GB-I00, and CAS19/00036 and CAM grant PEJ-2020-AI/TIC-18964.

REFERENCES

AIRLINES FOR AMERICA (2019), U.S. Passenger Carrier Delay Costs. https://www.airlines.org/dataset/per-minute-cost-of-delays-to-u-s-airlines/#

CHAWLA, N. (2010), Data Mining For Imbalanced Datasets: An Overview, Data Mining and Knowledge Discovery Handbook, Springer (pages 875–886)

GAMA, F., MARQUES A. G., LEUS, G. & RIBEIRO, A. (2019), Convolutional neural network architectures for signals supported on graphs, IEEE Transactions on Signal Processing, vol. 67, pp. 1034-49.

GERON, A. (2017), Hands-On Machine Learning with Scikit-Learn and Tensorflow. O'Reilly Media.

GOPALAKRISHNAN, K. & BALAKRISHNAN, H. (2017), A Comparative Analysis of Models for Predicting Delays in Air Traffic Networks, Air Traffic Management Research and Development Seminar.

HANLEY Z.J. (2015), Delay Characterization and Prediction in Major U.S. Airline Networks. Master's thesis, Massachusetts Institute of Technology.

HARARY F. (1972), Graph Theory. Addison-Wesley.

HASTIE, T., TIBSHIRANI, R. & FRIEDMAN, J. (2009), The elements of statistical learning: data mining, inference, and prediction, Springer Science & Business Media.

HOCHREITER, S. & SCHMIDHUBER J. (1997), Long short-term memory, Neural computation, vol. 9, pp.1735-1780.

MULLIGAN, J. (2019), Europe's flight delays cost €100 a minute, Independent.ie, https://www.independent.ie/business/world/europes-flight-delays-cost-100-a-minute-37934906.html

THE INTERNATIONAL AIR TRANSPORT ASSOCIATION (IATA) (2020), IATA's Annual Review. https://www.iata.org/contentassets/c81222d96c9a4e0bb4ff6ced0126f0bb/iata-annual-review-2019.pdf

REBOLLO, J.J. (2012), Characterization and prediction of air traffic delays,". Master's thesis, Massachusetts Institute of Technology.

REBOLLO J.J. & BALAKRISHNAN H. (2014), Characterization and prediction of air traffic delays, Transportation Research Part C: Emerging Technologies, vol. 44, p. 231-241.

UNITED STATES DEPARTMENT OF TRANSPORTATION: BUREAU OF TRANSPORTATION STATISTICS (2020), Understanding the Reporting of Causes of Flight Delays and Cancellations. https://www.bts.gov/topics/airlines-and-airports/understanding-reporting-causes-flight-delays-and-cancellations