# Predicting Flight Arrival Times with a Multistage Model

Gábor Takács
*Department of Mathematics and Computer Science*
*Széchenyi István University*
*Győr, Hungary*
*e-mail: gtakacs@sze.hu*

***Abstract*—Airlines are constantly looking for ways to cut flight delays, in order to enhance service quality and reduce operational costs. The goal of the data science contest, GE Flight Quest (https://www.gequest.com/c/flight), was to make flights more efficient by improving the accuracy of arrival time estimates. The data set of the contest was 128 GB in size and contained 252 data columns arranged in 34 tables. This paper presents my solution that won third prize under team name *Taki*. The solution employs a 6-stage model consisting of successive ridge regressions and gradient boosting machines, built on 56 features constructed from the raw data. The hardware environment used for training and running the model was a 64 core machine with 1 terabyte of memory.**

***Keywords*-GE Flight Quest, ridge regression, gradient boosting machine, parallelization.**

## I. Introduction

Operating an air transportation system efficiently is a highly complex problem. Changes in factors like in weather, flight patterns, or bottlenecks can lead to delays that are inconvenient for the passengers and costly for the airlines. Airlines are constantly looking for ways to reduce flight delays. Decreasing the average delay per flight by one minute could save millions of dollars in annual crew costs and fuel savings for a mid-sized airline [1].

The goal of the GE Flight Quest contest (http://www.gequest.com/c/flight) was to make flights more efficient, building on advancements in real-time big data analysis. The contest was set up on Kaggle in December 2012 with a prize pool of 250,000 dollars, and the result was announced in April 2013.

This paper describes my solution that won third prize in GE Flight Quest, under team name *Taki*. My approach employs a 6-stage model consisting of successive ridge regressions and gradient boosting machines, built on 56 features constructed from the raw data. The hardware environment used for training and running the model was a 64 core machine with 1 terabyte of memory.

The rest of the paper is organized as follows: Section II. defines the problem and introduces the data set. Section III. describes the proposed solution with implementation details. Section IV. overviews related work. Section V. contains experimental results and discussion. Section VI. concludes the paper.

## II. The Problem

The goal of the competition was to predict the runway and the gate arrival time of en route U.S. domestic flights as accurately as possible, based on flight history, weather, air traffic control, and other data available at a given time moment. The estimates had to be given as the number of integer minutes elapsed since 00:00 on the day of the flight. The accuracy was evaluated on a test set, in terms of a root mean squared error (RMSE) based metric.

The runway RMSE of a prediction was measured as

$$\mathrm{RMSE}_{\mathrm{runway}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{t}^{(i)}_{\mathrm{runway}} - t^{(i)}_{\mathrm{runway}}\right)^2},$$

where $\hat{t}^{(i)}_{\mathrm{runway}}$ and $t^{(i)}_{\mathrm{runway}}$ are the predicted and true runway arrival time associated with the $i$th test flight and $n$ is the number of test flights. $\mathrm{RMSE}_{\mathrm{gate}}$ is defined like $\mathrm{RMSE}_{\mathrm{runway}}$, but for gate arrival times.

The evaluation metric used in the competition was a weighted average of the runway and the gate RMSE:

$$\mathrm{RMSE} = \frac{1}{4}\mathrm{RMSE}_{\mathrm{runway}} + \frac{3}{4}\mathrm{RMSE}_{\mathrm{gate}}.$$

The weights indicate that the subtask of gate arrival prediction was considered as more important than runway arrival prediction. The two subtasks are closely related, since runway arrival typically precedes gate arrival by a few minutes.

### A. The Data Set

The data set of GE Flight Quest was 128 GB in size (in uncompressed CSV format), and it contained 252 data columns arranged in 34 tables. The time span of the data set was 109 days, from 2012-11-12 to 2013-02-28. Two test sets were allocated: the Public Leaderboard Set (PLS, 14 days of data from 2012-11-26) and the Final Evaluation Set (FES, last 14 days of data). The remaining data was split into 3 partitions, namely Initial Training Set (ITS), Augmented Training Set 1 and 2 (ATS1, ATS2). There was no data available for 8 days before the FES.

For the test sets, a random cutoff time was drawn for each day, and the data was available only up to the cutoff time.

To predict a test flight, the contestants could use the training set, and the day of the test set that the flight belonged to.

The tables of the data set can be arranged into groups:

- **FlightHistory**: The 2 tables of this group contained direct information about the flights such as departure and arrival location and time.
- **ASDI** (Aircraft Situational Display to Industry): The 7 tables of this group contained information about the planned and actual travel path of the flights.
- **ATSCC** (Air Traffic Control System Command Center): The 8 tables of this group contained data about air traffic control events such as ground delay programs.
- **Metar**: The 4 tables of this group contained actual weather data, originating from the Metar weather reporting system.
- **OtherWeather**: The 12 tables of this group contained weather forecast data, originating from multiple sources.

Most parts of the data was extracted from the FlightStats flight tracking system. The data set contains 2,390,185 U.S. domestic flights, 26,257,353 flight plans, 40,606,359 flight events, 1,156 airports, 229 airlines, and 5,434 weather report stations in total. The tables form a "real life like" relational database with complex relationship between the tables, and a significant amount of missing and noisy data. Most of the records in the data set are events, associated with an entity (like flight, airport, weather report station) and a time stamp.

### B. Data Columns

This subsection will try to give a picture on the actual data columns contained in the tables. Obviously, it would make no sense to describe all 252 columns here, therefore a subset was selected from them.

The *FlightHistory* table of the FlightHistory group contained among others the airline, the aircraft, the flight number, the arrival and departure airport associated with the flight, the scheduled and actual departure and arrival time of the flight. The departure and arrival times and their estimates were given in integer minutes.

Note that the actual runway and gate arrival times are the variables that had to be predicted. Figure 1 shows the empirical distribution of the difference between gate and runway arrival time.

It can be seen that most frequently, the gate arrival follows the runtime arrival by 4 minutes. The difference between the recorded gate and runway arrival time can be sometimes negative which indicates erroneous data entry. In some rare cases (not shown in the figure) it takes hundreds of minutes for the flight to get from the runway to the gate.

The *FlightHistoryEvents* table of the FlightHistory group contained an event log for each flight. Events include updating the estimated runway or gate arrival time, the arrival terminal or gate or recording the actual takeoff time. The FlightHistoryEvents table contained more recent information about arrival times than the FlightHistory table because its estimates were updated regularly, while the estimates of FlightHistory were filled in before take-off. The average number of events per flight was $\sim$17. My solution used only the most recent estimates contained in FlightHistoryEvents for predicting arrival times.

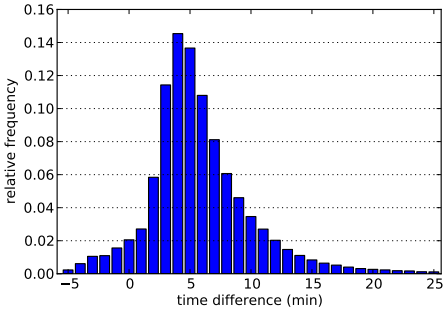


Figure 1. The distribution of gate minus runway arrival time.

A rare event that tends to result delays is divertion from the originally planned destination to another one. Predicting the arrival time of flights that have a "diverted" event in their history is an extraordinarily hard task that needs special treatment. Such flights were excluded from the Final Evaluation Set of the competition, therefore they were also excluded them from training and internal validation.

The *ASDIFlightPlan* table was the most informative data source in the ASDI group. It contained data on the flight plans filed for each flight. One flight may have many corresponding flight plans, as any change to the plan is represented as a new plan. The average number of flight plans per flight was $\sim$11. The flight plan contains an estimate on the runway arrival time (but not on the gate arrival time). Only the most recent flight plan of each flight was used in my model.

The *ASDIPosition* table from the ASDI group contained information about the actual geographic position of the flights, along with altitude and ground speed values. The geographic latitude and longitude coordinates were given with two digits precision which corresponds to about one kilometer precision. The position of a flight was typically updated about once per minute.

The ATSCC group contained data about air traffic control events such as delays, de-icing events and ground delay programs. An air traffic control event is usually associated with one or more airports and a given time interval. Air traffic control events are rare, but they can strongly affect arrival times.

The Metar group contained information about the actual weather at 5,434 weather report stations at given times. The most important weather features (according to my model) were precipitation conditions, visibility, wind speed and temperature.

About 92 % of the airports were equipped with an own weather report station. For such airports it was easy to extract the weather from the Metar tables. It is much more difficult to infer the weather at the actual location of an en route flight, since (a) it is not trivial how the measurements of the nearby weather report stations should be combined and (b) the altitude of the flight should also be taken into account. My solution used only the destination airport's weather for predicting the arrival times of a flight.

## III. The Proposed Solution

The applied data splitting scheme was quite complex, since it tried to match the special rules of the competition (cutoff times, prohibited days). The top level elements in my scheme are called configurations. A configuration contains three distinct time intervals:

- The *training period* is used to extract aggregate information about the entities (e.g. average taxi time per airport). No cutoff times are used.
- The *probe period* is used to extract per flight features, and to internally validate the model. A cutoff time is drawn for each day, and the data is kept only up to the cutoff time.
- The *test period* is the interval the model should give prediction for.

In the first phase of the competition, the test set was the Public Leaderboard Set. In the second phase it was the Final Evaluation Set. In both phases, I defined multiple configurations having the same test and training period, but different probe periods. Predictions were generated for the test set with each of the configurations, and the answers were averaged. The reason of using multiple configurations was to reduce time and memory requirement.

I used 3 configurations, $C_1$, $C_2$ and $C_3$, depicted on Figure 2 to create the final submission. In each configuration, a 10-fold cross-validation scheme was applied. This means that the probe set was randomly partitioned to 10 parts. In each fold, the training set and 9 parts of the probe set was used for training, and 1 part of the probe set was used for validation.

To generate prediction for the test set, I did *not* train a new model on the training set plus the full probe set. Instead of this, I took the models that were built during the cross-validation, and averaged their answers, since this solution ensures that only quality-checked models are involved in generating a submission.

In the initial phase of the competition, I generated the examples of the probe set the same way as the official test sets were generated. Later it turned out that instead of drawing a cutoff time for each day and taking the flights that are en route at that time, it is more efficient to draw cutoff times for the flights individually. This way, much more flights could be included in the probe set and more accurate models could be built.

My solution did not use every flight to generate a probe example. Flights that had a "diverted" or "redirected" event in their history, flights with unknown departure or arrival times, and flights with extreme (e.g. negative) flight duration were removed. After this outlier removal process, the probe set contained 253,852 examples in configuration $C_1$, 387,558 in $C_2$ and 214,264 in $C_3$.

### A. Features

This section describes the 56 features that were used in the final model (both for gate and runtime arrival prediction). The features can be arranged into groups. Every time stamp and duration is measured in minutes. The phrase "most recent $x$" means the most recent value of $x$ before the cutoff time corresponding to the given flight. Feature types are marked with the following symbols: ∘ indicates a continuous feature, † a binary feature, ◇ a single-valued categorical feature and ⋆ a multi-valued categorical feature. The operation $x \times y$ means the direct product ("concatenation") of categorical features $x$ and $y$.

*1) Main Features:*

∘ **fhx/era7**: Cleaned version of $t_{\mathrm{ERA}} - t_{\mathrm{cutoff}}$, where $t_{\mathrm{ERA}}$ is the most recent estimated runway arrival time in the flight history events, and $t_{\mathrm{cutoff}}$ is the cutoff time associated with the flight. The cleaning consists of filling in nulls and clipping the value into $[0, 400]$.

∘ **fhx/ega7**: Cleaned version of $t_{\mathrm{EGA}} - t_{\mathrm{cutoff}}$, where $t_{\mathrm{EGA}}$ is the most recent estimated gate arrival time in the flight history events.

∘ **fhx/eat7**: Cleaned version of $t_{\mathrm{ARD}} + (t_{\mathrm{ED}} - t_{\mathrm{EA}}) - t_{\mathrm{cutoff}}$, where $t_{\mathrm{ARD}}$ is the actual runway departure time of the flight and $t_{\mathrm{ED}}$ / $t_{\mathrm{EA}}$ are the estimated departure / arrival times contained in the most recent flight plan.

∘ **fhx/sat7**: Cleaned version of $t_{\mathrm{ARD}} + t_{\mathrm{SAT}} - t_{\mathrm{cutoff}}$, where $t_{\mathrm{SAT}}$ is the scheduled air time of the flight.

∘ **fhx/sbt7**: Cleaned version of $t_{\mathrm{ARD}} + t_{\mathrm{SBT}} - t_{\mathrm{cutoff}}$, where $t_{\mathrm{SBT}}$ is the scheduled block time of the flight.

∘ **fhx/ard7**: $t_{\mathrm{cutoff}} - t_{\mathrm{ARD}}$, clipped into $[0, 400]$.

*2) GroupBy Features:*

∘ **gb/gd3_ap**: Average gate delay (actual gate arrival time minus scheduled gate arrival time) at the destination airport, computed on the training set.

∘ **gb/tt3_ga**: Average taxi time for flights that arrive at the same airport and gate as the current flight, computed on the training set.

*3) ATSCC Features:*

† **atscc/grdelay**: Indicates if there is an ATSCC ground delay scheduled at the destination airport for the scheduled runway arrival time of the flight.

† **atscc/delay2**: Indicates if there is any ATSCC delay scheduled at the destination airport for the day of the flight.

*4) Metar Features:*

⋆ **metar/@pcs2**: This binary vector valued feature was extracted from the present conditions part of the Metar data. Its $j$th component is 1, if the $j$th condition (e.g. Light Snow) occurred at the destination airport at any time on the day of the flight.

∘ **metar/ws2**: Average Metar wind speed at the destination airport on the day of the flight.

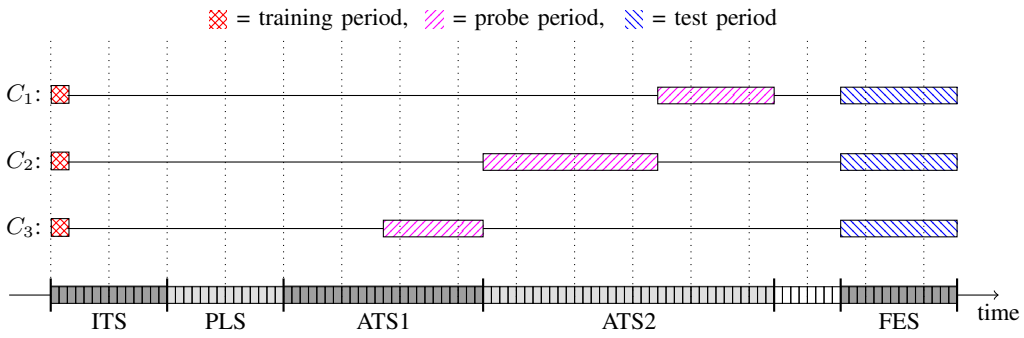


Figure 2. Configurations involved in generating the final submission.

- **metar/{wg, temp, vis}**: The most recent Metar wind gusts / temperature / visibility value at the destination airport.

*5) Sparse Features:*

- **fh/aac**: Arrival airport code from the FlightHistory table.
- **fh/alc**: Airline code from the FlightHistory table.
- **fh/aach**: Arrival airport code from the FlightHistory table × period of day at the destination at the scheduled arrival time.
- **fh/aact**: Arrival airport code from the FlightHistory table × most recent arrival terminal extracted from the FlightHistoryEvents table.
- **fh/aaca**: Arrival airport code × airline code (both from the FlightHistory table).
- ⋆ **fp/@lr**: The $j$th component of this binary vector valued feature is 1, if the most recent legacy route associated with the flight contains the $j$th legacy route station. The legacy route stations are given in the ASDIFlightPlan table.
- **fh/sact**: Scheduled aircraft type from the FlightHistory table.
- **fh/aaic**: Arrival airport ICAO code from FlightHistory.
- **fh/aacg**: Arrival airport code from the FlightHistory table × most recent arrival gate extracted from the FlightHistoryEvents table.
- **fh/edge**: Departure airport code × arrival airport code (both from the FlightHistory table).
- **fh/flight**: Airline code × flight number (both from the FlightHistory table).
- **fp/acid**: Aircraft identifier from the FlightHistory table.
- **fh/aacb**: Arrival airport code from the FlightHistory table × the most recent bag gate extracted from the FlightHistoryEvents table.

*6) Delta Features:*

- **fhx/tdelta1**: The time stamp (receive time) of $t_{\mathrm{EGA}}$ minus the time stamp of $t_{\mathrm{ERA}}$.
- **fhx/{tdelta2, tdelta2}**: The time stamp of the most recent flight plan minus the time stamp of $t_{\mathrm{ERA}}$ / $t_{\mathrm{EGA}}$.
- **fhx/adelta{70, 71, 72, 73, 74, 75, 76, 77, 78, 79}**: fhx/sbt7 – fhx/sat7, fhx/ega7 – fhx/era7, fhx/eat7 – fhx/era7, fhx/sbt7 – fhx/ega7, fhx/ega7 – fhx/eat7, fhx/sbt7 – fhx/eat7, fhx/sbt7 – fhx/era7, fhx/era7 – fhx/sat7, fhx/ega7 – fhx/sat7, fhx/eat7 – fhx/sat7.

Of course, a feature that is a linear combination of existing ones has no predictive value in a linear model. The utility of Delta features in the solution was that they helped to build more accurate non-linear models.

*7) Position Features:*

- **pos/{dist1, dist2, grsp, alt}**: Distance from the origin / distance from the destination / ground speed / altitude at the most recent position of the flight.

*8) Other Features:*

- **fh/cutoff**: Cutoff time minus midnight time.
- **fh/hour2**: Period of day at the destination airport at the scheduled arrival time.
- **fhx/{speed, speed2}**: Estimated average speed between the origin / most recent position and the destination.
- **fhx/speed1**: Average speed between the origin and the most recent position.
- **fhx/dist**: Distance between the origin and the destination.
- **fhx/{lat1, lon1}**: Geographic latitude / longitude of the origin, with nulls replaced to the average value.
- **fhx/{lat2, lon2}**: Geographic latitude / longitude of the destination, with nulls replaced to the average value.

### B. The Multistage Model

My prediction model consists of 6 stages that are trained on the residual of each other. This subsection specifies the regression method (with parameters) and the input features associated with the stages. Let $\mathcal{F}_0$ denote the feature set {fhx/era7, fhx/ega7, fhx/eat7, fhx/sat7, fhx/sbt7, fhx/ard7}. Let $p$* denote the group of features whose name begins with prefix $p$. The details of the stages are as follows:

- **Stage 1**: Baseline stage, outputting fh/cutoff.
- **Stage 2**: A ridge regression (RR) [2] model trained with the feature set $\mathcal{F}_2 = \mathcal{F}_0 \cup$ {gb/gd3_ap, gb/tt3_ga, atscc/grdelay, atscc/delay2, metar/*}. The value of the regularization coefficient was alpha=100.
- **Stage 3**: A gradient boosting machine (GBM) [3] trained with the feature set $\mathcal{F}_3 = \mathcal{F}_0 \cup$ {fhx/tdelta*, fhx/adelta*, pos/*, atscc/grdelay, atscc/delay2, fh/hour2, fhx/speed*, fhx/dist, fhx/lat2, fhx/lon2}. I used the scikit-learn implementation of GBM with parameters max_depth=10, n_estimators=50, learn_rate=0.16 and random_state=42. The target variable of this stage (i.e. the residual of the previous stage) was clipped to $[-10, 10]$ before training.
- **Stage 4**: A RR (alpha=50) trained with the feature set $\mathcal{F}_4 = \mathcal{F}_2 \cup$ {fh/aac, fh/alc, fh/aach, fh/aact, fh/aaca, fp/@lr, fh/sact, fh/aaic, fh/aacg, fh/edge, fh/flight,

fp/acid, fh/aacb}. The target variable of this stage was clipped to $[-20, 20]$ before training.

- **Stage 5**: A GBM (max_depth=10, n_estimators=50, learn_rate=0.16 and random_state=42), trained with the feature set $\mathcal{F}_3$. The target variable of this stage was clipped to $[-10, 10]$ before training.
- **Stage 6**: A GBM (max_depth=7, n_estimators=50, learn_rate=0.24, random_state=42), trained with the feature set $\mathcal{F}_5 = \mathcal{F}_0 \cup$ {fhx/adelta*, pos/*, atscc/grdelay, atscc/delay2, fh/hour2, fhx/lat2, fhx/lon2, fhx/lat1, fhx/lon1, metar/@pcs2}. The target variable of this stage was clipped to $[-10, 10]$ before training. The feature metar/@pcs2 was converted from sparse to dense (since scikit-learn's GBM expects dense input).

The role of Stage 1 is to transform the task from predicting the arrival time relative to midnight to predicting the remaining time to the arrival. Stage 2 is a linear model on a small, informative set of features. The role of Stage 3 is to capture most of the nonlinear effects in the data. Besides GBM, I also experimented with some other nonlinear models, but they performed worse. Random forests could achieve GBM-like accuracy, but only at the cost of significantly larger model size. The role of Stage 4 is to incorporate sparse features into the modeling. Stages 5 and 6 are nonlinear models, trying to squeeze out a bit of additional accuracy.

The presented model structure is a result of a heuristic, hand-run optimization process. The usual workflow was to iteratively define features and then try to incorporate them into the modeling so that they decrease the RMSE as much as possible.

My solution employs the same features and the same stages for the runway and gate arrival prediction task. The only thing that differed in the two cases was the target variable.

### C. Implementation Details

Except some utility Bash scripts, the whole system was implemented in Python. The most important Python modules I used besides the standard library were numpy, scipy, pandas and scikit-learn. For running the code, a 64 core Linux server with 1 terabyte of memory was used. Each core was an Intel Xeon X7560 @ 2.27GHz. The Experiments section will show that the system would still have tolerable running time on a smaller machine.

I approached the problem by defining data processing nodes that depend on each other. Each node expects a set of files as input an produces another set of files as output. I tried to break down the problem to small parts so that a most of the nodes are associated with relatively simple subproblems. An example node is for instance the conversion of raw FlightHistory table from CSV to binary format with parsed date values. Cross-validation folds and independent nodes were run in parallel.

I introduced a little tweak into the scikit-learn 0.12.1 implementation of RR. The original version evaluates the expression $X^T X v$ as $(X^T X)v$ while mine as $X^T(Xv)$, where $X$ is the (sparse) input matrix, and $v$ is an arbitrary vector. The reason why this modification was needed is that the scikit-learn implementation failed to converge with sparse features when the number of examples was large.

## IV. Related Work

The data set of GE Flight Quest was the first publicly available large-scale, heterogeneous data set for flight arrival prediction to the best of my knowledge. Prior works on flight arrival prediction typically focus on specific parts of the problem, for example modeling congestion at individual airports or delay propagation, analyze these subproblems in detail, and experiment with relatively small data sets. On the other hand, the contestants of GE Flight Quest had a short time span ($\sim$2 months) to build their prediction model, and their goal was to predict arrival times as accurately as possible from the available rich data. For the above reasons, the discussion of related work will be split into two parts. First, earlier work will be overviewed, second the approaches of other contestants will be outlined.

### A. Prior Work

One of the early works in the topic is Newell's study on airport capacity and delays [4]. Its purpose was to describe how the capacity of a runway configuration depends upon the operation sequencing strategy, the runway geometry, the instrument flight rules, and and other factors.

The approach of Peterson et al. [5] uses a combination of stochastic and deterministic components to model delay in a network of airports. The stochastic part of the approach is a semi-Markov model of airport capacities, and the deterministic part is a fluid-flow based delay propagation method.

The paper of Beatty et al. [6] proposes the concept of Delay Multiplier (DM) in order to demonstrate the effect of an initial delay on the operating schedule as a whole. Specifically, their process was developed to evaluate changes in the Federal Aviation Administration's Ground Delay Program procedures.

Abdelghany et al. [7] presents a tool for airline schedule recovery during irregular operations caused mostly by bad weather. Their approach employs a rolling horizon modeling framework that integrates a schedule simulation model and a resource assignment optimization model.

Tu et al. [8] develops a model for estimating flight departure delay distributions required by air traffic congestion prediction models. They employ non-parametric methods to model daily and seasonal trends and mixture distribution to estimate the residual errors.

A recent approach to flight delay estimation is the Approximate Network Delays (AND) model [9]. AND employs

an analytical queuing engine to estimate local delays and a domain specific delay propagation algorithm to model network effects. The foundations of AND were laid down in Malone's Ph.D. thesis [10].

### B. Approaches of Other Contestants

Team *Gxav & ** (the winner of the competition) used a mixture of gradient boosting and random forest models to predict gate and runway arrival times. Key to their success was careful feature selection with their final models using only 58 and 84 features for gate and runway arrivals, respectively, from the total 258 features they painstakingly constructed and optimized.

Team *As High As Honor* (2nd place) used a two-step approach that combined the results of a generalized linear model that encoded intuition about important variables with refinements derived from a random forest model.

Team *Sun*'s (4th place) approach to predicting gate and runway arrival times relied on creating a derived data set with new variables encoding information about the aircraft, airport, airway, gate, hour, and flight path times. Important features used in this model include aircraft GPS position, ASDI flight plans and the direction from which airplanes approached airport runways.

*Jacques Kvam*'s (5th place) approach for predicting runway and gate arrival times used gradient boosting for a model using 10,000 trees and a whopping 1,102 features trained on 260,000 flights. Most significant among these included the distance between the final waypoint and the arrival airport. Many weather features were important as well including temporary vertical visibility and wind speed at the arrival airport.

## V. Experiments

This section presents the results of my approach in different settings. The analysis will focus on accuracy and running time. The Public Leaderboard Set will be used for evaluation instead of the Final Evaluation Set, since the ground truth values of the latter are not known.

In the first experiment, I defined a single configuration $C_4$ and measured the accuracy after the 6 stages defined in subsection III. B. The training period was the first day and the probe period was the remaining 13 days of Initial Training Set. 10-fold cross validation was applied, and cutoff times were generated per flight to construct the probe set. The number of flights in the probe set was 242,055.

Table I shows the RMSE values per stage along with the total CPU time (measured in seconds) needed to achieve the results. The time value shown for a given stage includes the CPU time associated with the previous stages, but it does not include the time spent on feature extraction. For easier interpretability, regressor types and public leaderboard positions corresponding to stages are also indicated. The number of entries in the public leaderboard was 179.

Table I
RMSE per stage using a single configuration.

| Stage | Alg. | Time | RMSE | LB Pos. |
|---|---|---|---|---|
| 1 | | 0 | 93.8241 | 165. |
| 2 | RR | 295 | 6.5204 | 23. |
| 3 | GBM | 43,994 | 5.7619 | 9. |
| 4 | RR | 49,711 | 5.5486 | 6. |
| 5 | GBM | 73,481 | 5.5241 | 6. |
| 6 | GBM | **81,905** | **5.5200** | **6.** |

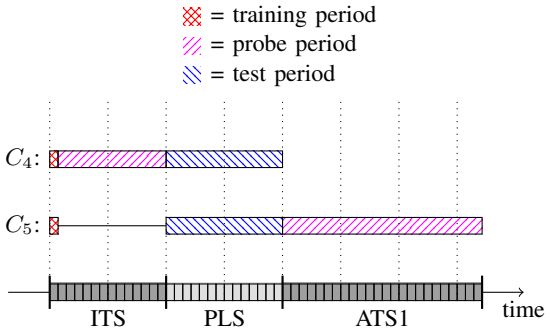


Figure 3. Configurations used for the experiments.

The RMSE of the 6th stage is 5.5200. It requires ∼23 CPU hours to produce this result that would achieve 6th place on the public leaderboard. One can observe that the last 4 stages improve the RMSE of Stage 2 by ∼15 % at the cost of significantly increased running time.

The second experiment investigates the effect of averaging the results of two configurations instead of using a single one. The training period of the second configuration $C_5$ was the same as for the $C_4$ The probe period was the 24 days of Augmented Training Set 1, and the number of flights in the probe set was 445,127. Configurations $C_4$ and $C_5$ are depicted in Figure 3.

Note that using configuration $C_5$ means that future data is involved in predicting the arrival times of the Public Leaderboard Set. This was allowed in the public leaderboard phase of competition, and the contestants did apply it. The results of the second experiment can be seen in Table II.

Table II
RMSE per stage using two configurations.

| Stage | Alg. | Time | RMSE | LB Pos. |
|---|---|---|---|---|
| 1 | | 0 | 93.8241 | 165. |
| 2 | RR | 1,279 | 6.4736 | 21. |
| 3 | GBM | 145,243 | 5.7108 | 7. |
| 4 | RR | 167,355 | 5.4262 | 5. |
| 5 | GBM | 255,610 | 5.3932 | 3. |
| 6 | GBM | **290,677** | **5.3818** | **2.** |

The RMSE of the last stage is now 5.3818 that would achieve 2nd place on the public leaderboard. The cost of this improvement is that now the total CPU time needed is ∼81 hours, 3.5 times more than previously. My best submission in the competition for the public leaderboard followed more

or less the same approach as this experiment, and it had an RMSE score of 5.3926.

### A. Simplifying the Solution

It is not necessary to apply 10-fold cross validation and a 6-stage model, if we can tolerate a slight loss in the accuracy. Fairy good RMSE can be achieved with using only 3 folds and 3 stages. Gradient boosting can be further speeded up by decreasing the value of the maximal tree depth (max_depth) parameter.

Another obvious simplification possibility is to leave out the 2 features extracted from the training period (gb/gd3_ap, tt3_ga). They were only kept in the final solution because I did not have enough time during the competition to precisely check the effect of omitting them.

In the next experiment, all of the above simplifications are applied, and the elements of the solution are added incrementally. The results are shown in Table III. The stages are separated by dashed lines. The regularization coefficient in the RR stage was alpha=100. The parameters of the GBM stage were max_depth=8, n_estimators=50, learn_rate=0.16 and random_state=42. The symbol + in front of a feature group name means that the features of the previous experiments (up to a dashed line) were also included among the features.

Table III
THE UTILITY OF VARIOUS FEATURE GROUPS AND REGRESSORS.

| Features | Alg. | Time | RMSE | LB Pos. |
|---|---|---|---|---|
| fh/cutoff | | 0 | 93.8241 | 165. |
| Main | RR | 1 | 6.8691 | 29. |
| + $\text{Sparse}_1$ | RR | 32 | 6.3515 | 19. |
| + $\text{Sparse}_2$ | RR | 55 | 6.2155 | 16. |
| + $\text{Sparse}_3$ | RR | 181 | 6.1051 | 16. |
| + $\text{Sparse}_4$ | RR | 254 | 6.0775 | 16. |
| + Metar | RR | 422 | 5.9307 | 12. |
| + ATSCC | RR | 453 | 5.9150 | 12. |
| + GroupBy | RR | 525 | 5.9172 | 12. |
| Main | GBM | 895 | 5.8804 | 12. |
| + Delta | GBM | 1,735 | 5.7748 | 10. |
| + Position | GBM | 2,399 | 5.6857 | 7. |
| + Other[1] | GBM | **3,641** | **5.6415** | **7.** |

It can be seen that GroupBy features should be indeed omitted, since the decrease the accuracy. The last line shows that if we can tolerate a 5 % increase in the RMSE compared to the 5.3818 result, then it is possible to reduce the time requirement by 98.7 %.

The Sparse feature group was divided into 4 sets in this experiment as $\text{Sparse}_1$={fh/aac, fh/alc}, $\text{Sparse}_2$={fh/aach, fh/aact, fh/aaca}, $\text{Sparse}_3$={fp/@lr, fh/sact, fh/aaic, fh/aacg}, $\text{Sparse}_4$={fh/edge, fh/flight, fp/acid, fh/aacb}. It is interesting to see that more than half of the improvement on the trivial solution with RMSE 6.8691 is due to the sparse features.

[1]The feature fh/cutoff was not included.

## VI. CONCLUSION

This paper presented my solution that won third prize in GE Flight Quest. The proposed model consists of 6 consecutive stages of ridge regressions and gradient boosting machines, trained on 56 features. I used a relatively low number of features and a relatively complex model compared to other contestants. To cope with the massive amount of available data, separate models were built on different partitions of the data and their answers were averaged. The time requirement of my solution can be reduced by 98.7 % at the cost of a 5 % loss in the accuracy.


## ACKNOWLEDGMENT

This paper was subsidized by TÁMOP-4.2.2.C-11/1/KONV-2012-0012/: "Smarter Transport" – IT for co-operative transport system – The Project is supported by the Hungarian Government and co-financed by the European Social Fund.


## REFERENCES


[1] J. Leber, "A data-crunching prize to cut flight delays," 2013, mIT Technology Review, http://www.technologyreview.com/news/513141/a-data-crunching-prize-to-cut-flight-delays.

[2] A. E. Hoerl and R. W. Kennard, "Ridge regression; biased estimation for non orthogonal problems," *Techonometrics*, vol. 12, pp. 55–67, 1970.

[3] J. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[4] G. F. Newell, "Airport capacity and delays," *Transportation Science*, vol. 13, no. 3, pp. 201–241, 1979.

[5] M. D. Peterson, D. J. Bertsimas, and A. R. Odoni, "Models and algorithms for queueing congestion at airports," *Management Science*, vol. 41, pp. 1279–1295, 1995.

[6] R. Beatty, R. Hsu, L. Berry, and J. Rome, "Preliminary evaluation of flight delay propagation through an airline schedule," *Air Traffic Control Quarterly*, vol. 7, pp. 259–270, 1999.

[7] K. F. Abdelghany, S. S. Shah, S. Raina, and A. F. Abdelghany, "A model for projecting flight delays during irregular operation conditions," *Journal of Air Transport Management*, vol. 10, no. 6, pp. 385–394, 2004.

[8] Y. Tu, M. O. Ball, and W. S. Jank, "Estimating flight departure delay distributions – A statistical approach with long-term trend and short-term pattern," *Journal of the American Statistical Association*, vol. 103, no. 481, pp. 112–125, 2008.

[9] N. Pyrgiotis, K. M. Malone, and A. Odoni, "Modelling delay propagation within an airport network," *Transportation Research Part C: Emerging Technologies*, vol. 27, pp. 60–75, 2013.

[10] K. Malone, "Dynamic queuing systems: Behavior and approximations for individual queues and networks," Ph.D. dissertation, Massachusetts Institute of Technology, 1995.