Syracuse University

Pushback to the Future: Predict Pushback Time at US Airports (Model documentation)

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

We are a group of two phD students at Syracuse University and two machine learning engineers working in Ho Chi Minh, Vietnam.

2. What motivated you to compete in this challenge?

We are interested in the privacy aspect of the challenge and would like to implement a federated learning framework on real world data. The federated learning framework allows different organizations to train a shared global model without the need to share their own local data.

3. High level summary of your approach: what did you do and why?

In a nutshell, we perform different feature engineering to extract time-series signals from different tables. For example, we extract the day in the week (e.g., Sunday) , block time of the day (e.g., morning), number of departure/arrival flights just before the time of making predictions. After that we train an XGBoost regressor model to predict the minutes until pushback time.

4. How did knowing that you might need to federate this model change your approach to this challenge problem?

Federated learning is suitable for gradient-based learning methods such as neural network classifiers or logistic regression. Extending our solution (boosted trees) to distributed learning is not trivial. However, there was some prior work which attempted to do that. For example, see the paper “Federated Boosted Decision Trees with

Differential Privacy". We think we can extend our solution in Phase 2 based on those prior work.

5. Do you have any useful charts, graphs, or visualizations from the process?

We do not have.

6. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

- The following line of codes helps to join two tables, one is the training data (or submission data), the other is the etd table **quickly**. We believe that the processing time for feature engineering is crucial due to the time limit in code execution. The output is one table which shows the departure_runway_estimated_time for a gufi, given right before the timestamp (the time we are making the prediction)

```
final_pd = pd.merge_asof(test_pd.sort_values(by=['timestamp']),
etd.sort_values(by=['timestamp']), on='timestamp', by='gufi',
tolerance=pd.Timedelta("30h"), direction='backward'
```

- The next following line of codes decompose the timestamp into 24 segments of hours. Based on date_hour (instead of full time stamp), we can join two tables (e.g., runways table and submission table) based on gufi and date_hour **quickly.**

```
rw['prev_timestamp_1hr'] = rw['timestamp'].apply(lambda x: x -
timedelta(hours=1))
rw['prev_date_hour'] = rw['prev_timestamp_1hr'].apply(lambda x:
str(x).split(":")[0])
```

- The following line of codes extract the name of the airline. Based on our data exploration, we found that some airline can take longer time for the pushback than the others.

```
test_pd['airline'] = test_pd['gufi'].apply(get_airline).astype('category')
```

7. Please provide the machine specs and time you used to run your model.
● *CPU (model):* Apple M1 Pro
● *GPU (model or N/A*): N/A

● *Memory (GB):* 32GB
● *OS:* MacOS 13.3.1 (22E261)
● *Train duration:* 1 to 2 hours per airport, 15 hours in total for 10 airports.
● *Inference duration:* ~3 minutes per airport, 30 minutes in total for 10 airports

8. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

No, but we encourage to use high memory machine (at least 16 GB) since some of the tables is pretty big and can occupy a lot of memory

9. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No, we did not use any unpopular tools. We only use newest version of Pandas, XGBoost and Sklearn.

10. How did you evaluate performance of the model other than the provided metric, if at all?
We just only used MAE (the provided metric) to train and evaluate the XGBoost model. The key thing here is that XGBoost version >=1.7.x allows to minimize the MAE objective function directly. The previous version might only allow us to minimize the proxy objective function such as RMSE (root mean squared error).

11. What are some other things you tried that didn't necessarily make it into the final workflow
(quick overview)?

We tried other features but it turned out they can not improve the performance (based on the prescreened area ranking). For example, based on TFM table, we computed some statistics on the arrival_runway_estimated_time column, like the difference between the latest subtracted the earliest arrival_runway_estimated_time. These features help to reduce the MAE on Open Area but not on Prescreened Area.