CS523 Project Report for NASA Airport Throughput Prediction Challenge

Krishna Patel[1] and Samuel Wu[1]

[1]Boston University

December 11, 2024

Abstract

Purpose - This study aims to build a prediction model that can, given an airport, date, and time, accurately predict the number of arrivals at that given runway. Knowing this will enable aviation operators to perform better flight planning and improve the flight efficiency.

Design/Methodology/Approach - The study employs both a multi-step multi-modal time series RNN and an XGBoost Regressor. Data were collected from the NASA Airport Throughput Prediction Challenge.

Findings - Our models were able to predict airport arrivals up to a root mean squared error (RMSE) of 5.299 and 3.6 for the LSTM and XGBRegressor respectively. In total, we managed a final rank of 20 in the NASA Competition Leader board.

Keywords: RNN; LSTM; NASA; XGBoost; Time Series Forecast; Airport Throughput

1 Introduction

The goal of this project was to build a prediction model that can, given an airport, date, and time, accurately predict the number of arrivals at that given runway. Knowing this will enable aviation operators to perform better flight planning and improve the flight efficiency. To achieve this, our project had two main goals. First, we sought to process the raw data into usable and correctly formatted inputs through extensive preprocessing and quantization of human-readable data. Second, we aimed to build a multimodal, multi-step LSTM for time series forecasting. We built this model, along with an XGBoost Regressor, to compare the benefits of both models in time series forecasting.

2 Method

Data Preprocess

We worked backwards to plan the preprocessing of the raw data. The competition submission required predictions to be sorted by ID that contained an airport, date, and time, organized 15 minute intervals. Because of this, we also split the input data into these buckets. For aggregate features, such as the number of arrivals, we added the total number of flights landing within a certain interval. For other features, such as temperature, cloud cover, and wind speed, we took the average over the interval. We also included one-hot encodings of certain features such as season, lightning probability, and cloud type.

To best capture the patterns of flight, we also included a myriad of time related features. This included day of year, month, season, and time of day. For cyclical features such as day of year or time of day, we fed the data through a sine function, in order to capture the repeating nature of the data. For example, 11:59 pm and 00:01 am are numerically distant, but fall near each other in time, while numbers like 00:00 am and 12:00 pm are very distant in time. Running the data through a sine function, the former values would fall closer together, while the latter become most distant.

There are 4 types of data provided by the competition: FUSER, METAR, CWAM, and TAF. The FUSAR data contained runway arrivals, along with low level weather predictions. METAR contained high level, precise weather predictions. TAF contained weather forecasts specifically for aviation. Finally, CWAM contained weather information in the form of a list of polygons at a

specific altitude within which there is a percentage of chance greater than 60, 70, 80 percent to be impacted by convective weather. For our purposes, we focused on FUSAR, METAR, and TAF data.

METAR

For the METAR dataset, we utilized the metar python package to decode each line of the METAR dataset. We found that many of the lines were either duplicates or very similar to one another, and so we filtered out duplicate data and utilized the ones that were either most recent or captured a unique occurrence within a recent time frame. In this case, a unique occurrence refers to any event that could cause an airport to deliver a METAR shortly after an autogenerated one; examples include rainfall, fog or snow.

METAR is inherently meant to be a human-readable source of information, despite certain METARs being generated automatically, and so an obstacle we encountered often was that certain METARs would often be left unparsed by the package due to national or personal difference in how they were written. Due to duplicate data and a propensity for certain features to be missing or remain unparsed, we additionally attached a secondary feature for each primary feature that indicated whether the primary was missing, using a numerical value of -1 within the primary itself so that the model could learn the difference between present and missing features.

METAR additionally contains several sources of qualitative data such as cloud cover and weather observations. Cloud cover data is typically expressed in words as FEW (few), SCT (scattered), BKN (broken) or OVC (overcast). Weather data is expressed in two letter representations of common whether phenomena with an added adjustment for heavy or light. Our model cannot meaningfully parse natural language in the form that the metar parser returns, and as a result we quantize these qualitative metrics.

Cloud cover can be quantized into oktas, how many eighths of the sky are covered by clouds, with FEW, SCT, BKN and OVC mapping to 1-2, 3-4, 5-6 and 7-8 oktas respectively. For our purposes we simply take the averages of 1.5, 3.5, 5.5 and 7.5. Cloud cover is described fractionally and so is a prime target for normalization but the LSTM will require normalized data regardless, so we do not normalize at this stage.

Weather poses a greater difficulty due to the large number of sparse weather phenomena which have a small possibility of occurring. In preprocessing our data, we pick five of the most common and impactful weather conditions in rain, thunderstorms, fog, snow, hail and mist to include in the final dataset. Weather conditions can also be either light or heavy depending on severity, and so we use 0 to indicate a lack of weather, and 1,2 and 3 for light, moderate and heavy weather phenomenon, respectively.

icao_code	date	time	wind_speed	wind_speed_mi	wind_direction	wind_direction_mi	visibility	visibility_mi	temperature	temperature_mi	dewpoint	dewpoint_mi	pressure	pressure_mi	peak_wind	peak_wind_mi	precipitation	precipitation_mi	rain	thunderst	hail	snow	mist	fog
AGGH	9/1/2022	0:00:00	17	0	90	0	10000	0	31	0	25	0	1011	0	-1	1	-1	1	0	0	0	0	0	0
AYMH	9/1/2022	0:00:00	4	0	-1	1	10000	0	20	0	17	0	1022	0	-1	1	-1	1	0	0	0	0	0	0
AYPY	9/1/2022	0:00:00	6	0	-1	1	10000	0	30	0	25	0	1010	0	-1	1	-1	1	0	0	0	0	0	0
BGBW	8/31/2022	23:45:00	2	0	70	0	10000	0	4	0	0	0	1021	0	-1	1	-1	1	0	0	0	0	0	0
BGGH	8/31/2022	23:45:00	10	0	70	0	10000	0	5	0	0	0	1022	0	-1	1	-1	1	0	0	0	0	0	0
BGJN	8/31/2022	23:45:00	6	0	190	0	10000	0	3	0	2	0	1025	0	-1	1	-1	1	1	0	0	0	0	0
BGKK	8/31/2022	23:45:00	7	0	230	0	900	0	3	0	2	0	1011	0	-1	1	-1	1	0	0	0	0	0	2
BGSF	8/31/2022	23:45:00	6	0	60	0	10000	0	4	0	1	0	1026	0	-1	1	-1	1	0	0	0	0	0	0
BGTL	8/31/2022	23:45:00	4	0	310	0	10000	0	2	0	2	0	1016.5966	0	-1	1	-1	1	0	0	0	0	0	0
BGTL	8/31/2022	23:45:00	5	0	300	0	10000	0	2	0	2	0	1016.5966	0	-1	1	0.01	0	0	0	0	0	0	0
BGTL	9/1/2022	0:15:00	5	0	270	0	10000	0	2	0	-1	1	1016.935	0	-1	1	-1	1	0	0	0	0	0	0
BIAR	9/1/2022	0:00:00	14	0	190	0	10000	0	11	0	4	0	1009	0	-1	1	-1	1	0	0	0	0	0	0
BIEG	9/1/2022	0:00:00	19	0	170	0	10000	0	16	0	9	0	1009	0	-1	1	-1	1	0	0	0	0	0	0
BIKF	9/1/2022	0:00:00	11	0	240	0	10000	0	8	0	6	0	1013	0	-1	1	-1	1	0	0	0	0	0	0
BIKF	9/1/2022	0:30:00	8	0	220	0	10000	0	9	0	7	0	1013	0	-1	1	-1	1	0	0	0	0	0	0
BIRK	9/1/2022	0:00:00	10	0	210	0	10000	0	9	0	6	0	1013	0	-1	1	-1	1	0	0	0	0	0	0
BKPR	9/1/2022	0:00:00	3	0	350	0	9000	0	17	0	16	0	1014	0	-1	1	-1	1	0	0	0	0	0	0
BKPR	9/1/2022	0:30:00	2	0	-1	1	10000	0	17	0	16	0	1014	0	-1	1	-1	1	0	0	0	0	0	0
CAHR	9/1/2022	0:00:00	13	0	130	0	-1	1	21.4	0	20	0	-1	1	21	0	0.01	0	0	0	0	0	0	0
CBAR	9/1/2022	0:00:00	7	0	80	0	-1	1	13	0	11	0	1004.0670	0	-1	1	-1	1	0	0	0	0	0	0
CBBC	9/1/2022	0:00:00	3	0	330	0	14484	0	23	0	14	0	1014.9034	0	-1	1	-1	1	0	0	0	0	0	0
CERM	9/1/2022	0:00:00	0	0	0	0	-1	1	17.2	0	15.7	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CMFM	9/1/2022	0:00:00	4	0	270	0	-1	1	12.1	0	9.9	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CMGB	9/1/2022	0:00:00	3	0	210	0	-1	1	17.6	0	10.9	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CPBT	9/1/2022	0:00:00	6	0	290	0	-1	1	31.6	0	6.9	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CPEH	9/1/2022	0:00:00	0	0	0	0	-1	1	31.5	0	9.5	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CPFI	9/1/2022	0:00:00	7	0	280	0	-1	1	31.6	0	11.2	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CPIN	9/1/2022	0:00:00	8	0	120	0	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	0	0	0	0	0	0
CPIR	9/1/2022	0:00:00	9	0	240	0	-1	1	32.7	0	8.2	0	-1	1	-1	1	-1	1	0	0	0	0	0	0
CPRO	9/1/2022	0:00:00	8	0	250	0	-1	1	32.8	0	7.2	0	-1	1	-1	1	-1	1	0	0	0	0	0	0

Figure 1: Our features contain both quantitative data and qualitative data, with qualitative measures such as heaviness of weather quantized.

TAF

We initially created a similar TAF parser with quantization and preprocessing to the previously shown METAR parser using the pytaf library but felt that it would not have benefited our model without a significant change to our data processing. Much of the TAF data overlapped with the METAR and the unique features which did not were difficult to integrate due to the TAF forecasting future weather rather than providing real-time data to pilots as the METAR does. This mismatch means that it would be imprudent to simply add the TAF data directly into our

dataset at the current times. We felt that integrating the TAF by 'pulling' the data towards the predictions would be detrimental without a more sophisticated method that we did not have time to implement.

PCA

Prior to training, we performed some PCA in order to determine which features were most useful. One useful feature was the estimated number of arrivals. This feature was calculated using the estimated arrival times of flights, where the arrival time was counted towards its respective interval. This feature, along with others such as temperature and time of day, showed strong correlations to the number of arrivals.

A sample of the processed data is shown below.

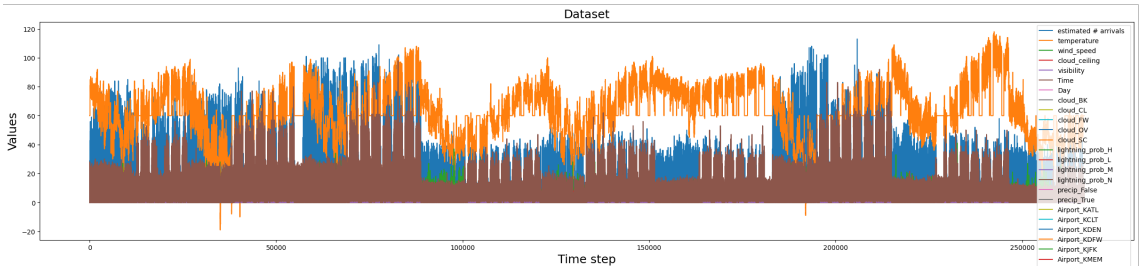

Figure 2: X corresponds to time, with y corresponding to the value of each feature shown in the legend.

Multistep/multimodel LSTM

In order to input data into the LSTM, we must first normalize the data. To do this, we use a MinMaxScaler to fit and transform the features. We then save this scaler for the inverse transformation of the predictions. The model takes the features as rows according to the time step index and trains an RNN on the features for each timestep. This allows the RNN to make predictions multiple steps into the future, though for our purposes we only predict one time step into the future. Thus, to make predictions, the model is given n steps in and predicts 1 step out. We found that the model's predictions became more accurate the larger n was, thus we increased the n to around 100, or 1 day worth of inputs.

We also introduced a custom layer that artificially increased the initial weight for specific features. This was done to encourage the RNN to 'pay attention' to certain features. In our case, we increased the initial weight of the 'estimated number of arrivals' feature as it showed the strongest correlation to the number of arrivals.

We experimented with different components of the RNN to see which hyperparameters produced the best predictions during testing. We experimented with LSTM layers from 2-4, timesteps in from 100-720, LSTM layer size from 64 to 128, initial weight size from 1 to 64, and batch size from 256 to 1024. Our best performing model had an initial weight of 64, 3 LSTM layers of size 64, in steps of 420, and batch size of 1024, with a RMSE of 5.299 and a competition score (exp(-RMSE/10)) of 0.589.

XGBoost Regressor

We also experimented with an XGBoost Regressor. We chose this model because of its quick training times and capacity to handle large amounts of data. The input features remained the same as the LSTM.

We experimented with differing hyperparameters in order to maximize the RMSE. We tested n estimators from 200 to 20000 and max depth from 8 to 16, with the highest scoring model receiving a RMSE of 3.685 and a competition score (exp(-RMSE/10)) of .69.

We also evaluated the importance of each feature used in the XGBoost model by examining its feature importance scores. This analysis revealed that certain features, such as temperature and wind speed, had a disproportionately higher influence on the model's predictions compared to others. Leveraging this insight, we considered simplifying the model by removing less impactful features to reduce overfitting and enhance interpretability, but this adjustment did not yield significant improvements in performance.

Data Postprocess

For both processes, data post-processing was required. For the LSTM in particular, predictions were run through a multi-threaded program to improve prediction speeds. These predictions were then re-indexed according to the submission format buckets and saved as a .csv. We experimented with rounded and unrounded outputs in our submissions and found no significant difference in score. The best submissions can be found on the GitHub.

3 Results

The performance of the proposed models was evaluated using RMSE and the competition exclusive metric of (exp(-RMSE/10)). Both models demonstrated high precision and robustness during testing, effectively distinguishing important features and arrival time patterns. Below are the key results:

XGBoost Regressor

- **RMSE:** 3.685 (# of arrivals)
- **exp(-RMSE/10):** .69

Mutlistep Multimodal LSTM

- **RMSE:** 5.299 (# of arrivals)
- **exp(-RMSE/10):** 0.589

These metrics indicate that the models were able to capture large patterns in the dataset in order to predict the number of arrivals. However, it failed to capture smaller details that may have decreased the RMSE, thus increasing the final score. Further approaches may include removing features that introduce too much noise and possibly introducing more features that may help capture small patterns still unnoticed by the model. This feature extraction may be limited due to the sheer volume of the competition dataset, which was one of the largest obstacles in working on this competition.

3.1 Performance Visualization

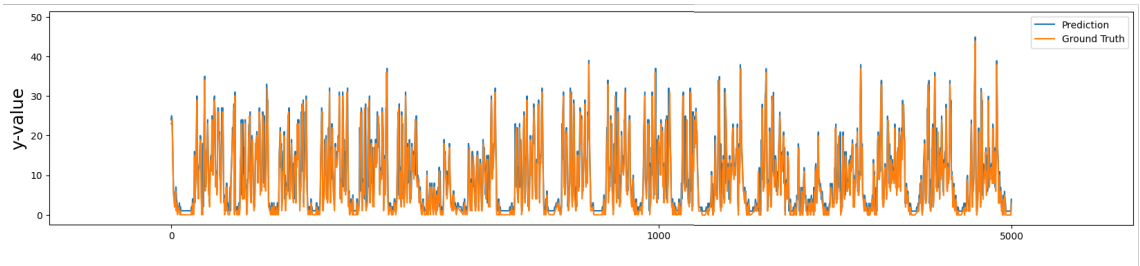

Figure 3: Model performance on a sample dataset. The closeness of the blue (ground truth) and orange (predicted) lines shows the accuracy of the model.

The figure above shows the trends of key performance metrics over the training epochs. The stability of these metrics while managing to capture the pattern of the data indicates that the model generalizes well without overfitting.

4 Conclusion

This paper details our methodology and results to predict airport arrival for the NASA challenge. Overall, we observe that our feature engineering and preprocessing especially with regard to the time intervals and PCA, respectively, aided our model significantly. We did not notice a substantial improvement from using all the data and noticed that a substantial amount of it contained overlapping information, especially the METAR and the TAF datasets. Our RMSE of 3.685 and

ranking 20th on the leaderboard show that our model achieved significant precision in predicting arrivals. Future work could focus on ensembling methods and attention based mechanisms to give greater weight to features with a larger impact on the arrivals.