CSCI 5525 - Project Final Report

Adam Bilski, Sam Penders, Hung-Da Wen
Group 8
University of Minnesota
Minneapolis, MN 55414, USA

1 Introduction and Problem Description

Air travel is a massive industry in the United States. Each year, there are over 16 million flights handled by the Federal Aviation Administration. At peak operational hours, there are 5,400 airplanes in the sky, and nearly 20,000 airports in the US alone [1]. Millions of travelers every day rely on flights to take off and arrive on time, but everyone who has flown before knows the frustration of an unexpected flight delay. In fact, over 20% of flights arrive to their destination late [2]. Thus, travelers would benefit from an advanced warning that their flight will arrive late to allow them to alter their plans. By predicting delays accurately, airline companies could also optimize their aircraft route assignments. According to the FAA, a flight is delayed if it arrives 15 minutes or more later than its scheduled time.

Our work focused on two goals. One was to produce a model that accurately predicts individual flight arrival delays six hours ahead of departure time. This information is beneficial to passengers, as well as aircraft operators who optimize routes across the whole system. For example, travelers can adjust their plans based on the model's warnings of flight delays. We focused this problem on flights from San Fancisco (SFO) to Los Angeles (LAX) due to a high availability of data, and a relatively high delay rate. Our other goal was to predict the delays in a sequence of flights for a single aircraft. Airlines can use this model to identify the overall rate of delay across an entire flight sequence to optimally manage aircraft utilization.

2 Technical ML Problem and Algorithms Used

Numerous factors impact the delay of a flight. These may include weather, prior flight delays, time of day, week, month, year, flight route, airline, and the specific aircraft. Simple models like logistic regression, which classifies observations based on a linear boundary, are unlikely to capture the complexities of the interrelationships between predictors and the response. The business use cases we present focus on predictive power over interpretability. As a result, we place emphasis on deep learning architectures, which can learn these non-linear decision boundaries. These models were implemented using PyTorch.[1]

2.1 Multilayer Perceptron

Specifically, for predicting whether a flight would be delayed 6 hours before its scheduled departure, we have the dataset in tabular form, where each row is a flight, with a binary delayed or not response as well as a set of features, extracted hours before the actual flight occurs. A multi-layer perceptron (MLP) architecture was chosen to predict the late arrival of flights. This architecture was chosen due to its combination of simplicity, ability to learn non-linear decision boundaries, tunability through choices of depth and layer width, and suitability for tabular data. We focused on MLP models with 2-4 hidden layers and 20-300 hidden nodes per layer. This architecture is shown in Figure 1.

2.2 Long-Short Term Memory

On the other hand, for the airline route optimization scheme, we explicitly take into account the sequential nature of flights. Through `Tail_Number`, the unique ID associated with each aircraft, we can identify every route an aircraft flies over on a given day. We can then treat each aircraft-date combination as a time sequence and predict delays for every flight in that sequence. This is an example of many-to-many time series prediction. The sequential nature of our problem naturally leads us to consider RNN or LSTM. Since vanilla RNN tends to encounter the vanishing gradient problem, we consider LSTM exclusively, which is generally considered to model long-term dependencies better.

3 Data Description and Preprocessing

Our data comes from the Airport Reporting Carrier On-Time Performance Dataset, which was released by the US Bureau of Transportation Statistics and compiled by IBM [3]. It consists of 200 million US domestic flights between 1987 and 2020. Relevant features include basic flight information, such as airline, scheduled/actual departure/arrival date/time, airport, flight distance, and scheduled duration. We also have access to actual departure/arrival delay and the assignment of the number of minutes of delay to five different reasons for delay, although these are not utilized. We discuss the data preprocessing for MLP and LSTM separately below.

[1] Source code for our project can be found on Github:
`https://github.com/sampenders/Flight-Delay-Prediction`

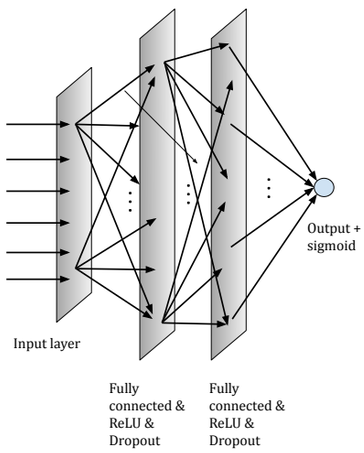

Figure 1: Architecture of MLP used. All of the layers are fully connected, and a dropout layer with a ReLU activation function follows each hidden layer. Note that the number of hidden layers and nodes was varied. This graphic was created by the authors.

3.1 MLP: Data Preprocessing

The data for MLP consists of two parts: the aforementioned flight dataset and the externally obtained daily weather data.

- Flight data: the dataset was reduced to include only flights between 2010 and 2020 and those that either departed or landed at SFO or LAX. We cleaned the data by removing rows missing the flight delay status, and interpolated missing weather data from the neighboring days. There were very few missing values to handle. The eight airline values were one-hot encoding labels for the airlines. We also derived the following features:
  - `prev_tail_delay_percentage`: with the fundamental intuition that specific aircrafts might be more prone to delays than others, we created a feature to capture this. Using the tail number, every flight is assessed by examining all of its previous flight history, and then a percentage of history for which flights are delayed is calculated. Therefore, one tail number can have many different values over time, depending on how well the plane has been performing in the past.
  - Airport Arrival/Departure `DelRate`: Because current flight delays influence future flight delays, additional features were needed to encode information on previous delays. For both SFO and LAX and both departures and arrivals, a prior delay rate feature was created. These features are an exponential weighted mean of the delay status for all previous flights departing or landing up to six hours prior to prediction time. The half-life for this exponential weighted mean was chosen to be one hour so that flights in the hours preceding prediction time had the greatest influence. As an example, the `SFODepDelRate` feature for a flight departing SFO at 8pm is the exponential weighted mean of the departure delay status of all flights departing SFO, to any destination, before 2 pm.
- Weather data: Daily weather data was obtained for SFO and LAX through the `meteostat` Python library. We utilized the average, minimum, and maximum temperatures, precipitation amount, and wind speed. All of the weather data is processed and merged in with the corresponding day of each flight.

Besides eight one-hot encoded airline names, we have 23 features from the flight dataset in total, as shown in Table 1. The features were standardized to have zero mean and unit standard deviation, shuffled, and split into 68%/12%/20% training/validation/testing sets.

3.2 LSTM: Data Preprocessing

The dataset used for LSTM is similar to what was used for MLP, except for certain areas that are unique to the sequential nature of our problem.[2] We did not incorporate the weather data as it requires obtaining the merging key for every US airport.

- Dataset format: as discussed earlier, each observation is a `Tail_Number–FlightDate` pair. An aircraft does not fly back-to-back nonstop; it typically flies for at most 18 hours and is on stand-by between midnight and early morning. Therefore, it is reasonable to assume the delay pattern does not propagate across days. To narrow the

[2]Data file: `https://drive.google.com/file/d/1gm5NuXvK8YTDuJRBm9FJOSKZhtRldLp7/view`

scope, we restrict to aircrafts that did exactly 6 flights on a day (which has the most `Tail_Number-FlightDate` pairs) and use only data from 2017 to 2019. Each observation has a feature matrix of dimension `6 x number of features` and a response vector of length 6.

- Target encoding and category grouping: given the number of unique values in airport codes, we decided to keep only the names of top 20 airports and change all other airports to "AllOthers". This restricts the number of airport-related features from more than 700 to just 42. To further reduce sparsity, we also employ target encoding for `DepTimeBlk` and `ArrTimeBlk`. This is done by creating a numeric column whose values are the delay rates for each level of `DepTimeBlk` or `ArrTimeBlk`. To prevent data spillover, we only use the target encoded column derived from training to be merged into validation and testing sets. The dataset is shuffled and split into train/validation/test sets of size 80%/10%/10%. All features were standardized to have zero mean and unit standard deviation.

MLP		LSTM	
Features	**Description**	**Features**	**Description**
Year	Year of flight	Month	Month of Flight
Month	Month of flight	DayOfWeek	Day of Week of flight
DayofMonth	Day of Month of flight	Reporting_Airline	Reporting airline
DayOfWeek	Day of Week of flight	Origin	Origin airport code
CSRDepTime	Scheduled Departure Time	Dest	Destination airport code
CSRArrTime	Scheduled Arrival Time	DepTimeBlk_target_encoding	Target encoded column for departure hour time block
SFODepDelRate	**SFO Departure Delay Rate**	ArrTimeBlk_target_encoding	Target encoded column for arrival hour time block
LAXDepDelRate	**LAX Departure Delay Rate**	CRSElapsedTime	Scheduled flight duration in minutes
SFOArrDelRate	SFO Arrival Delay Rate	Distance	Distance between origin and destination
LAXArrDelRate	LAX Arrival Delay Rate		
sfo tavg SFO	current flight day temperature average		
sfo tmin SFO	current flight day temperature minimum		
sfo tmax SFO	current flight day temperature maximum		
sfo prcp SFO	current flight day precipitation total		
sfo wspd SFO	current flight day wind speed		
sfo pres SFO	current flight day pressure		
lax tavg LAX	current flight day temperature average		
lax tmin LAX	current flight day temperature minimum		
lax tmax LAX	current flight day temperature maximum		
lax prcp LAX	current flight day precipitation total		
lax wspd LAX	current flight day wind speed		
lax pres LAX	current flight day pressure		
prev tail delay percentage	Tail number's previous delay percentage		

Table 1: List of features for MLP and LSTM. For MLP: specific airlines were one-hot encoded and added to the feature list above. The data included 8 unique major commercial airlines flying to and from SFO and LAX.

4 Model Training

4.1 MLP

To benchmark our MLP results, we trained several baseline models for comparison, including Logistic Regression, Random Forest, and XGBoost on the same train/test sets as the other models. These models were chosen because of their widespread use.

The MLP models were trained on the training and validation sets, using a weighted binary cross-entropy loss function and the Adam optimizer. Delayed flights were given a weight of three because there are approximately three times as many not-delayed flights in the data. The models were trained using early stopping, such that the training ended when the validation loss became worse than the loss ten epochs prior.

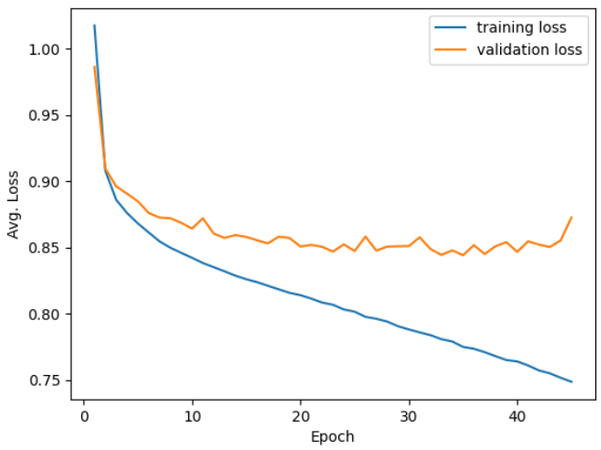

Figure 2: Training and validation loss for 4 hidden layer, 300 hidden nodes/layer MLP.

4.2 LSTM

Similar to MLP, our LSTM models are trained with a weighted binary cross-entropy loss function and the Adam optimizer. We fix the `pos_weight` parameter for `BCEWithLogitsLoss` to be 4.62, which is the ratio of non-delayed count/delayed count to address class imbalance. Learning rate is fixed to be 0.001. Early stopping is determined based on validation loss: after we trained our model for at least 5 epochs, if the validation loss from the most recent epoch is larger than that from two epochs back, then we stop training and calculate metrics on the test set. For hyperparameters, we explored 1 or 2 recurrent layers (corresponding to the `num_layers` parameter from `nn.LSTM`) and 30 or 50 hidden layer features (corresponding to the `hidden_size` parameter from `nn.LSTM`).

5 Results and Discussion

5.1 MLP

Table 2 shows the performance metrics for the MLP models and baseline models on the test dataset. XGBoost performed the best in terms of accuracy (0.76), precision (0.51), F1 score (0.58), and AUC (0.73), although its performance was a mere 1-2 percentage points better than the best MLP. The deepest, widest MLP (4 hidden layers, 300 nodes wide) performed the best among the MLPs in all metrics other than recall and AUC. This MLP had 0.74 accuracy, 0.49 precision, 0.58 F1, and 0.71 AUC. The AUC scores were esentially equal for all MLP models.

Notably, the simplest MLP, with 2 hidden layers and 20 hidden nodes/layer, had the highest recall out of all models, scoring 0.737. The recall tended to increase, and precision decreased, for MLP models as the model complexity decreased. We suspect that the simpler models could not learn as complex of decision boundaries, so the greater weight given to the delayed flight class led the models to classify more instances as delayed.

Model	Hidden Layers	Width	Learning Rate	Accuracy	Precision	Recall	F1	AUC
MLP	2	20	0.00001	0.700806	0.439378	0.706172	0.541707	0.702593
MLP	2	20	0.00010	0.682088	0.422687	**0.737031***	0.537257	0.700383
MLP	2	50	0.00001	0.692346	0.432147	0.728119	0.542383	0.704258
MLP	2	50	0.00010	0.691480	0.431391	0.729715	0.542229	0.704212
MLP	2	100	0.00001	0.700340	0.439271	0.711492	0.543184	0.704053
MLP	2	100	0.00010	0.700673	0.439751	0.713089	0.544015	0.704807
MLP	3	20	0.00001	0.691247	0.430233	0.718542	0.538209	0.700336
MLP	3	20	0.00010	0.707834	0.445846	0.686619	0.540637	0.700770
MLP	3	50	0.00001	0.707834	0.446693	0.698856	0.545021	0.704844
MLP	3	50	0.00010	0.708966	0.447250	0.687949	0.542082	0.701968
MLP	3	100	0.00001	0.725320	0.466495	0.675047	0.551720*	0.708580
MLP	3	100	0.00010	0.712563	0.452373	0.702447	0.550333	0.709195*
MLP	3	200	0.00001	0.726286	0.467430	0.668130	0.550044	0.706921
MLP	3	200	0.00010	0.725653	0.465496	0.645118	0.540782	0.698837
MLP	4	300	0.000001	0.715594	0.454137	0.672386	0.542120	0.701207
MLP	4	300	0.000010	0.743805*	0.490962*	0.628625	0.551330*	0.705452
XGBoost				**0.757327**	**0.511868**	0.665470	**0.578649**	**0.726741**
Rand. Forest				0.682288	0.415063	0.656824	0.508679	0.673809
Logistic Reg.				0.679723	0.419973	0.732243	0.533792	0.697211

Table 2: Performance metrics for MLP models. The bolded numbers indicate the best in that category. The numbers with the asterisk indicate the best among the MLP models.

5.2 LSTM

Table 3 shows the the performance metrics for the LSTM models on the test dataset. The models with 1 recurrent layer perform better in terms of accuracy and precision but perform worse in terms of Recall and F1 than those with 2 recurrent layers. In a route optimization scheme, we might prefer models offering higher recall, as it is more undesirable to incorrectly predicting a flight as not delayed.

Number of Recurrent Layers	Hidden Size	Accuracy	Precision	Recall	F1	AUC
1	30	0.78309021	0.361278	0.353516	0.357355	0.697106
1	50	**0.78939289**	**0.368789**	0.329601	0.348095	0.696244
2	30	0.77560897	0.354923	**0.385724**	**0.369683**	0.699464
2	50	0.78125637	0.362597	0.372396	0.367431	**0.700477**

Table 3: Performance metrics for LSTM models. The bolded numbers indicate the best in that category.

6 Conclusion

6.1 Incorporating Feedback

Initially we received feedback from colleagues to incorporate weather data into the model. This improved the results, but there are still other features we want to add to training. Originally, we proposed a number of different flight delay-related questions to work on, and we accepted the feedback to focus on just one or two. We incorporated the feedback of using different weights to address class imbalance by using the weighted binary cross entropy loss. Finally, we took the feedback to treat this as a time-series problem and proposed a LSTM framework to optimize aircraft route utilization in a given day. We also added features that represent previous delays in the MLP models.

6.2 Next Steps

For LSTM specifically, we could do more expansive hyperparameter tuning given enough computing power. Since the main goal is to optimize routing, identifying ways to represent each airport location, such as using techniques like word embedding in natural language processing, might prove to be helpful. The outputs of the LSTM could potentially be included as a constraint (such as having overall delay rates below a certain percentage) in an airline profit maximization model - we leave that for future researchers to explore. Lastly, we can expand model applicability by allowing for inputs to have number of flights other than 6.

In practice, one needs to apply weather forecast data in the model. We trained the MLP models on actual historical weather data, which might not be available if we deploy in real time. This is something we can improve on in future iterations of this work. In the future we would also like to include macro-economic indicator variables as these may correlate with airline disruptions. We also want to scale up the scope of prediction, and fit the MLP model to airports other than SFO and LAX.

References

[1] *Air traffic*. URL: https://www.faa.gov/air_traffic.

[2] *Airline On-Time Statistics and Delay Causes*. URL: https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp.

[3] *Airline Reporting Carrier On-Time Performance Dataset*. Nov. 2020. URL: https://developer.ibm.com/exchanges/data/all/airline/.