Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

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.

Normen Yu: He is an undergraduate student studying Computer Science at the Pennsylvania State University. He has a passion for working in the aerospace community. For example, he interned at Collins Aerospace working on UAVs, as well as a finalist on a team for the NASA BIG Idea Challenge.

Dr. Mehrdad Mahdavi: he is a professor at Pennsylvania State University. His research focuses on Machine Learning and he teaches a number of graduate and undergraduate machine learning classes.

2. What motivated you to compete in this challenge?

The project stemmed out of an honors project that Normen was taking in Dr. Mahdavi's class. In order to gain some practical experiences, this competition was chosen. Its dataset and problem offers an opportunity to explore many classical classification methods taught in class (SVM, Logistic Regression, Decision Tree, gradient boosting), with a twist for potential temporal decay concerns. Therefore, this competition was chosen because its scope matched the scope of the materials taught in class in a practical manner.

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

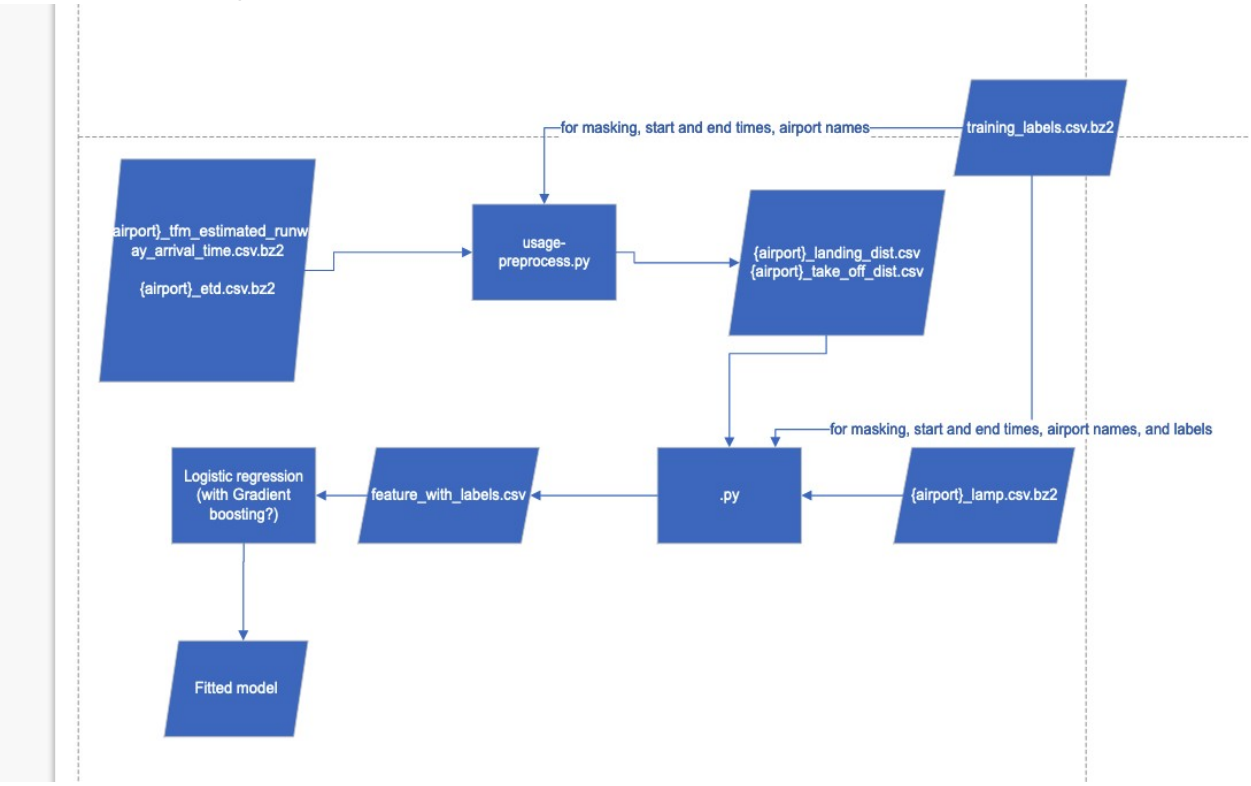
A number of algorithm was tested. However, Logistic Regression was ultimately used because its objective function matched closest to the objective function of the problem statement. It is also less prone to overfitting.
For each airport, the following was done independently of each other:
First, projected take-off and projected landing data for each airport were processed into an array of 4 columns: projected landing/takeoff 1 hour to 30 minutes ago, 30 minutes to 0 minutes ago, 0 minutes to 30 minutes into the future, and 30 to 1 hour into the future.

Then, this data was added into the mix of other data that required less processing: wind_speed, wind_gust, cloud_ceiling, visibility, cloud, lightning_probability, wind_direction (wind speed direction was splited/"kernelized" into a cosine and sine component), precipitation, and how many hours we are trying to look-ahead.

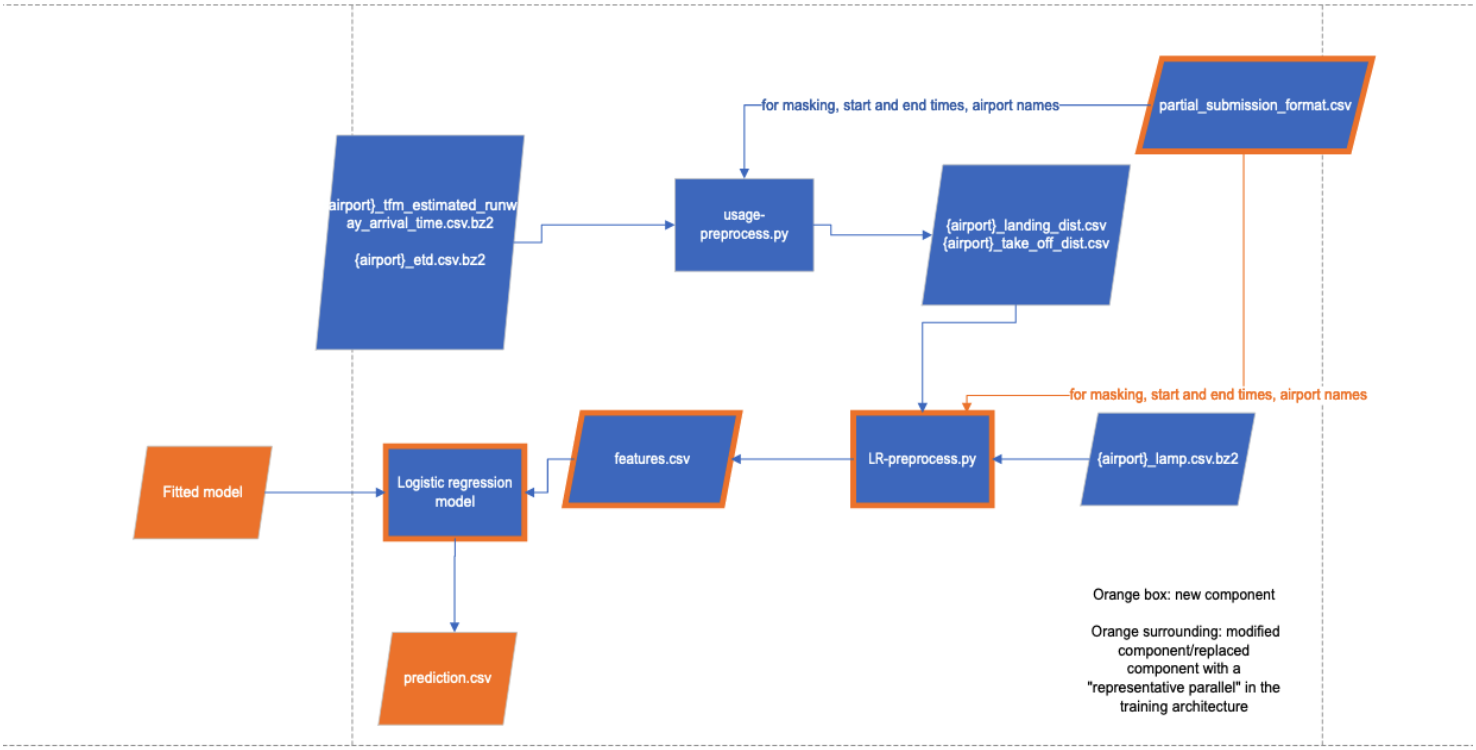
We finally standardized the data to have the same mean and variance and train the model. We utilize the training standardization to standardize the test data. These models are saved as a pickled file and used during testing.

Due to the temporal decay that we observed, we also upweighted the training data to place a higher weight on the data points in the May/June months.

4. Do you have any useful charts, graphs, or visualizations from the process?
   This training architecture explains the different pieces of the code:



This testing architecture explasin the code used to test the model, and how some of the code relates/are identicle to the training code:



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

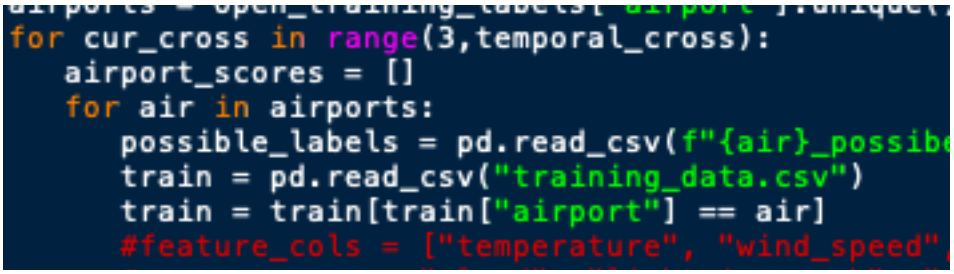
```
weather_data["precip"] = (weather_data["precip"])*1
weather_data["wind_direction_cos"] = np.cos(weather_data["wind_direction"]*10*np.pi/360)
weather_data["wind_direction_sin"] = np.sin(weather_data["wind_direction"]*10*np.pi/360)
weather_data = weather_data.drop('wind_direction', 1)
```

Because on the linear nature of logistic regression, this method of kernelization of the wind direction allowed for a stronger correlation between the angle of wind and the configuration that is used.

```
getChunk1 = getChunk[(chunk["estimated_runway_arrival_time"] > curTime+dt.timedelta(hours=(j-2)*.5))]\
                    [chunk["estimated_runway_arrival_time"]<=curTime+dt.timedelta(hours=(j-1)*.5)]

timeBinsLanding[i][j] = (timeBinsLanding[i][j] + getChunk1["gufi"].unique().shape[0])
```

This block of code allowed for the meaningful utilization of the take-off and landing data. The data that was generated reflects a histogram (except, each row represents a “sliding window” on the temporal axis, reflecting the aircrafts that are planning to take-off/land right before and right after the time specified). The processing of this data was extremely time consuming. Doing this in the preprocessing stage required retrieving the data chunk by chunk to avoid maxing out the memory.



While this was not directly used in the final code, splitting the data into temporally segregated chunk was essential in the hyperparameter tuning of our code (especially picking the optimal regularization values). We found out that using an i.i.d. distribution does not completely capture the nature of the problem due to temporal dependencies. Therefore, we split the data into chunks according to time to train on one chunk and test on the next.

6. Please provide the machine specs and time you used to run your model.
   Training computer: c2-standard-4 Google Cloud computing engine
   - CPU (model): Intel Cascade Lake
   - GPU (model or N/A): n/A
   - Memory (GB): 25 GB
   - OS: Linux
   - Train duration: 20 hours preprocess (12 hours to process landing/takeoff data, 8 hours to preprocess) + 2 hour model training = 22 hours
   - Inference duration: 20 hours preprocess (done during training) + 15 minutes to produce prediction.csv for open arena

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

N/A

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?
   No

9. How did you evaluate performance of the model other than the provided metric, if at all?
   None other was used.

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

Gradient boosting was tested but was too easy to overfit the data.
A polynomial kernel was attempted on a number of configuration as well, but those also shown to be too complex to train from the limited data provided.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

The first method would be to utilize more visualization techniques to see which features actually have a high correlation to the configuration used. After we isolate out those features, we should attempt to combine our model with the baseline regency code in a coherent way (such as a weighted sum of the two results). We believe that by combining our efforts to utilize other factors (such as weather and landing throughput needs) and a more hyperparameter-tuned baseline (which takes the temporal aspect into account from most recent configurations), a strong model can be created.