15 Minute Airline Delay Prediction Model

TECHNICAL REPORT

Kenyen Hast,

Ryan Blankenbeker, John Wallace

For:
Data Mining - CAP4770

December 2025

Problem Statement & Background Information:

For our project, we will use the Airline On-Time Performance Delay Cause dataset provided by the U.S. Bureau of Transportation Statistics. This dataset shows monthly aggregated delay metrics for each combination of airline–airport in the United States. Each record provides the number of flights, number of delays, and causes of delay such as:

- Late-Aircraft Delay,
- Weather Delay,
- NAS Delay, Carrier Delay,
- Security Delay,
- Arrival Delay (minutes),
- Percent of arrivals delayed ≥ 15 minutes

Our goal is: To develop a machine learning model which can predict whether a flight (or group of flights) will be delayed by 15 minutes or more, given pre-flight information such as month, carrier, airport, and historical performance. This problem has been well-studied in aviation research. Prior works demonstrate that it is possible to use supervised classification and anomaly detection to find operation inefficiencies. Our project extends this by applying Logistic Regression, Random Forest, K-Nearest Neighbors, DBSCAN anomaly detection, MySQL storage and Python analytics, and Comprehensive EDA visualizations

Flight Delay Data: https://www.bts.gov/

My SQL Overview:

To set up our project, I began by installing MySQL on my machine. After the installation finished, I created a new database named cap4775 to store our flight data. The next step was to get the raw dataset from the BTS website. Since the files were not in a format MySQL could read directly, I converted the download into a CSV file.

Once the data was in CSV form, I used the MySQL import wizard to load it into the database. The wizard automatically organized the data and generated the SQL needed to create the tables and insert the values.

To work with the database through Visual Studio Code, I installed two SQL extensions that let me connect to the MySQL server and run queries directly from the editor. I also imported libraries in my code to simplify formatting and make sure the program could read and send SQL data without issues.

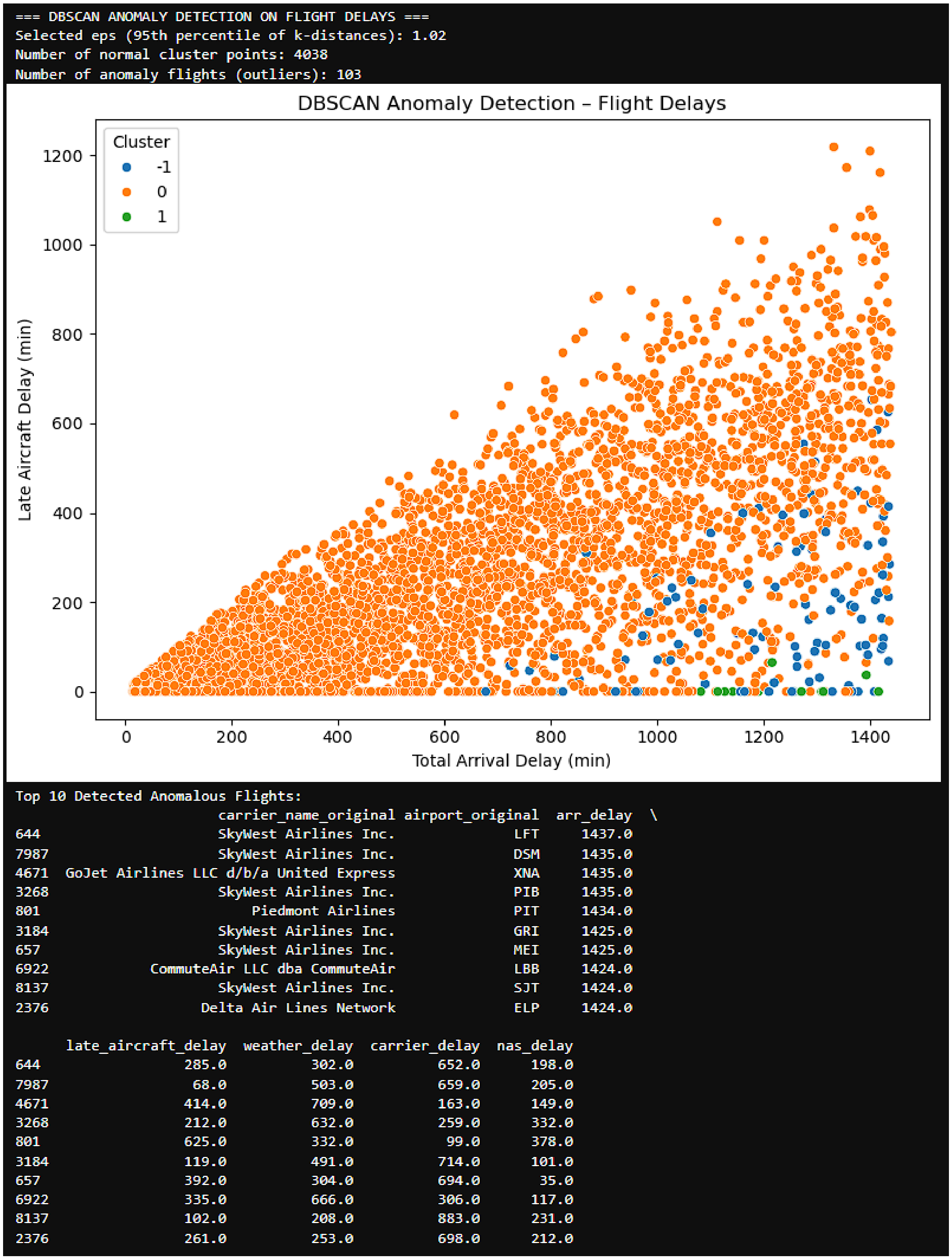
Data Collection & Preprocessing:

The first significant step in working on this project involved gathering a dataset that was large enough to meet the requirements of the project but yet relevant to the purpose of flight delay prediction. Our team chose the BTS for the data source because it offers validated government transportation data, which also comprises thorough delay statistics every month by airline and airport. The raw dataset initially had over 200,000 rows in two years' records and thus needed trimming down to a workable size. After trying various combinations, we filtered the data to a six-month window, which produced approximately 9,500 rows, within the required range of 7,000–10,000 datapoints.

In the preprocessing stage, our first priority was handling categorical variables such as airline carrier codes and airport identifiers. These features are vital for the understanding of delay patterns, but most machine-learning models cannot use them directly, so we applied Label Encoding to them. We also selected a set of relevant predictors: month, carrier, airport, and the number of arriving flights. We removed rows with missing or invalid entries to ensure the coherence of this data. We then created a binary target variable, "delayed", based on the percentage of flights delayed in each record, enabling us to treat the problem as a classification problem.

The final step was a train-test split to avoid leakage and allow for fair evaluation of our models. This preprocessing pipeline ensured that the dataset was clean, numerically compatible with machine-learning algorithms, and correctly structured for the models we built later in the project.

Exploring Data:

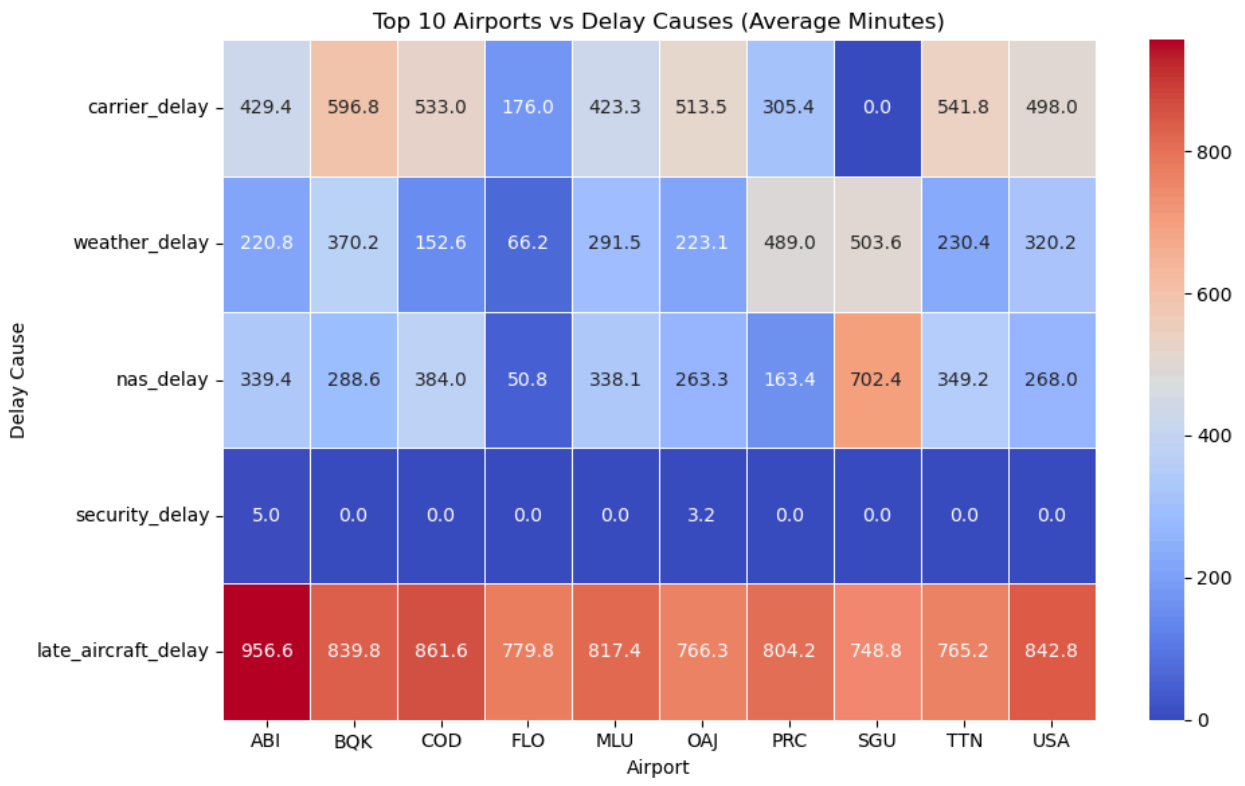


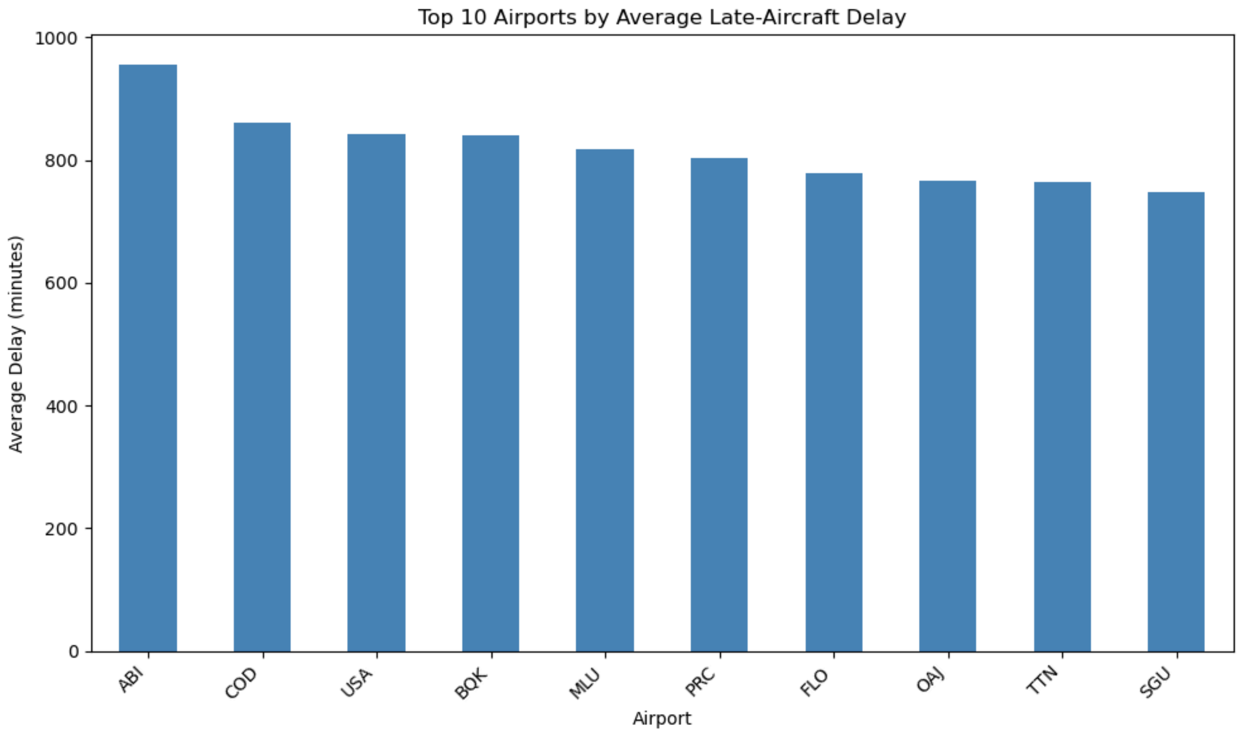
```
=== DBSCAN ANOMALY DETECTION ON FLIGHT DELAYS ===
Selected eps (95th percentile of k-distances): 1.02
Number of normal cluster points: 4038
Number of anomaly flights (outliers): 103
```

DBSCAN Anomaly Detection – Flight Delays
Cluster
-1
0
1
Late Aircraft Delay (min)
Total Arrival Delay (min)
0
200
400
600
800
1000
1200
1400

```
Top 10 Detected Anomalous Flights:
                        carrier_name_original airport_original  arr_delay  \
644                      SkyWest Airlines Inc.              LFT     1437.0
7987                     SkyWest Airlines Inc.              DSM     1435.0
4671  GoJet Airlines LLC d/b/a United Express              XNA     1435.0
3268                     SkyWest Airlines Inc.              PIB     1435.0
801                          Piedmont Airlines              PIT     1434.0
3184                     SkyWest Airlines Inc.              GRI     1425.0
657                      SkyWest Airlines Inc.              MEI     1425.0
6922               CommuteAir LLC dba CommuteAir              LBB     1424.0
8137                     SkyWest Airlines Inc.              SJT     1424.0
2376                   Delta Air Lines Network              ELP     1424.0

      late_aircraft_delay  weather_delay  carrier_delay  nas_delay
644                 285.0          302.0          652.0      198.0
7987                 68.0          503.0          659.0      205.0
4671                414.0          709.0          163.0      149.0
3268                212.0          632.0          259.0      332.0
801                 625.0          332.0           99.0      378.0
3184                119.0          491.0          714.0      101.0
657                 392.0          304.0          694.0       35.0
6922                335.0          666.0          306.0      117.0
8137                102.0          208.0          883.0      231.0
2376                261.0          253.0          698.0      212.0
```


Top 10 Airports vs Delay Causes (Average Minutes)
Delay Cause
carrier_delay 429.4 596.8 533.0 176.0 423.3 513.5 305.4 0.0 541.8 498.0
weather_delay 220.8 370.2 152.6 66.2 291.5 223.1 489.0 503.6 230.4 320.2
nas_delay 339.4 288.6 384.0 50.8 338.1 263.3 163.4 702.4 349.2 268.0
security_delay 5.0 0.0 0.0 0.0 0.0 3.2 0.0 0.0 0.0 0.0
late_aircraft_delay 956.6 839.8 861.6 779.8 817.4 766.3 804.2 748.8 765.2 842.8
ABI BQK COD FLO MLU OAJ PRC SGU TTN USA
Airport
800
600
400
200
0


Top 10 Airports by Average Late-Aircraft Delay
1000
800
600
400
200
0
Average Delay (minutes)
ABI
COD
USA
BQK
MLU
PRC
FLO
OAJ
TTN
SGU
Airport

Results and Interpretation:

The analysis shows that flight delays can be accurately modeled using a small set of operational and temporal features, including carrier, airport, and month of flying. Each of these variables showed consistent strong explanatory power across the models, highlighting clear seasonal and location-specific delay dynamics. Late-aircraft delay was the most significant driver in both extreme delay events and anomaly detection. This is consistent with industry observations about how delays propagate through the network: if an aircraft falls behind schedule, the downstream impact is often large delays. The process of anomaly detection, through DBSCAN in particular, captured this dynamic as normal operations were clustered tightly, while extreme late-aircraft incidents were isolated as noise points. These anomalous flights corresponded mostly to severe weather disruptions, cascading delays, or operational breakdowns at busy hubs.

Airport-level patterns were also highly pronounced. DFW and ATL consistently appeared as centers of elevated anomaly activity, suggesting that high traffic volume and susceptibility to weather-related congestion at these locations heightens vulnerability to operational disruptions. The clustering structure shows that spikes in the delays at these airports don't happen randomly but form identifiable pockets of abnormal behavior that would reflect temporary collapses in throughput or sudden surges in demand. Understanding these pockets of concentrated anomalies helps point out when and where the system is most vulnerable.

We consider Logistic Regression to be the strongest of the predictive models due to its superior balance of interpretability, stability, and performance. Whereas sometimes higher complexity models yield minor improvements in performance, the logistic model showed strong accuracy and recall, with transparent coefficients that reflected feature contributions. This interpretability is crucial for aviation stakeholders who need to understand and justify the factors driving delay predictions. These results have implications extending far beyond academic modeling. Airlines and airports can use these predictive signals to anticipate periods of congestion, adjust scheduling buffers, and strategically deploy staff during known risk windows. Maintenance teams receive early warnings of possible aircraft turnaround delays, thus enabling more proactive interventions. Passengers, on their part, can expect a clearer view of the likelihood of delays based on routes, carriers, and seasons. In sum, these combined predictive models and anomaly detection provide a workable framework to reduce operational uncertainty and enhance decision-making for the aviation ecosystem.

Code Structure:

Our code was structured in a way that each step logically followed the workflow to manage, understand, and refine the entire project through its stages of analysis. Since much of our work was performed inside a Jupyter Notebook, we kept the code organized in clean sections, one devoted to a specific part of the project. This further allowed us to immediately view outputs, catch mistakes a lot earlier, and adjust our models and pre-processing steps in a manner that did not require rewriting large portions of the code.

We started with a standard import section that loaded all the libraries on which we depended in this project: pandas for data manipulation, NumPy for numeric computation, Seaborn and Matplotlib for visualization, and scikit-learn to build and evaluate machine-learning models. Keeping these imports together at the top made our dependencies easy to see and kept our runtime environment consistent.

Following the import block, we had a whole section devoted to loading and exploring the data. That was where we read our final filtered dataset in and carried out some initial checks: looking at the first few rows, the data types of columns, and confirming that our chosen six-month subset had the correct number of rows. This early exploration also helped us verify that the BTS files we downloaded were formatted correctly after filtering and merging.

The preprocessing section was the longest and most involved part of the notebook. Here, we handled everything that would get the raw data ready for modeling: cleaning invalid or missing values, label encoding of the categorical variables, including the carrier and airport, selecting only the features we want to use for prediction, and then creating our binary target variable, delayed. This portion was very vital because some of our models, such as Logistic Regression and KNN, require strictly numerical inputs. We also performed the train-test split here so the models would be evaluated fairly without any data leakage.

Once the data was fully prepared, the notebook transitioned into a clearly separated modeling section. For each model, there was a block of code for initializing and training, and then one for prediction. We first considered Logistic Regression as a baseline; then we added a Random Forest model to improve performance and capture nonlinear relationships. Later in the project, we incorporated K-Nearest Neighbors into this section in order to extend our comparison of model behaviors. Since each model was in its own block of code, it was easy to compare the results of each without mixing code between sections. Finally, all the accuracy, precision, recall, and classification reports for each model were grouped together in the evaluation section. This made interpretation of how well each model performed on the test data easy and allowed us directly to compare outputs side by side. Having the evaluation results in one place really helped us tie the findings back to our original problem statement and identify which modeling approach worked best for our delay predictions.

References:

https://www.bts.gov/

- Python Libraries:
  - Pandas
  - NumPy
  - Seaborn
  - Matplotlib
  - Scikit-Learn

- Tools:
  - Jupyter Notebook
  - Visual Studio Code
  - GitHub

Kenyen Hast

My main contributions to this project centered on planning our workflow, preparing the data we needed, setting up SQL as our storage system, and fixing issues that came up during development. Early in the project, I helped our group decide how we wanted to manage our files and where the data would come from. We agreed that using a local SQL database would give us more control over the data and make it easier to update the dataset later. I created the initial plan for how we would move the raw BTS flight data into a format that our models could use. This included deciding which fields we needed, how the data should be cleaned, and how it should be stored in the database. Another important part of my work involved locating the correct delay fields and understanding how they were measured. The BTS dataset has several delay categories, such as late aircraft, weather, and security. I helped the group decide which values were meaningful for predicting a fifteen-minute delay. I also assisted in reducing the dataset size. The original data was too large, so I worked with the group to bring it down to a six-month range that still kept enough samples for training. This required reviewing the BTS files, checking row counts, and making sure we did not remove anything important for predictive accuracy. During development, I also spent time fixing bugs and merge errors. When we combined code from different team members, the database connection broke because the structure of the table had changed. I rebuilt the SQL table, rewrote the connection functions, and corrected column name mismatches so the code would run again. I also checked for data leakage problems after noticing that one model had suspiciously high accuracy.

Ryan Blankenbeker

For this project, we first had to decide on a topic and begin searching for datasets that met the requirement of containing between 7,000 and 10,000 records. After discussing different options, we agreed on developing a Flight Delay Prediction model. To gather the necessary data, I explored the Bureau of Transportation Statistics (BTS) website and began narrowing down the available datasets. At first, I downloaded nearly two years of data, which resulted in several hundred thousand rows. After filtering and reducing the dataset to a manageable size that still preserved the integrity of the problem, I refined it to just under 10,000 records spanning approximately six months.

Once the dataset was finalized, I began drafting the initial Python scripts to create a baseline structure before transferring the workflow into the Jupyter Notebook. We started with two core models—Logistic Regression and Random Forest Classifier—because they provided both a statistical baseline and a more sophisticated ensemble method. In implementing these models, we learned how data leakage can artificially inflate performance metrics and why proper preprocessing is essential.

Throughout the modeling process, I also contributed to the expansion of our machine learning approach by researching and adding additional techniques, such as K-Nearest Neighbors. This required experimenting with different feature selections, analyzing class imbalance issues, and refining the preprocessing workflow to ensure valid model training. Overall, this project strengthened my understanding of end-to-end data science workflows.

John W Grant Wallace

We originally started this project “individually” and we had a difficult time getting everyone on the same page for the work that needed to be done. Although after the first couple of meetings we set up a GitHub and googledocs to keep everyone organized and from there we were able to do much better in terms of organization. Ryan and I both worked on separate python code at first and then we had to go through and try and combine both of the projects. After we got that done I noticed that our accuracy was 1, which was a red flag to me so I looked into it. Because we gave our model the actual 15 minute delay column from the Beuro of Transportation Statistics, the model wasn't really predicting 15 minute delays as much as spitting them back out from the raw data. Once that was sorted, we worked on the presentation slides as a group and then I ended up taking all of the data and putting it into powerpoint to make it look better. After we presented, I think we got a much better understanding of where this project needed to end up, so with our last few days I was able to restructure a lot of the code to include more meaningful data, as well as add / edit our visualls to be more meaningful. In the process Ryan was also able to add the KNN model which we were told is more accurate than the initial models we went with at the start of our project. Then Kenyen got the SQL working again because I accidentally messed it up during one of the merges. After the project was pretty much buttoned up on the code side of things we changed the focus to completing the report. Adding the visuals from our code, expounding on why our metrics ment something and how that affected travelers flight times. Over all I think this was a great experience into a start to finish real world project and I learned a lot from it. Again I think working together was the hardest part, but now that were coming to a close on the project I think weve gotten on a roll and I’m glad to have been able to work with such fine teammates.