The University of San Diego, Knauss School of Business

From Delays to Decisions: Predictive Insights for U.S. Airlines

Valeria Breton, Lilie Catania, and Liza Scott

BUAN 381- Predictive Analytics & Big Data
Dr. Steven Levkoff

May 18, 2025

Executive Summary

This project set out to explore how predictive modeling could be used to estimate flight departure delays using U.S. flight data. We started by cleaning and analyzing a large dataset, then we created models using numeric and character variables to estimate both the likelihood and potential severity of delays. Key variables included day of the week, airline performance (ranked by average delays), and flight departure groupings such as morning, afternoon, and night. These variables were chosen based on both real-world relevance and early patterns noticed during exploratory analysis. The data used for this project was sourced from the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics accessed through Kaggle. It included over 5.8 million records of domestic flights within the United States from 2015. The dataset contains a wide range of variables such as flight times, delay durations, carrier codes, origin and destination airports, and specific causes of delay (e.g., weather, air system, security).

The main goal was to understand how delay predictions could help airlines make more informed operational decisions, such as optimizing staffing levels, adjusting flight schedules, improving customer communication, and managing resources more efficiently. The project highlights how prediction tools can support proactive planning and smarter resource use in airline operations.

Our raw dataset contained 5,819,079 records and 31 observations. The numeric variables included:

VARIABLE	DESCRIPTION
YEAR	The year the flight took place
MONTH	The month the flight took place
DAY	The day of the month the flight took place
DAY_OF_WEEK	The day of the week the flight took place
FLIGHT_NUMBER	The flight number
DEPARTURE_DELAY	Number of minute by which a flight was delayed at departure
TAXI_OUT	Number of minutes between the flight leaving the gate and wheels off
SCHEDULED_TIME	The amount of time the flight is planned to take

ELAPSED_TIME	Number of minutes to taxi out of departure gate + number of minutes spent in the air + number of minutes to taxi to arrival gate upon landing
AIR_TIME	The number of minutes between wheels off and wheels on
DISTANCE	The distance between two airports in miles
TAXI_IN	The number of minutes between a plane's landing and arrival to the destination gate
ARRIVAL_DELAY	The number of minutes by which a plane is late to arrive at its destination gate
DIVERTED	A 1 indicates the plane had to land at an airport different from the one originally scheduled, a 0 means the plane did not have to be diverted
CANCELLED	A 1 indicates the flight was cancelled, a 0 indicates the flight was not cancelled
AIR_SYSTEM_DELAY	Number of minutes of delay caused by the air system
SECURITY_DELAY	Number of minutes of delay caused by security
AIRLINE_DELAY	Number of minutes of delay caused by airlines
LATE_AIRCRAFT_DELAY	Number of minutes caused by an aircraft arriving late to depart
WEATHER_DELAY	Number of minutes of delay caused by weather conditions

The character variables included:

VARIABLE	DESCRIPTION
AIRLINE	The name of the airline the flight is apart of
TAIL_NUMBER	The unique code used to identify a specific plane
ORIGIN_AIRPORT	The airport from which the plane is departing
DESTINATION_AIRPORT	The airport at which the plane will arrive
SCHEDULED_DEPARTURE	The time at which the plan is expected to have wheels-off
WHEELS_OFF	The time at which the plane was wheels off
WHEELS_ON	The time at which the plane was wheels on
SCHEDULED_ARRIVAL	The time at which the plane was scheduled to arrive at the final destination
ARRIVAL_TIME	The time at which the plane actually arrived at the final destination
CANCELLATION_REASON	A letter indicating the circumstances under which the plane was cancelled. A: Airline / Carrier, B: Weather, C: National Air System, D: Security

Data Cleaning

Investigating Missing Values

Our data cleaning process was evolving as we became more and more familiar with the dataset and how its features would be helpful for our analysis.

```
> dim(flights) #Starting dimensions
[1] 5819079      31
> sum(is.na(flights)) #starting number of NAs
[1] 30465274
```

Our first step was to investigate the number of missing values in the raw dataset. We needed to evaluate reasons why they might be missing and then determine the best course of action to navigate them with minimal amounts of information lost. We can see that out of the 5.82 million observations in the raw dataset, there were 30,465,274 missing values. We knew we had to address these before performing any further analysis.

It is also important to note that our data was not balanced and contained outliers. We learned towards the end of our project that we would have gained better insights on our model performances if we had conducted outlier cleanings, to prevent our training models from trying to overfit them. If our data was balanced at any point during the modeling process, it was benchmarked against the other models on the same balanced data and included a summary of performance on unbalanced data as well. If a model, such as our support vector regression model, was run on a different partition or balanced partition of the dataset, it was noted in this report and was not benchmarked against the other models who were trained on the unbalanced, clean dataset.

Removing the CANCELLATION_REASON Field

When taking a brief view over the dataset, the column for the CANCELLATION_REASON variable stood out in particular due to the amount of NAs present. We found that in the entire column, 98.46% of the observations were NAs. We were able to determine that this was because the flights were not cancelled at all, therefore they were not marked for a cancellation reason code. We decided to remove this column altogether because our predictive question was concerned with departure delays, not cancellations. This kept the number of records in the dataset the same, 5,819,079, the same and reduced the number of fields by 1, for a total of 30. Even after the removal of this column, the dataset still contained 24,736,079 missing values.

```
> na_columns[na_columns > 0]
       TAIL_NUMBER     DEPARTURE_TIME     DEPARTURE_DELAY            TAXI_OUT          WHEELS_OFF
             14721              86153              86153              89047              89047
    SCHEDULED_TIME       ELAPSED_TIME           AIR_TIME          WHEELS_ON            TAXI_IN
                 6             105071             105071              92513              92513
      ARRIVAL_TIME      ARRIVAL_DELAY  AIR_SYSTEM_DELAY     SECURITY_DELAY      AIRLINE_DELAY
             92513             105071            4755640            4755640            4755640
LATE_AIRCRAFT_DELAY     WEATHER_DELAY
            4755640            4755640
```

Following the removal of this column we looked at an overview of the column names who still contained rows with missing values, and the number of missing values in them.

Cleaning the DEPARTURE_DELAY Field

We decided an important next step to take would be to remove all records who had a missing value for DEPARTURE_DELAY because this was our variable of interest. We noticed that the number of missing values in this column was equal to the number of records that had values for the cancellation flags in the field we previously removed. We were able to infer that they were marked as NA in the DEPARTURE_DELAY column because the flight had never taken off due to being cancelled.

```
> dim(flights_no_na)
[1] 5732926      30
```

Following the removal of records with missing departure delay values, we observed that we lost approximately 86,153 rows. Although this is a large number of rows, it only amounted to 1.48% of our total dataset being lost. There were still 23,429,063 missing values in the dataset across the following fields:

TAXI_OUT	WHEELS_OFF	SCHEDULED_TIME	ELAPSED_TIME	AIR_TIME
2894	2894	6	18918	18918
WHEELS_ON	TAXI_IN	ARRIVAL_TIME	ARRIVAL_DELAY	AIR_SYSTEM_DELAY
6360	6360	6360	18918	4669487
SECURITY_DELAY	AIRLINE_DELAY	LATE_AIRCRAFT_DELAY	WEATHER_DELAY	
4669487	4669487	4669487	4669487	

Cleaning of the AIR_TIME Field

We then moved onto cleaning the AIR_TIME column, because if no values were logged for the amount of time the flight had spent in the air, we could reasonably assume that the flight had never departed or gone in the air at all. This would indicate a bigger problem, such as a cancellation or a diversion. Although these are also consequences of airport misfortunes, we were interested in predicting strictly departure delays, which is why we were able to safely remove records from this field. Following this cleaning step, our dataset was left with 5,714,008 records and 30 fields. 23,252,845 missing values remained throughout the data frame.

```
> na_columns[na_columns > 0]
   AIR_SYSTEM_DELAY      SECURITY_DELAY       AIRLINE_DELAY LATE_AIRCRAFT_DELAY       WEATHER_DELAY
            4650569             4650569             4650569             4650569             4650569
```

After removing the records of the flights who did not depart at all, the only fields which still contained missing values were the ones of the specific kinds of delays that contribute to overall departure delay. Upon further investigation, these observations were labeled as NA because a delay of those kinds did not affect those flights at all.

Imputing Missing Values Across Delay Types

We concluded that any missing values in the AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, AND WEATHER_DELAY columns were missing because there was no delay at all. For the sake of completeness and quantifying the fact that there were 0 minutes of delay for each of them, we decided to impute all of the NAs with the number 0.

```
> # INMPUTE NUMBER 0 FOR NAS IN DELAY TYPE FIELDS
> library(tidyr)
> flights_no_na <- flights_no_na %>%
+   replace_na(list(
+     AIR_SYSTEM_DELAY = 0,
+     SECURITY_DELAY = 0,
+     AIRLINE_DELAY = 0,
+     LATE_AIRCRAFT_DELAY = 0,
+     WEATHER_DELAY = 0
+   ))
> # CONFIRM NO MISSING VALUES REMAIN
> sum(is.na(flights_no_na))
[1] 0
>
```

With this step alone, we were able to clean out 23,252,845 missing values and reduce the number down to 0. We also were able to keep the same number of rows and observations from the previous step.

Removing Unnecessary Fields

We proceeded to remove unnecessary fields from our dataset that were not going to provide robustness to our analysis, or contribute to predicting departure delays. Although this step is not always necessary in every data cleaning process, we believed that reducing the number of values that needed to be stored in our data frame would lead to improved computational efficiencies throughout the remainder of our project. We removed the field ELAPSED_TIME because we were not concerned with what happened, chronologically, after the plane left the origin airport. We also removed the field DIVERTED, because once again, these diversions happen once the plane is in the air and are irrelevant to predicting departure delays. This cleaning step left our data frame with 5,714,008 records and 27 fields.

Final Removal of Unnecessary Records

Finally, the DEPARTURE_DELAY field contained both positive, zero and negative values. A negative value for minutes of departure delay indicated that the plane had left earlier than its scheduled departure time, a value of 0 indicates that a plane left exactly on time, and a positive value represents the number of minutes by which the plane had a delayed departure. We decided to remove all of the records where departure delay was less than 0 minutes, because our business implications to our questions were not concerned with early departures. We only wanted to look at flights that were delayed as those were the ones that would end up costing their airlines, and flights that were right on time to make comparisons and benchmark performance. This left a clean dataset with 2,443,491 records and 27 fields.

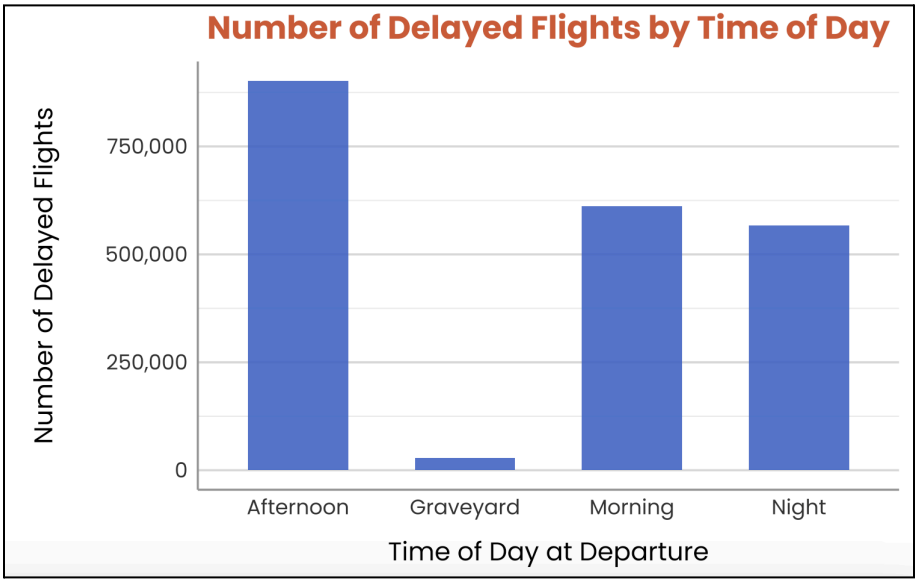
Partitioning the Data

We partitioned the data using a 70-15-15 split. The training dataset contained 70% of the cleaned observations and was selected by randomly shuffling the row numbers and creating an index of the random row numbers to subset the training set. The holdout partition was then

designated from the remaining rows left after the training partition was established. The same process was implemented, this time with only 50% of the remaining 30% of the data going to each partition. We then used simple calculations to ensure that the partitions were actual 70-15-15. Finally to ensure reproducibility, we wrote the partition into separate .csv files that would be used throughout our model building, validation, and testing processes.

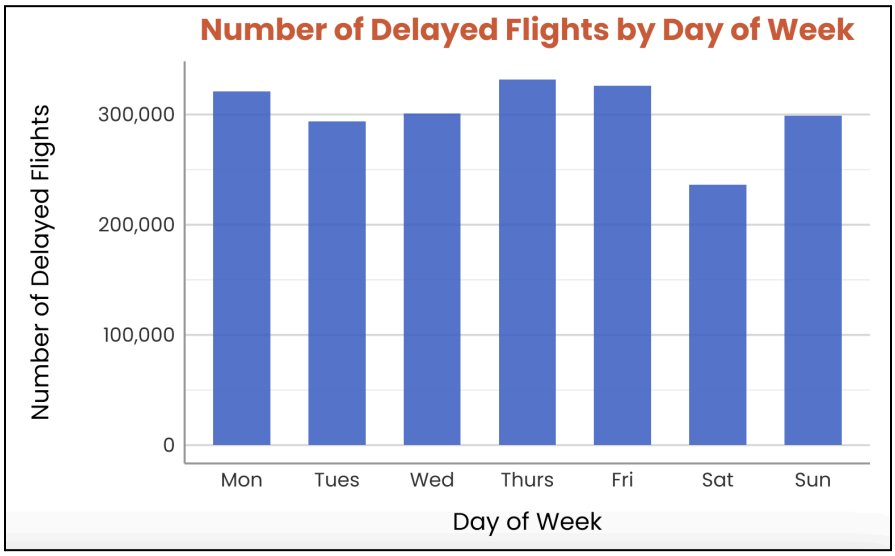
Exploratory Analysis

Before jumping into our predictive modeling, we began with an exploratory analysis to better understand the structure and behavior of our dataset. While we weren't drawing formal conclusions at this stage, our goal was to establish a business mindset and identify potential operational patterns and pain points that airlines might care about. In particular, we focused on uncovering when, where, and why delays were most likely to happen. These patterns would later help guide which predictors to include in our models. Ultimately, this part of our analysis focused on the real-world implications of staffing decisions, scheduling strategies, and customer experience. The following visualizations provided a foundation for making those connections.

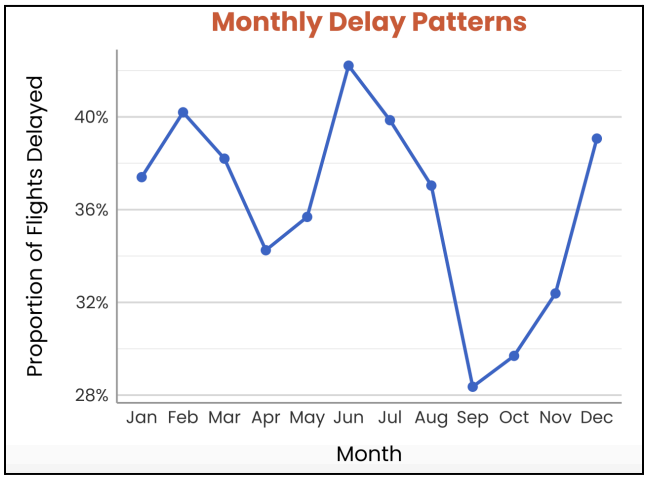


We created the visualization shown above to examine the relationship between time of day and departure delays. Flights were binned into four distinct categories based on their scheduled departure time using a 24-hour format: Graveyard (00:00–05:59), Morning (06:00–11:59), Afternoon (12:00–17:59), and Night (18:00–23:59). What stood out immediately was the disproportionate number of delays in the afternoon. This pattern likely reflects what is often referred to as the "operational snowball effect." Disruptions early in the day tend to spill over, creating compounding issues for flights scheduled later. In contrast, graveyard flights had the fewest delays, which makes intuitive sense: fewer scheduled flights, minimal air traffic

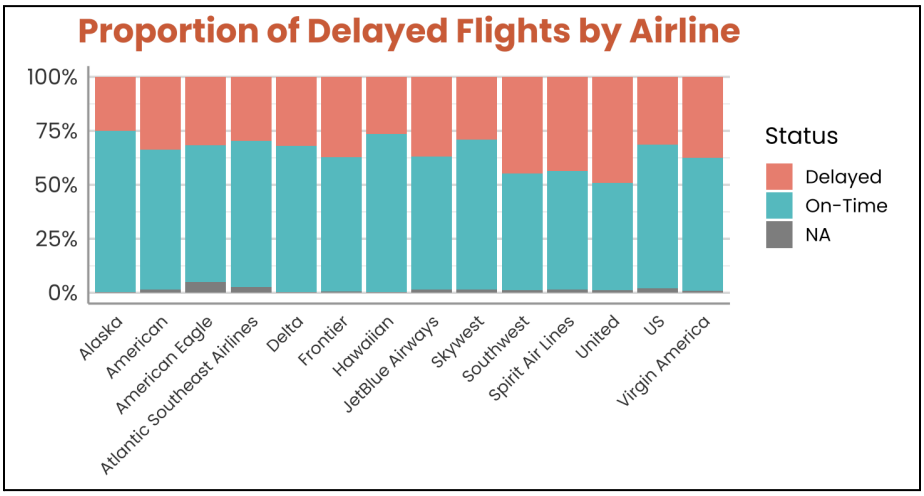
congestion, and airport noise curfews all contribute to smoother operations during these hours. It's also important to note that our dataset's total number of graveyard flights was significantly lower, partly explaining the steep drop in volume. While we aren't making causal claims from this chart alone, the trend helped reinforce the importance of departure time as a predictive feature in our classification models. From a business standpoint, this pattern could be valuable for flight scheduling teams, encouraging them to buffer afternoon routes or address bottlenecks to minimize worsening delays.



To identify whether certain days of the week are more prone to delays, we visualized the total number of delayed flights by weekday. From the chart above, Thursdays and Fridays stand out as the most delay-heavy days, with each surpassing 330,000 delays in our dataset. Mondays follow closely behind, while Saturdays see a noticeable drop, recording the fewest number of delayed flights overall. While we can't draw statistical conclusions from this alone, this trend aligns with operational realities in the airline industry. Business travel tends to spike on Mondays and Thursdays, while Friday delays may reflect increased leisure travel and end-of-week congestion. On the other hand, Saturdays likely had fewer delays because fewer people tend to travel on that day; most weekend travelers have already reached their destinations by then, leading to lighter flight schedules and less congestion. This pattern helped confirm that the day of the week was worth retaining as a feature in our later predictive models. From a business perspective, this insight could inform staffing schedules and gate availability planning, especially during high-volume travel periods when even minor disruptions can escalate quickly.

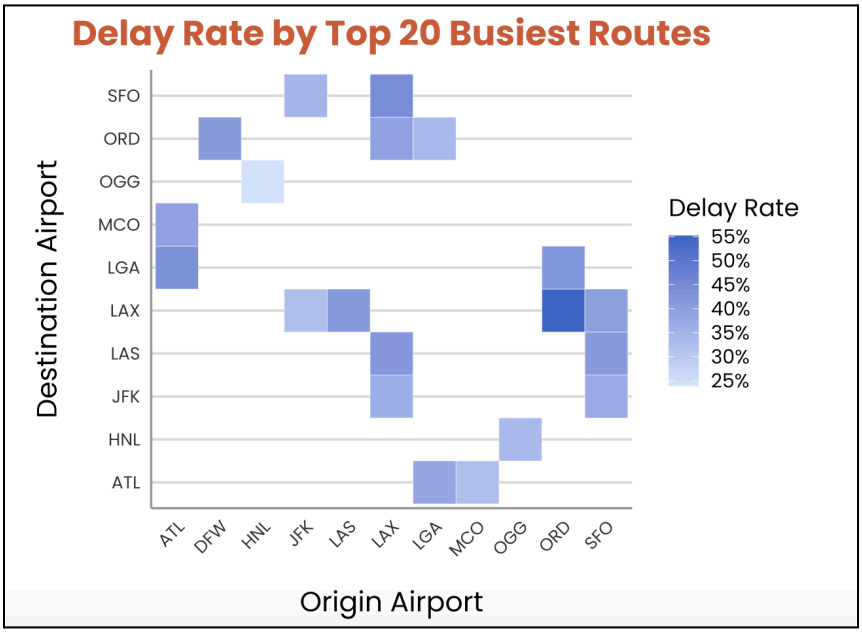


This line graph shows how the proportion of delayed flights fluctuated across the calendar year. Unsurprisingly, we observed clear spikes in June and July, which align with the summer travel season and its associated surge in passenger volume and weather-related disruptions like thunderstorms. The sharp dip in September and October represents the travel industry's shoulder season as demand is lower, and weather conditions are generally more stable, leading to fewer delays overall. As expected, delay rates began climbing again in November and December, likely due to holiday travel traffic and the onset of winter weather conditions. Even though not all seasonal patterns were ultimately included as predictors in our final models, visualizing these trends early on was helpful in framing the broader operational environment. For airlines, these fluctuations could inform seasonal staffing, capacity planning, and proactive customer communication strategies.

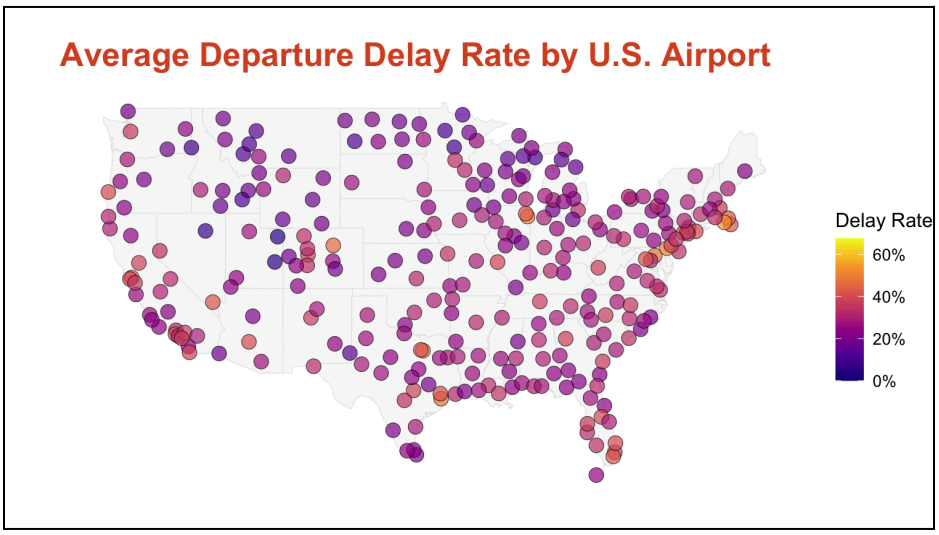


In this chart, we plotted the proportion of delayed versus on-time flights within each airline carrier, which gave us a sense of how frequently delays occurred relative to each airline's total number of flights rather than by raw volume. Again, while we're not making any causal

claims at this stage, clear patterns began to emerge. Some airlines, like Hawaiian and Alaska, maintained relatively high on-time rates, whereas others (such as Southwest, Spirit, and United) showed a noticeably higher proportion of delays. These trends suggested there may be underlying differences in internal operational factors, such as staffing levels, route logistics, and overall workflow efficiency. For example, airlines with better turnaround processes or more flexible crew scheduling may be better equipped to recover from unexpected disruptions. Similarly, carriers operating out of highly congested hubs or flying more delay-prone routes could face additional challenges that impact on-time performance. From a modeling standpoint, this analysis helped us recognize that airline identity could be a meaningful predictor. These early proportions laid the groundwork for our decision to incorporate airline-based ranking variables and explore whether certain carriers consistently underperformed.



This heatmap highlights the delay rates for our dataset's top twenty most frequently traveled routes. While the intensity of delays varied across different origin-destination pairs, some high-traffic hotspots, particularly those involving LGA, ORD, and SFO, consistently showed elevated delay rates. This visualization reinforces what we began to observe earlier: congestion levels at major airports likely contribute to increased delays. Although this chart doesn't give us the full context behind every route's delay behavior, it helped guide our thinking in terms of including both origin and destination airports as potential predictors in our predictive models.



This map visualizes the average departure delay rates at airports across the United States. While our earlier heatmap focused on the top twenty busiest routes, this visualization offered a broader, more visually digestible overview of regional delay patterns (useful for presentation purposes). We observed clear geographic trends with airports in the Northeast, California, and the Southeast consistently experiencing higher average delay rates. This makes sense, given that these regions are home to some of the largest airports in the United States. That said, smaller Midwest airports were not immune to delays; several showed elevated delay rates, which may reflect weather variability or resource limitations at less-equipped hubs. From an operational lens, this visualization reinforced the idea that delay-prone regions may benefit from more mitigation efforts through increased staffing, more efficient gate management, or revised flight schedules. While this chart reflects earlier findings, it provided an accessible way to highlight spatial trends and support our rationale for wanting to include origin and destination airport data as modeling features.

Bivariate Regression Modeling

Correlation Table

Variable	Cor_Departure_Delay
DEPARTURE_DELAY	1.00000000
ARRIVAL_DELAY	0.96318133
AIRLINE_DELAY	0.65516525
LATE_AIRCRAFT_DELAY	0.62648670
AIR_SYSTEM_DELAY	0.28217188
WEATHER_DELAY	0.26129387
TAXI_OUT	0.05205599
TAXI_IN	0.03877155
SECURITY_DELAY	0.02299526
DISTANCE	-0.02201989
SCHEDULED_TIME	-0.01505878

Our partitions had 27 fields, 11 of which were numeric. This meant that our output from the cor() function was an 11x11 matrix that was difficult to read. Our variable of interest was DEPARTURE_DELAY, so we decided to extract only the correlations of that variable, and sort them by their absolute values in descending order. This made it easier to inspect which variables had the strongest relationships with DEPARTURE_DELAY. It can be seen that the variable ARRIVAL_DELAY had the strongest relationship with DEPARTURE_DELAY, however we did not use this variable in our models, as any destination arrival delays are symptoms of departure delays at the origin. It also did not make sense chronologically, as arrival delays will always occur after the departure delays. This led us to move forward with choosing the variable with the second strongest association, AIRLINE_DELAY, for our bivariate regression models.

Bivariate Linear Regression Model

```
Call:
lm(formula = DEPARTURE_DELAY ~ AIRLINE_DELAY, data = flights_train)

Residuals:
    Min      1Q  Median      3Q     Max
-103.75  -18.92  -12.86    0.10 1437.08

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.992e+01  2.996e-02     665   <2e-16 ***
AIRLINE_DELAY 9.982e-01  8.801e-04    1134   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 38.01 on 1710441 degrees of freedom
Multiple R-squared:  0.4292,    Adjusted R-squared:  0.4292
F-statistic: 1.286e+06 on 1 and 1710441 DF,  p-value: < 2.2e-16
```

Model

This model is regressing DEPARTURE_DELAY on AIRLINE_DELAY (coded as M1). It seeks to predict the number of minutes by which a flight will be delayed based on the minutes of delay its airline is experiencing.

Coefficients

The model’s intercept can be interpreted as the mean minutes of departure delay when there is no airline delay. In this case, when airline delay is zero minutes we would expect to see an average departure delay of 19.92 minutes. The coefficient on AIRLINE_DELAY represents that a one minute increase in airline delay is associated with a 0.998 minute increase in departure delay, on average, holding all else constant.

Diagnostics

It can be seen that AIRLINE_DELAY’s coefficient is statistically significant at the 99.99% confidence level, meaning it is highly unlikely that the relationship we are observing is due to chance. The same statistical significance can be seen on the intercept term. The model’s F-statistic is very large, thereby rejecting the null hypothesis that a relationship does not exist between airline delays and departure delays.

The model’s R-squared is 0.4292, this indicates that the AIRLINE_DELAY variable is explaining 42.92% of the variability in DEPARTURE_DELAY, in sample. This also means that due to the low complexity of this model, it is unable to explain approximately 52% of the variability in minutes of departure delays.

Bivariate Quartic Regression Model

```
Call:
lm(formula = DEPARTURE_DELAY ~ AIRLINE_DELAY + AIRLINE_DELAY2 +
    AIRLINE_DELAY3 + AIRLINE_DELAY4, data = flights_train)

Residuals:
    Min      1Q  Median      3Q     Max
-104.48  -18.84  -12.84    0.16 1437.16

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.984e+01  3.112e-02  637.69   <2e-16 ***
AIRLINE_DELAY   1.025e+00  2.358e-03  434.47   <2e-16 ***
AIRLINE_DELAY2 -2.250e-04  1.486e-05  -15.14   <2e-16 ***
AIRLINE_DELAY3  3.640e-07  2.247e-08   16.20   <2e-16 ***
AIRLINE_DELAY4 -1.465e-10  9.216e-12  -15.90   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 38.01 on 1710438 degrees of freedom
Multiple R-squared:  0.4293,    Adjusted R-squared:  0.4293
F-statistic: 3.217e+05 on 4 and 1710438 DF,  p-value: < 2.2e-16
```

Model

This model is attempting to capture non-linear relationships between minutes of airline delay and minutes of departure delay, by regressing DEPARTURE_DELAY on a fourth order polynomial transformation of the AIRLINE_DELAY variable (coded as M2.1). We chose to do a quartic model because there was little to no improvement in model diagnostics for the quadratic and cubic regression models we estimated beforehand, relative to the linear bivariate regression model.

Coefficients

The coefficient on AIRLINE_DELAY represents that we would expect to see an increase of approximately 1.03 minutes in DEPARTURE_DELAY for each one minute increase in AIRLINE_DELAY, on average holding all else constant. The intercept coefficient represents that we would expect average minutes of departure delay to be 19.84 minutes if the flight's airline is not experiencing delays. The polynomial terms quadratic term is negative, indicating that the increasing effect the AIRLINE_DELAY variable has on DEPARTURE_DELAY is being decelerated, the quartic term is negative because it is stabilizing the upward curvature of the cubic term to prevent DEPARTURE_DELAY from increasing uncontrollably for larger values of AIRLINE_DELAY.

Diagnostics

Once again, all of the model's coefficients are statistically significant beyond the 99.99% confidence level. The model's F-statistic is still very large, informing us that there is a statistically significant relationship that exists between the AIRLINE_DELAY terms and the DEPARTURE_DELAY variable.

The R-squared of 0.4293 indicates that by estimating complex, non-linear relationships between minutes of airline delay and minutes of departure delay, has increased the bivariate model's explanatory power by 0.01% (compared to the linear bivariate model).

```
> TABLE_VAL_2b
             LINEAR  QUARTIC
RMSE_IN   38.01200 38.00906
RMSE_OUT 38.16955 38.16800
```

We can benchmark the improvement of our bivariate regression model's performance by comparing the in-sample and out-of-sample RMSEs of each model. We can see that by adding complexity with a 4th order polynomial transformation on the AIRLINE_DELAY variable, we have improved the model's ability to predict both in sample and out of sample by a *very* small amount. The linear model's in-sample predictions for minutes of departure delay were off by 38.012 minutes on average while the quartic model's were off by 38.009 minutes on average. This means that the average in-sample error of predicted minutes did not improve by even a whole second. The linear model's out of sample predictions for departure delay were off by 38.170 minutes on average while the quartic model's were off by 38.160 on average. Again, this is a very small improvement in the context of our predictive question. It is worth noting that the in-sample and out-of-sample errors are very close to each other, for both models. This may suggest that our model is not overfitting to the training data and is estimating a very strong predictive relationship.

Attempting to Regularize the Quartic Bivariate Regression Model

lmridge()

Although there are many ways to perform RIDGE regression in R, we found that many methods were computationally expensive to run given the size of our partitions. To regularize our nonlinear polynomial regression model we decided to leverage the lmridge() package in R. It is important to note that we were not able to use this package to run over a sequence of lambda values, however, we were able to use it to manually tune the hyper parameter "K" in order to explore the effect of regularization on the model.

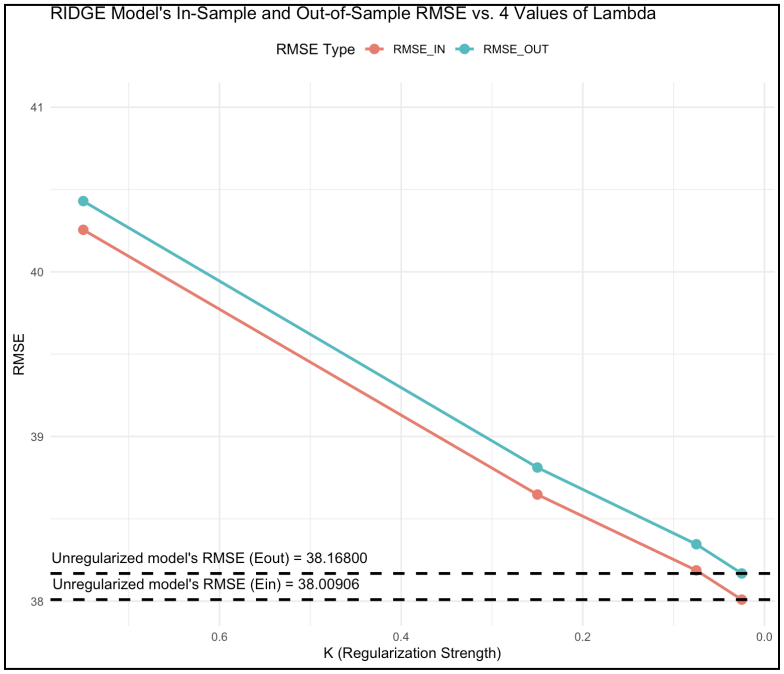
Method

We ran 4 RIDGE regression models, where K was decreased by 66% each time. The four lambda values we set were K = 0.75, K=0.25, K = 0.075, and K = 0.025. We then made predictions on our training and hold out partitions and calculated the in-sample and out of sample-errors for all four models.

Regularization Performance

```
> TABLE_VAL_RIDGE
          K=0.75   K=0.25  K=0.075  K=0.025    UNREG
RMSE_IN  40.25567 38.64745 38.18686 38.06209 38.00906
RMSE_OUT 40.42987 38.81153 38.34554 38.22019 38.16800
```

This table is comparing the in-sample and out-of-sample error for each lambda value. The unregulated model, (Quartic, M2.1) was included to benchmark the performance of the RIDGE models. It can be seen that the stronger the regularization was, the worse the model performed. This was to be expected, because our out-of-sample error in the unregularized model was hardly different from the in-sample error to begin with. The benefits of regularization are best observed when out-of-sample error is much higher than in-sample error, due to a model's inability to generalize well, which is why we do not see them here.



The graph above is a visual representation of the declining in-sample and out-of-sample errors as regularization strength decreases from left to right. The dashed lines in black show the in and out of sample errors of the unregularized model. This graph shows us how the errors of the RIDGE models are converging to those of the unregularized model, indicating that implementing regularization had no effect on improving the model's performance.

Implementing SPLINE

```
Family: gaussian
Link function: identity

Formula:
DEPARTURE_DELAY ~ s(AIRLINE_DELAY)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 28.15487    0.02906   968.8   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                   edf Ref.df      F p-value
s(AIRLINE_DELAY) 8.685   8.96 143644  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.429   Deviance explained = 42.9%
GCV = 1444.6  Scale est. = 1444.6     n = 1710443
```

Model

This SPLINE model uses a smooth curve to demonstrate the relationship between AIRLINE_DELAY and DEPARTURE_DELAY. The link function output tells us that the model is predicting the actual units of departure delay minutes and that no non-linear transformations were applied.

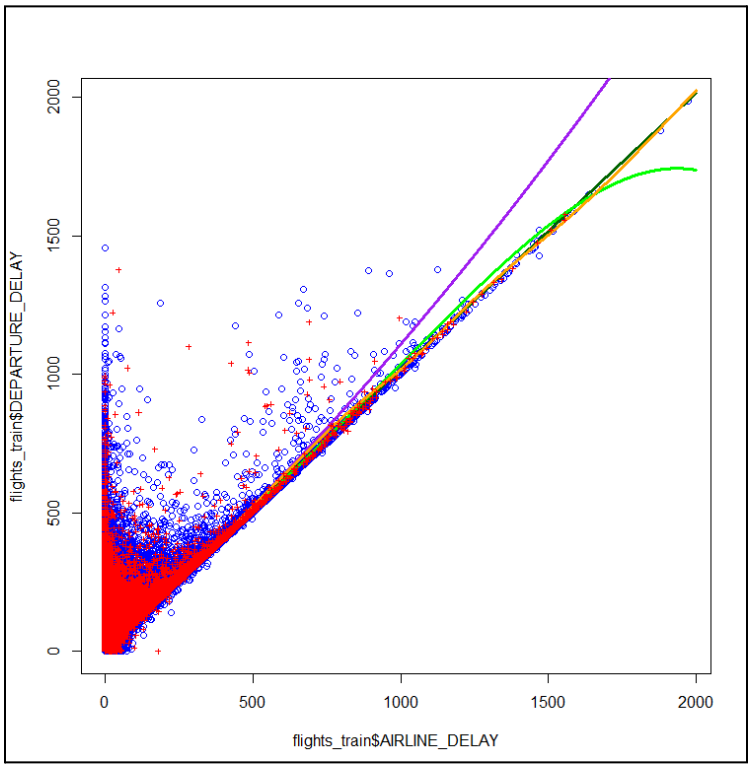
Coefficients

The model’s estimated parameter for the intercept is 28.155. This model indicates that when AIRLINE_DELAY is 0, we estimate DEPARTURE_DELAY to be approximately 28 minutes, on average .It can be seen that the intercept value is statistically significant beyond the 99.99% confidence level. The t-statistic is large and the p-value is very small, informing us that the SPLINE model’s estimate for the intercept term is significantly different from 0. The F-statistic on the s(AIRLINE_DELAY) term informs us that the smooth linear curve is outperforming a linear model in-sample.

Diagnostics

The model’s deviance explained and adjusted R-squared are still moderately low and similar to the other bivariate models. There is still only 42.9% of the variability in minutes of departure delays being explained, and in-sample fit is moderate. This also tells us that there is room to add more features on the right hand side to improve how well the model captures the variability in departure delays. Despite the R-squared being similar to that of the other bivariate models, the SPLINE had the best in-sample performance due to its freedom to adapt to the training data, it is able to look at and fit small sections of the data points. It is worth noting that this model risks overfitting the data.

Bivariate Plot of Estimated Models



LINEAR	QUARTIC	RIDGE	SPLINE
Orange	Light Green	Purple	Dark Green

Generating this plot required a lot of computational power. We initialized a blank grid for the transformations of our independent variable, AIRLINE_DELAY. We then used the many points in the X grid to generate and plot our predicted minutes of AIRLINE_DELAY. The training set is shown by the blue circles, while the test set is shown by the red "+"s. The linear model is orange, and the SPLINE model is dark green. We can see that both of these models predicted the DEPARTURE_DELAY, on both partitions, similarly. The quartic model is in neon green, and demonstrated how the fourth order polynomial transformation of AIRLINE_DELAYs decelerates predicted DEPARTURE_DELAY at higher values. When we compare it to the RIDGE model in purple, we can see that it is not incorporating the 4th degree polynomial, because the cubic transformation is causing the model's predictions to exponentially increase at higher values. We can see that there are still many data points which are not captured by any of the models, indicating that we will have to include more variables on the right hand side to explain their minutes of departure delay.

In-Sample and Out-of-Sample Performance of Bivariate Models

```
> TABLE_VAL_SPLINE
           LINEAR  QUARTIC    RIDGE   SPLINE
RMSE_IN  38.01200 38.00906 38.06209 38.00713
RMSE_OUT 38.01200 38.16800 38.22019 38.16498
```

We can see that across all of our bivariate models, the SPLINE model performed the best in sample, and barely out performed the RIDGE model. The SPLINE model's predictions were off by 38 minutes on average in-sample. It is not surprising that the SPLINE out-performed the other bivariate models in-sample, because the data is driving the model's description of the relationship between AIRLINE_DELAYS and DEPARTURE_DELAYS, without any prior assumptions. This means that the spline model is the most flexible and able to capture more complex relationships and drive in-sample fit up. The RIDGE model performed the worst in sample, having an average error of 38.06 minutes in-sample, on average. This was to be expected as it was over-regularizing the quartic model, driving both in-sample and out-of-sample error up, relative to the quartic model. The linear model performed the best out of the sample because it is the least complex and therefore has the strongest ability to generalize and perform well on the partitions it was not trained on. The RIDGE model also was the worst performer out-of-sample, this is because the regularization was restricting the model's ability to adequately capture the amount of variability in DEPARTURE_DELAY that we saw in the quartic model.

Bivariate Model Selection and Reporting Estimated Out-of-Sample-Error

```
> TABLE_CHOSEN
          EIN     EOUT  E[EOUT]
RMSE 38.01200 38.16955 38.46936
```

Our linear bivariate regression model performed the best out-of-sample on the validation partition, making it our chosen model to predict DEPARTURE_DELAY by regressing it on AIRLINE_DELAY. The table above shows the model's performance across all partitions. The out-of-sample error from predictions on the validation partition and then estimated out-of-sample error on the test partition are very similar, indicating that the model is able to generalize well on new data. In the grand scheme of our purpose for predicting minutes of departure delay, the differences in error are virtually insignificant, as they do not even amount to full seconds. The closeness in error also signifies that the validation error provides good estimates for the out-of-sample error on unseen data.

Multivariate Regression Modeling

Multivariate Linear Regression Model

Model

```
Call:
lm(formula = DEPARTURE_DELAY ~ AIRLINE_DELAY + LATE_AIRCRAFT_DELAY +
    WEATHER_DELAY + AIR_SYSTEM_DELAY, data = flights_train)

Residuals:
    Min      1Q  Median      3Q     Max
-185.09   -5.29   -1.29    5.71  582.72

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.2887165  0.0098706   535.8   <2e-16 ***
AIRLINE_DELAY       0.9932436  0.0002686  3697.9   <2e-16 ***
LATE_AIRCRAFT_DELAY 0.9978065  0.0002891  3450.9   <2e-16 ***
WEATHER_DELAY       0.9356422  0.0006605  1416.7   <2e-16 ***
AIR_SYSTEM_DELAY    0.6844440  0.0004655  1470.2   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.6 on 1710438 degrees of freedom
Multiple R-squared:  0.9469,    Adjusted R-squared:  0.9469
F-statistic: 7.62e+06 on 4 and 1710438 DF,  p-value: < 2.2e-16
```

This model is regressing DEPARTURE_DELAY on AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY, and AIR_SYSTEM_DELAY. Referring back to our correlation matrix, these all had moderately strong to strong associations with minutes of departure delay. Both the dependent and independent variables' units are in minutes of delay

Coefficients

The intercept term indicates that if there are no airline, aircraft, weather, or air system delays, we would expect departure delay to be 5.29 minutes on average. The parameter estimate on AIRLINE_DELAY indicates that for every one minute increase in airline delay we would expect to see an approximate one minute increase in minutes of departure delay, on average, holding all else constant. The coefficient on LATE_AIRCRAFT_DELAY can be interpreted in the same way. The coefficient on WEATHER_DELAY indicates that we would expect to see a 0.93 minute increase in DEPARTURE_DELAY for every one minute increase in weather delay, on average, holding all else constant. The coefficient on AIR_SYSTEM_DELAY can be interpreted as, for every one minute increase in air system delay, we would see an associated 0.68 minute increase in minutes of departure delay (approximately half of a minute). All of our estimated model parameters are significant beyond the 99.99% confidence level, indicating that the relationships observed between them and DEPARTURE_DELAY are likely not due to chance.

Diagnostics

We can see that once we added more features to the right hand side, our linear model was able to explain more of the variability in minutes of departure delay. This improvement is

denoted by the R-squared of 0.9469, a 51.77% increase in explanatory power from our first bivariate linear regression model. The F-statistic is very large, and the P-value is very small, telling us that this model is jointly significant.

Attempting to Regularize Multivariate Linear Regression Model

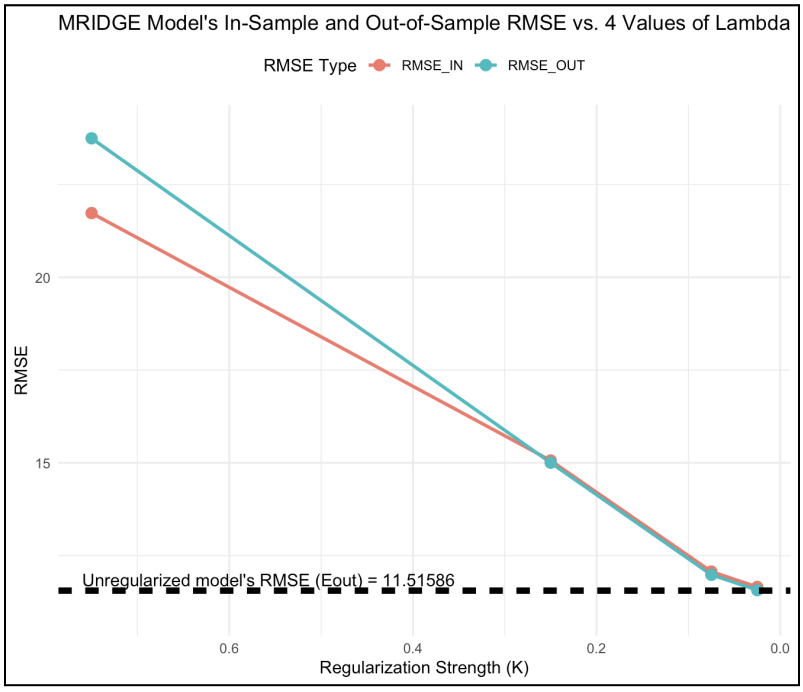
Method

Once again, we attempted to regularize our model by leveraging the lmridge() package and tuning performance by decreasing the hyperparameter, lambda, by 66% across four RIDGE models. We then made predictions on all four models and calculated their in-sample and estimated-out of sample performances.

Regularization Performance

```
> TABLE_VAL_MM_RIDGE
           K=0.75   K=0.25  K=0.075  K=0.025    UNREG
RMSE_IN  23.73083 15.06338 12.07047 11.65685 11.59823
RMSE_OUT 23.74574 15.00398 11.98404 11.57112 11.51586
```

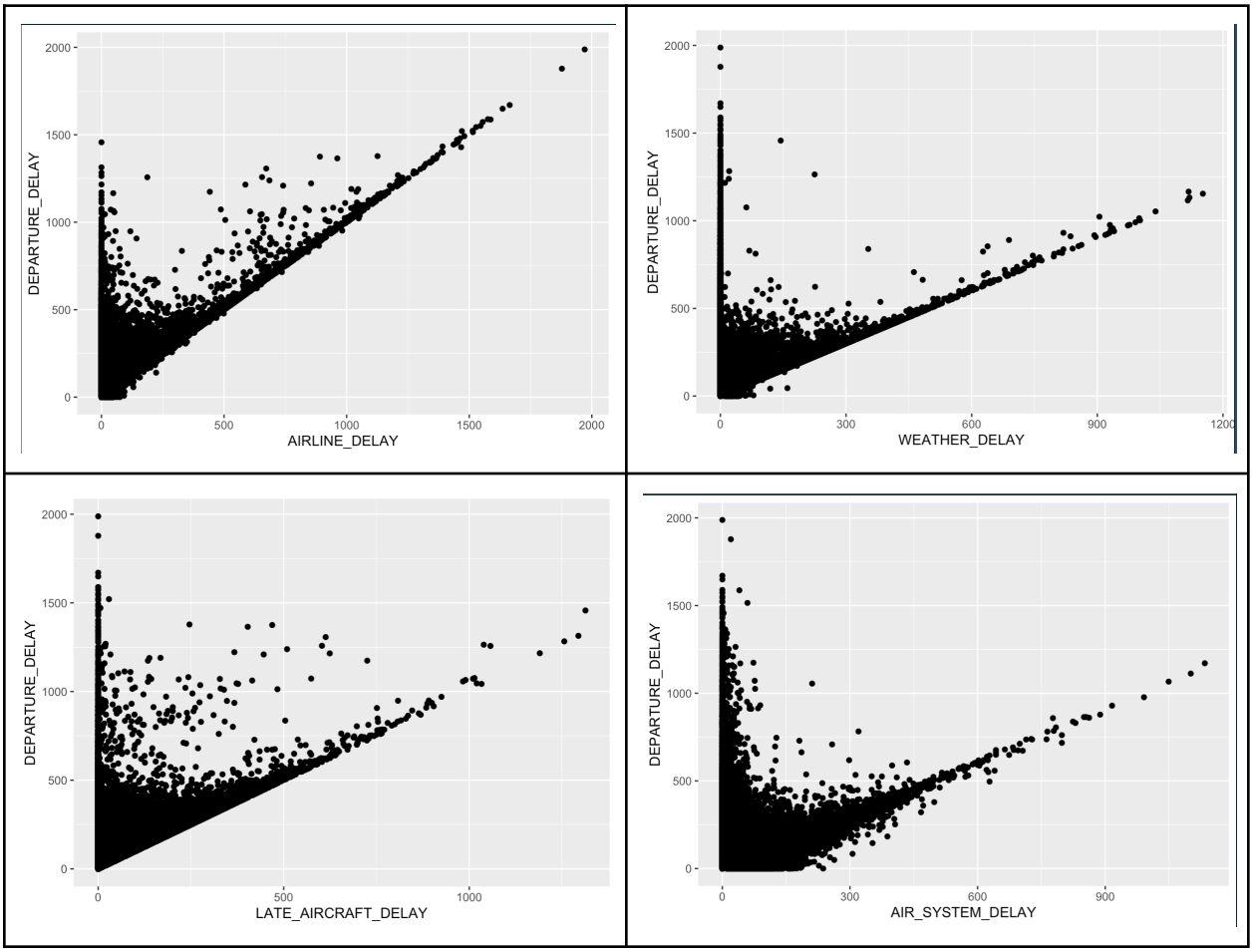
The results of regularization follow the same pattern we saw when we were regularizing the quartic bivariate regression model. As lambda's effect decreases, the in-sample and estimated out-of-sample errors improve. It is important to observe that the estimated out of sample error actually begins to be lower than the in-sample error. There are a few reasons why this might be happening. If the linear model is fitting an outlier in the in-sample data, it will be able to capture more noisy points in the training set. This could also be due to a random fluke from the partitioning, where the training points fall closely to the regression line. It is also important to note that once again the regularized model is not able to outperform the unregularized one.



The graph above is a visual representation of the declining in-sample and out-of-sample errors as regularization strength decreases from left to right. The dashed lines in black show the in and out of sample errors of the unregularized model. This graph shows us how the errors of the RIDGE models are converging to those of the unregularized model, indicating that implementing regularization had no effect on improving the model's performance both in-sample and out-of-sample. We can observe that the in-sample and estimated out-of-sample errors move closer together, indicating that as regularization decreases, generalization increases. This means that RIDGE is having the opposite effect on our model. We can also see that the errors approach the black dashed lines (that are very close together due to the multivariate linear regression model's strong generalization ability).

Multivariate SPLINE Regression Model

Scatterplot of Regressors vs. DEPARTURE_DELAY



We viewed the scatterplots of each independent variable we were considering, to see if we could visualize any non-linear relationships that needed to be captured. It quickly became apparent that a linear regression line alone would not be able to capture any of the variables' relationships with DEPARTURE_DELAY, so we decided to perform SPLINE regression to see if added complexity would capture more of the variability in minutes of departure delay than the multivariate linear regression model.

Model

```
Family: gaussian
Link function: identity

Formula:
DEPARTURE_DELAY ~ s(AIRLINE_DELAY) + s(LATE_AIRCRAFT_DELAY) +
    s(WEATHER_DELAY) + s(AIR_SYSTEM_DELAY)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 28.154874   0.007685    3663   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                        edf Ref.df       F p-value
s(AIRLINE_DELAY)       8.886  8.995 2041349  <2e-16 ***
s(LATE_AIRCRAFT_DELAY) 8.453  8.874 1811181  <2e-16 ***
s(WEATHER_DELAY)       8.937  8.998  313159  <2e-16 ***
s(AIR_SYSTEM_DELAY)    8.999  9.000  382026  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.96   Deviance explained =   96%
GCV = 101.03  Scale est. = 101.03    n = 1710443
```

We performed both a second degree polynomial regression and a SPLINE model, the SPLINE model was able to explain 0.7% more of the variability in minutes of departure delay than the cubic regression model.

Coefficients

The model’s estimated parameter for the intercept is 28.155. This model indicates that when AIRLINE_DELAY is 0, we estimate DEPARTURE_DELAY to be approximately 28 minutes, on average .It can be seen that the intercept value is statistically significant beyond the 99.99% confidence level. The t-statistic is large and the p-value is very small, informing us that the SPLINE model’s estimate for the intercept term is significantly different from 0. The F-statistic on all of the smoothed terms informs us that the smooth linear curve is outperforming a multivariate linear regression model, in-sample.

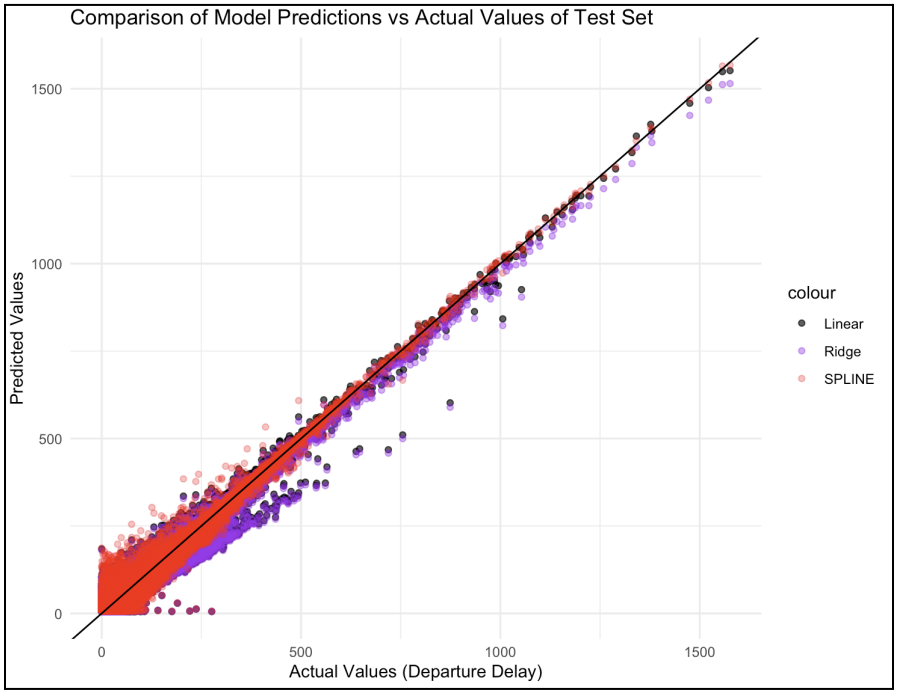
Diagnostics

The model’s deviance explained and adjusted R-squared are much higher than the ones we saw in the bivariate SPLINE regression model. The deviance explained is 96%, a 54% improvement from the bivariate SPLINE regression model, signaling a much stronger in-sample fit. The adjusted R-squared and deviance explained are considered to be very high, putting this model at risk for overfitting.

Benchmarking

	LINEAR	REGULARIZED	SPLINE
RMSE_IN	11.59823	11.65685	10.05125
RMSE_OUT	11.51586	11.57112	10.00250

It came as a surprise that the SPLINE model performed the best out of the sample. We expected it to perform well in sample, as its diagnostic information showed the highest estimate in sample fit. The SPLINE model's predictions are off by 10 minutes on average whereas the other multivariate regression models are predicting incorrectly by more than 11.5 minutes on average. This is one of the first cases where we've seen one model have a significant edge over others. We believe that the in-sample error is slightly worse than the out of sample error due to noise in the training data that the model is reaching to fit.



This plot shows where the predicted data points of each model lie along the black line that represents the actual vs. predicted values. It can be seen that the ridge follows the line the closest. We can also see how the RIDGE regression model with lambda = 0.025 performed very closely to the multivariate linear model.

Support Vector Regression on Principal Components

The problem

When running a support vector regression model, an n x n matrix is created, storing the distances between data points. This meant that with over 1.7 million observations in our training dataset, if we used our entire training data set, the computer would have to store a matrix of

almost 3 trillion distance values in its memory, which was virtually impossible to do with our resources. Aside from an insufficient amount of memory, the amount of time to train this would take days.

The solution: Principal component analysis

Principal component analysis is an unsupervised dimensionality reduction technique. By reducing the dimensionality of our training data, we were able to move the data points into a more manageable space to increase the computational speed with which our SVM model was trained. A problem we wanted to address was that when we used the principal components as our regressors in our support vector regression model, the estimated out-of-sample error was better than the in-sample error. After doing some digging to diagnose the problem we realized two things. The training partition contained outliers that the support vector regression model was aiming to fit, in turn capturing more of the validation data points and driving estimated out-of-sample error down. We also noticed that the cumulative variances captured by each of the principal components in the validation data was higher than those of the training data.

The method

We reshuffled the rows of our raw data set and repartitioned the data in case this phenomenon was happening due to a random fluke in the original partitioning. We then removed any outliers in the DEPARTURE_DELAY columns to ensure that the model would not attempt to fit them. It is important to note that this model will not be benchmarked against the others due to the modifications we had to make to our data partitions. It is also important to note that these re-partitions were not used in any other models.

We then levered the prcomp() function on only the dependent variables, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY, SECURITY_DELAY, to get the principal components.

Summary of principal components from the training data:

```
> summary(flights_train.pc1)
Importance of components:
                          PC1     PC2     PC3     PC4
Standard deviation     11.7991 10.7882 10.0412 3.90671
Proportion of Variance  0.3745  0.3131  0.2713 0.04106
Cumulative Proportion   0.3745  0.6877  0.9589 1.00000
```

Summary of principal components from the validation data:

```
> summary(flights_valid.pc1)
Importance of components:
                          PC1     PC2     PC3     PC4
Standard deviation     11.8351 10.7203 10.0295 3.93553
Proportion of Variance  0.3775  0.3097  0.2711 0.04174
Cumulative Proportion   0.3775  0.6872  0.9583 1.00000
```

These tables show the cumulative proportion of variance for each principal component. The greater the cumulative variance, the more information about our original variables is captured in the lower-dimensional space. In order to increase the support vector machine's ability to generalize well, we were looking for the training data's principal components to have a greater

cumulative proportion of variance captured. We can see that the validation data's PC1 captures more information than the training data's PC1 which is why we used PC1 + PC2 as our right-hand side variables in the support vector regression model.

The support vector regression model

Our support vector regression model formula was DEPARTURE_DELAY ~ PC1 + PC2, using the reduced dimensionality features to predict our target variable. We immediately noticed a difference and were excited to see how much quicker our model was able to converge when we used the PCs as our regressors.

In order for the model to reach convergence in a relatively quick manner, we decided to train and validate the model with subsets of our SVM specific data partitions. We ran the SVM model on the first 20,000 observations of the training dataset, and the first 2,000 observations of the validation dataset (note that rows were re-shuffled prior to repartitioning). We used a linear kernel and the epsilon regression argument.

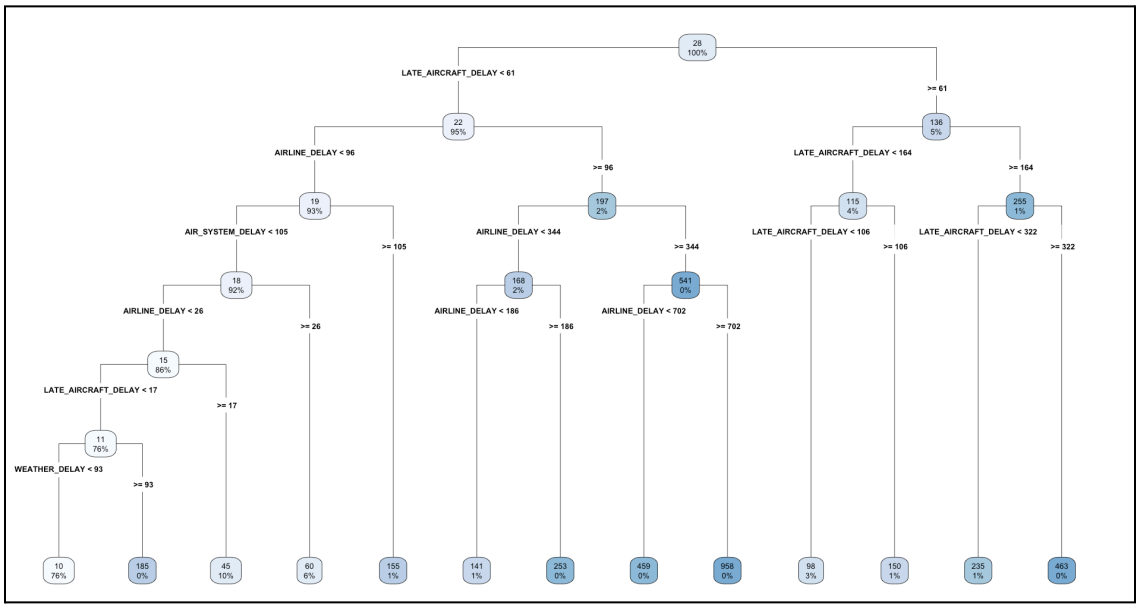
```
Call:
svm(formula = flights_train.y ~ PC1 + PC2, data = ols.data_svm, type = "eps-regression",
    kernel = kern_type, cost = 1, gamma = 1/(ncol(ols.data_svm) - 1),
    coef0 = 0, degree = 2, scale = FALSE)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.25
    epsilon:  0.1


Number of Support Vectors:  19495
```

An observation we made was that the number of support vectors is extremely close to the number of observations used to train the model. This can sometimes indicate overfitting as a majority of the observations were used individually.

RMSE_IN	RMSE_OUT
12.56635	13.43854

After many efforts, tuning, and time spent diagnosing we were able to report an out-of-sample error that is indicative of a multivariate support vector regression model that is able to generalize. We want to reiterate that this model is not benchmarked against our other multivariate models in this report, due to the extraneous steps taken to reach this result.

Regression Tree



Nodes

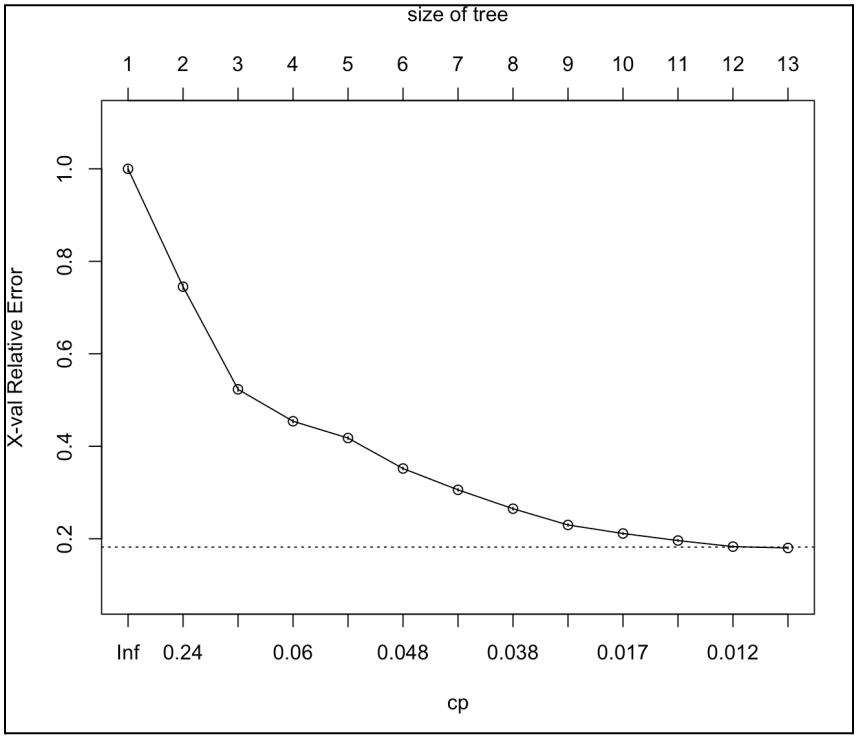
Our regression tree seeks to predict DEPARTURE_DELAY with the same input variables as the other regression models; AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY, and AIR_SYSTEM_DELAY.

It can be seen that the top node is LATE_AIRCRAFT_DELAY, meaning that this variable was found to be the most important to initially split the data. This meant that splitting the variable for the number of minutes by which an aircraft is late led to the best-performing splits. We can also see that the most important factors in predicting minutes of departure delay were the AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, and AIR_SYSTEM_DELAY.

Observations

This tree has many leaves, 13 to be exact. Five of the leaves values for minutes of departure delay will be predicted 0% of the time, meaning that this regression tree model may be overfitting the training data. Another observation is that the right side of the tree seems to be predicting and fitting to outliers, as 1% of the data points are being predicted for a majority of those leaves. This observation is emphasized when we see that at the initial split, 95% of the observations move down to the left side of the tree, whereas only 5% go to the right side of the tree.

Elbow plot



This elbow plot highlights the theme we have seen with regularization on our previous regression models. When the degree of regularization decreases, so do our errors.

Tuning the tree

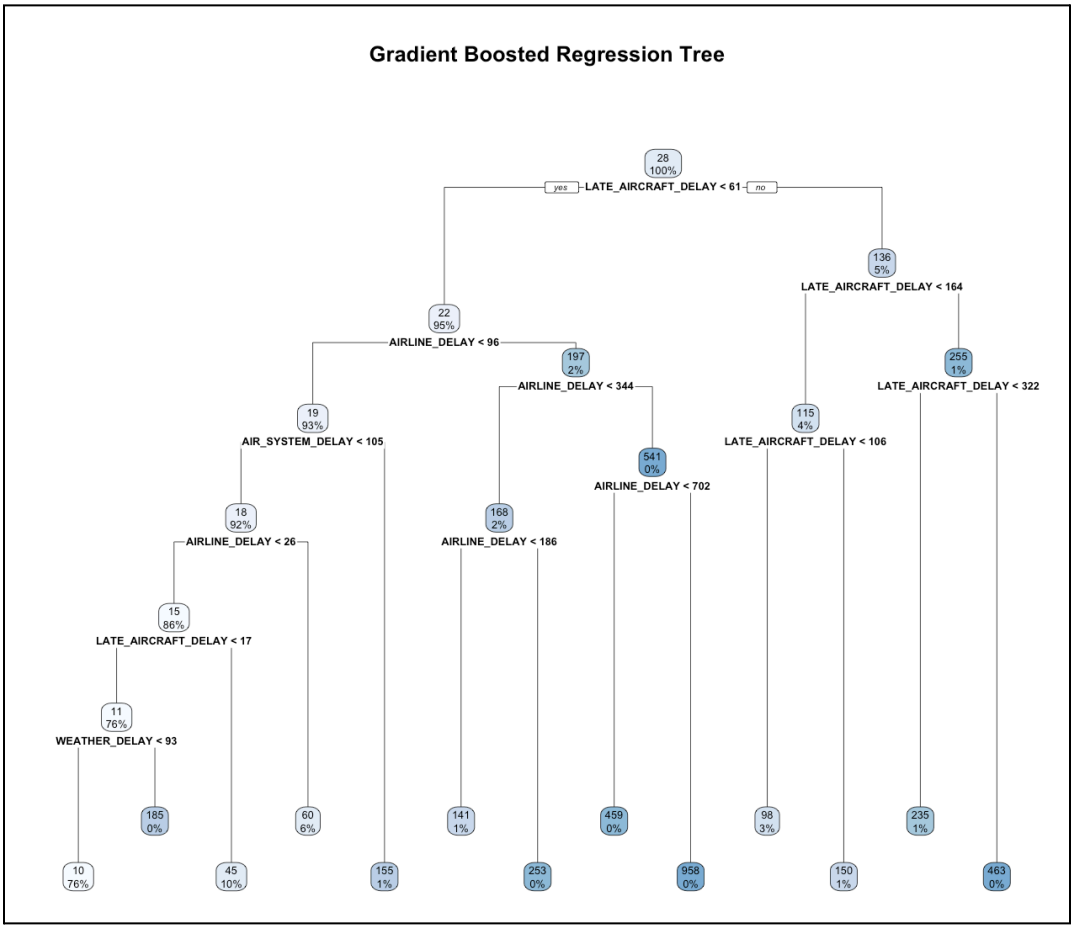
We performed 3-fold cross validation to find the optimal parameters to tune our regression tree with. This was another computation that took more time due to the 1.7 million observations in our training data set.

```
cost_complexity tree_depth min_n .config
          <dbl>      <int> <int> <chr>
   0.0000000001         15     2 Preprocessor1_Model07
```

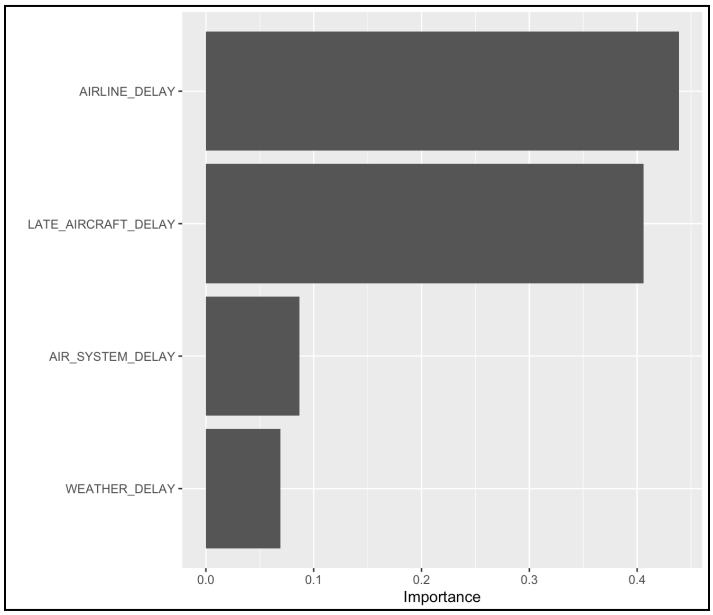
```
             REGTREE     TUNED
RMSE IN   20.894967  8.937537
RMSE OUT 20.916804  9.518980
```

We can see that the best parameters found through cross-validation confirm our analysis of the elbow plot, that regularization is harmful to our model's predictive performance both in and out of sample. After applying the best tuning parameters to the tree we see a huge leap in model performance. The model now incorrectly predicts minutes of departure delay by an average of 8 minutes in-sample. The regression tree is now estimated to predict departure delay by an average of 9.51 minutes out of sample. The model is performing well both in and out of sample, and the model appears to be generalizing well on data it was not trained on. Unfortunately this regression tree model is too complex to plot.

Gradient Boosted Forest



We chose to estimate a gradient-boosted forest model. We decided on a gradient boosting model because its implementation of gradient descent or stochastic gradient descent provides greater optimization than AdaBoost, and we need all of the optimization we can get with the size of our training set. Additionally, gradient boosted forests work well with datasets with many observations as the model is able to find many strong non-linear relationships. Given that our training dataset has over 1.7 million observations, there is a sufficient amount of observations to support the dimensionality of the parameters. The model uses the same input and output variables as the rest of the multivariate models to ensure adequate benchmarking



Feature importance with these gradient descent boosting models is important because they act like a black box, where interpretation can get lost easily. We can see that both AIRLINE_DELAY and LATE_AIRCRAFT_DELAY are responsible for a majority of the error reduction in this model. This tells us that in this model, airline specific delays, and late aircraft arrival times are the primary reasons for increases in departure delay times.

In-Sample and Out-of-Sample Performance of Multivariate Models

	LINEAR	RIDGE	SPLINE	TUNED	GXBOOST
RMSE IN	11.598228	11.656852	10.051247	8.937537	9.030226
RMSE OUT	11.515862	11.571123	10.002503	9.518980	9.205486

After performing validation to compute in and out of sample error on our models, we found that the tuned regression tree had the best in-sample fit, predicting incorrectly by 8.93 minutes on average. The gradient boosted regression tree performed the best out of sample, with a prediction error of 9.2 minutes on average.

Multivariate Model Selection and Reporting Estimated Out-of-Sample-Error

```
# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        9.31
```

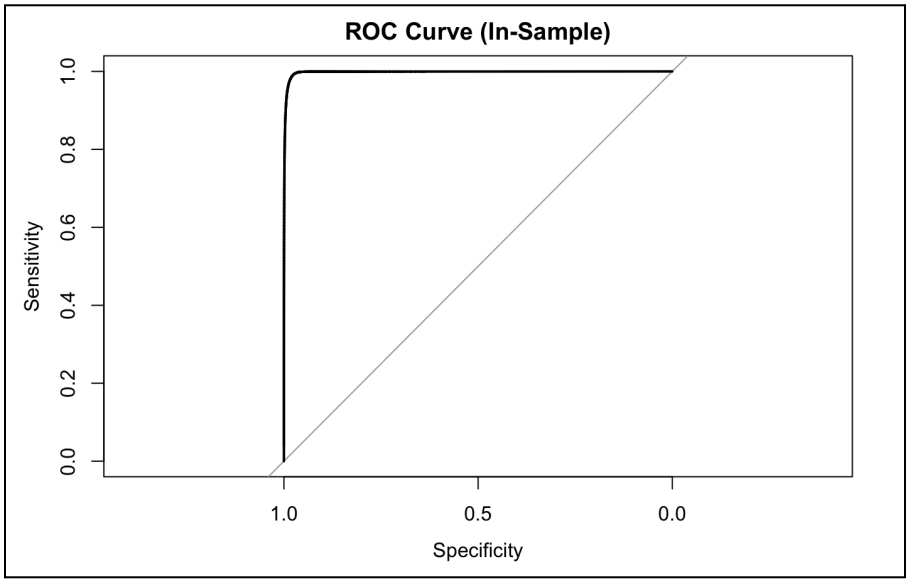
After running the gradient boosted regression tree on our hidden set, we can report an estimated out of sample error of 9.31 minutes.

Classification Models

We started our classification analysis by creating a binary version of the DEPARTURE_DELAY variable, which we called DELAYED. At first, we thought it would make sense to define a delay as anything over two hours, since the Department of Transportation (DOT) mandates refunds for delays of three hours or more. However, when we tested that logic, the results came out extremely skewed — there were barely any flights delayed over three hours, so we didn't have enough variation for the model to learn from.

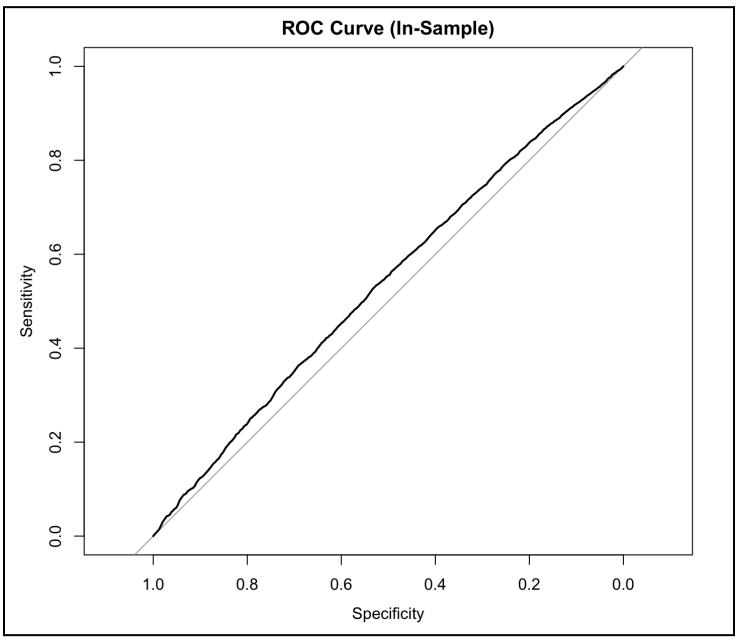
We then pivoted to using a two-hour threshold but found that interpretation was a bit awkward. Labeling something as "not delayed" just because it was under two hours didn't align well with how people typically think of delays. After some trial and error with different models and thresholds, we ended up defining DELAYED as 1 if the flight was delayed by any amount of time (greater than 0), and 0 if it wasn't. This gave us a more balanced and intuitive classification.

From there, we started modeling using GLMs, running both logit and probit regressions. Our first approach focused on variables like WEATHER_DELAY, LATE_AIRCRAFT_DELAY, AIRLINE_DELAY, and AIR_SYSTEM_DELAY. Not surprisingly, these predictors gave us very high accuracy.



Since these delay types essentially form part of the outcome we were trying to predict, the models were basically saying “delays cause delays,” which, while true, wasn’t very helpful.

So, we shifted gears and ran a model that excluded the delay variables. Instead, we tested predictors like DAY_OF_WEEK, DISTANCE, and a breakdown of scheduled departure times (SCHEDULED_GRAVEYARD_DEPARTURE, SCHEDULED_MORNING_DEPARTURE, SCHEDULED_AFTERNOON_DEPARTURE, and SCHEDULED_NIGHT_DEPARTURE). The model wasn’t as accurate, but it offered more useful insights.



Moving forward, we expanded the feature set by hot-encoding the DAY_OF_WEEK variable into seven dummy variables. We also created CAT_DELAY_RANK, a categorical variable that grouped airlines based on their average AIRLINE_DELAY time. We calculated those averages from the full raw dataset and then ranked the airlines from least to most delayed. We also used a stepwise regression to identify which predictors were statistically strongest. Unsurprisingly, it pointed us right back to the delay-type variables. Even though those would’ve given us perfect accuracy, we decided to leave them out for the final version since they didn’t offer any new insights.

In the end, we landed on a model that included: All seven DAY_OF_WEEK dummies, SCHEDULED_GRAVEYARD_DEPARTURE, SCHEDULED_MORNING_DEPARTURE, SCHEDULED_AFTERNOON_DEPARTURE, SCHEDULED_NIGHT_DEPARTURE, and CAT_DELAY_RANK. When we ran initial versions of this model, we noticed that the output variable was highly imbalanced, with about 86.56% of the flights were classified as delayed (1). We downsampled the majority class to create a more balanced training set. This step helped

ensure that the model could learn to distinguish instead of automatically defaulting to the majority class.

Logit Regression

```
Call:
glm(formula = DELAYED ~ MONDAY + TUESDAY + THURSDAY + FRIDAY +
    SCHEDULED_GRAVEYARD_DEPARTURE + SCHEDULED_MORNING_DEPARTURE +
    SCHEDULED_AFTERNOON_DEPARTURE + CAT_DELAY_RANK, family = "binomial",
    data = flights_train_bal)

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                    0.701991   0.015833  44.338  < 2e-16 ***
MONDAY                         0.063777   0.009022   7.069 1.56e-12 ***
TUESDAY                        0.028861   0.009297   3.104  0.00191 **
THURSDAY                       0.059980   0.008933   6.715 1.89e-11 ***
FRIDAY                         0.036432   0.008948   4.072 4.67e-05 ***
SCHEDULED_GRAVEYARD_DEPARTURE -1.101114   0.023035 -47.801  < 2e-16 ***
SCHEDULED_MORNING_DEPARTURE   -0.729247   0.008016 -90.976  < 2e-16 ***
SCHEDULED_AFTERNOON_DEPARTURE -0.219917   0.007790 -28.230  < 2e-16 ***
CAT_DELAY_RANK2               -0.386695   0.015128 -25.561  < 2e-16 ***
CAT_DELAY_RANK3               -0.379942   0.015167 -25.051  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 637554  on 459897  degrees of freedom
Residual deviance: 625921  on 459888  degrees of freedom
AIC: 625941

Number of Fisher Scoring iterations: 4
```

This is a logistic regression model on the balanced training dataset. This allowed us to generate interpretable insights about the scheduling and airline-related factors associated with delays. The results showed strong statistical significance across all variables.

```
                                    2.5 %    97.5 %
(Intercept)                    2.0177659 1.9561583 2.0814107
MONDAY                         1.0658544 1.0471734 1.0848699
TUESDAY                        1.0292818 1.0106960 1.0482096
THURSDAY                       1.0618157 1.0433878 1.0805703
FRIDAY                         1.0371038 1.0190744 1.0554526
SCHEDULED_GRAVEYARD_DEPARTURE 0.3325006 0.3177893 0.3478213
SCHEDULED_MORNING_DEPARTURE   0.4822719 0.4747525 0.4899067
SCHEDULED_AFTERNOON_DEPARTURE 0.8025855 0.7904222 0.8149321
CAT_DELAY_RANK2               0.6792983 0.6594379 0.6997269
CAT_DELAY_RANK3               0.6839012 0.6638564 0.7045213
```

Flights scheduled during the graveyard and early morning hours were associated with significantly lower odds of delay. In contrast, flights later in the day, particularly afternoon

departures, had a higher likelihood of being delayed. This effect might be attributed to a snowball effect throughout the day, delays building one on top of each other to leave afternoon and night time departures with higher likelihood of delay. When looking at the days of the week, Thursday stood out as having the strongest positive relationship with delays, followed by Monday and Friday. This pattern may reflect increased flight volume and congestion toward the end of the workweek. The CAT_DELAY_RANK variable also provided meaningful insight. Compared to the best-performing airlines (rank 1), flights operated by airlines in rank 2 or rank 3 had significantly higher odds of delay, confirming that airline quality plays a measurable role in departure reliability. Because this model was trained on balanced data, it wasn't biased toward predicting delays simply because they were more common in the original dataset. This gave us a more accurate picture of how each factor contributes to the probability of a flight being delayed.

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0  21461  92666
         1  27789 224608

               Accuracy : 0.6714
                 95% CI : (0.6698, 0.6729)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0923

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7079
            Specificity : 0.4358
         Pos Pred Value : 0.8899
         Neg Pred Value : 0.1880
             Prevalence : 0.8656
         Detection Rate : 0.6128
   Detection Prevalence : 0.6886
      Balanced Accuracy : 0.5718

       'Positive' Class : 1
```

The overall accuracy of the model was 67.14% which is substantially lower than the No Information Rate of 86.66%, however since our data was highly skewed it is important to judge the model based on not only accuracy but also other important factors. Sensitivity (true positive rate) was 70.79%, indicating the model was fairly effective at identifying delayed flights. Specificity (true negative rate) was lower at 43.58%, meaning it struggled more to correctly identify non-delayed flights. Balanced accuracy, which averages sensitivity and specificity, came in at 57.18% — suggesting moderate ability to distinguish between classes in a highly imbalanced setting.

Probit Regression

```
Call:
glm(formula = DELAYED ~ MONDAY + TUESDAY + THURSDAY + FRIDAY +
    SCHEDULED_GRAVEYARD_DEPARTURE + SCHEDULED_MORNING_DEPARTURE +
    SCHEDULED_AFTERNOON_DEPARTURE + CAT_DELAY_RANK, family = binomial("probit"),
    data = flights_train_bal)

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                    0.437491   0.009801  44.638  < 2e-16 ***
MONDAY                         0.039725   0.005623   7.065 1.61e-12 ***
TUESDAY                        0.017940   0.005795   3.096  0.00196 **
THURSDAY                       0.037268   0.005568   6.694 2.18e-11 ***
FRIDAY                         0.022575   0.005577   4.048 5.17e-05 ***
SCHEDULED_GRAVEYARD_DEPARTURE -0.685289   0.014055 -48.758  < 2e-16 ***
SCHEDULED_MORNING_DEPARTURE   -0.455729   0.004986 -91.403  < 2e-16 ***
SCHEDULED_AFTERNOON_DEPARTURE -0.137258   0.004857 -28.262  < 2e-16 ***
CAT_DELAY_RANK2               -0.240458   0.009372 -25.658  < 2e-16 ***
CAT_DELAY_RANK3               -0.235979   0.009396 -25.115  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 637554  on 459897  degrees of freedom
Residual deviance: 625921  on 459888  degrees of freedom
AIC: 625941

Number of Fisher Scoring iterations: 4
```

To further validate we ran a probit regression with the exact same predictors from the logit model. We can see that all of the predictors continue to all hold statistical significance. We see that the probit model produced nearly identical results which is to be expected as we reinforce all the relationships which we observed previously. For example, flights departing in the graveyard and morning time slots still showed a strong negative association with delays, while Thursday and lower-ranked airlines continued to be positively associated with a higher likelihood of delay, This consistency across models strengthened the reliability of our final variable selection and confirmed that the relationships we uncovered are not dependent on a specific functional form.

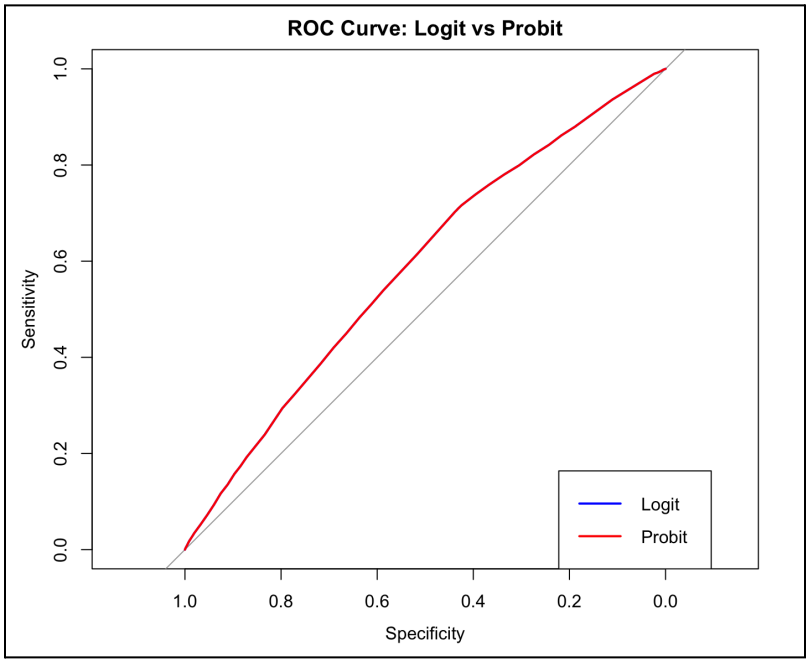
```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0  21555  93348
         1  27695 223926

               Accuracy : 0.6698
                 95% CI : (0.6682, 0.6713)
    No Information Rate : 0.8656
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0918

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7058
            Specificity : 0.4377
         Pos Pred Value : 0.8899
         Neg Pred Value : 0.1876
             Prevalence : 0.8656
         Detection Rate : 0.6109
   Detection Prevalence : 0.6865
      Balanced Accuracy : 0.5717

       'Positive' Class : 1
```

The confusion matrix for the probit model showed results that were almost identical to the logit model. Accuracy landed at 66.98%, which is lower than the No Information Rate of 86.56%. That's expected given how skewed the original data was before we balanced it.

The model did a decent job identifying delayed flights, with a sensitivity of 70.58%, but had a harder time correctly predicting on-time flights — specificity came out to 43.77%. Similar to the logit model, when it predicted a delay, it was right most of the time (almost 89%), but when it predicted an on-time flight, it was usually wrong (NPV of just 18.76%).

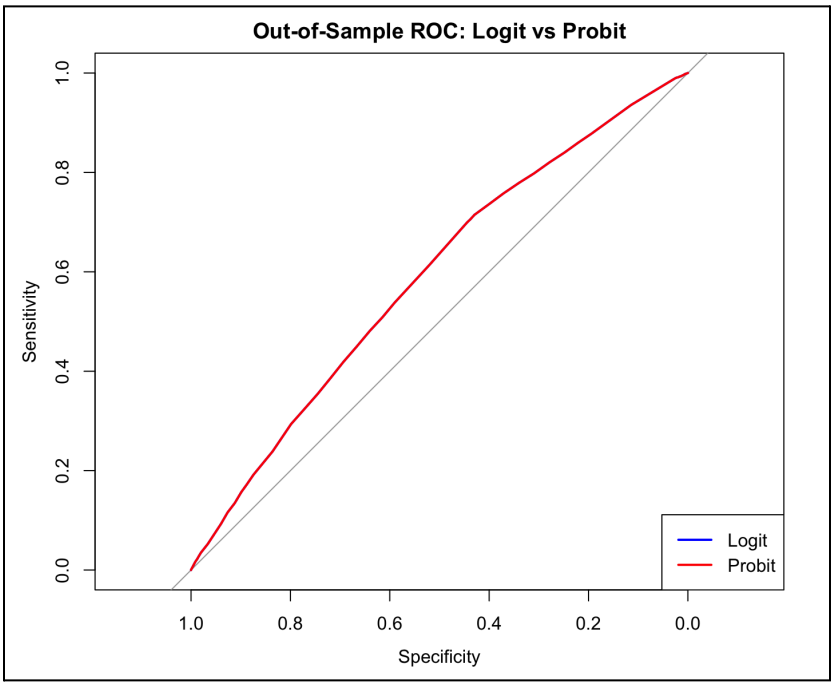
The balanced accuracy, which averages sensitivity and specificity, came out to 57.17%, basically the same as in the logit model. Overall, the confusion matrix backed up what we saw earlier: the probit and logit models gave us the same story. Both were more effective at catching delays than predicting on-time flights, and both were consistent in how they handled the data.

In-Sample ROC Curve



We compared the ROC curves for both the in-sample logit and probit models. The curves show that both models performed slightly better than random guessing, which is expected given the complexity of flight delay prediction and the minimal use of historical delay-type variables in the final models. The AUC (Area Under the Curve) for the logit model came out to 0.59, while the probit model followed closely behind at 0.58. This small gap reflects what we saw earlier — both models behaved almost identically, with the logit model only slightly outperforming in terms of classification performance. While these AUC scores aren't particularly high, they do confirm that the models learned something meaningful from the input variables. That said, the low separation from the baseline also highlights the real-world challenge of predicting delays using only schedule- and airline-based features. There's statistical value in the models, but their practical predictability is still somewhat limited.

Out-Sample ROC Curve



To evaluate how well the models generalize beyond the training data, we also tested them on a holdout validation set. As expected, both the logit and probit models showed very similar performance out of sample, with the logit model performing slightly better in terms of classification accuracy and ROC curve shape. The ROC curves confirm that while both models are picking up on meaningful patterns, their out-of-sample predictive power is modest. This isn't surprising given the limited feature set , we deliberately avoided using variables like actual delay causes to focus more on schedule- and airline-based predictors.

These results suggest that while the models are statistically sound, there's definitely room for improvement. Adding more granular or real-time features, such as weather conditions, airport traffic, or historical delay trends, could improve generalizability and boost out-of-sample performance in future iterations.

Binary Support Vector Machine (SVM) Classification Models

```
Call:
svm(formula = DEPARTURE_DELAY ~ SCHEDULED_MORNING_DEPARTURE + SCHEDULED_AFTERNOON_DEPARTURE +
    SCHEDULED_NIGHT_DEPARTURE + MONDAY + TUESDAY + WEDNESDAY + THURSDAY + FRIDAY + SATURDAY +
    SUNDAY + LATE_AIRCRAFT_DELAY + DELAY_RANK, data = flights_train_small, type = "C-classification",
    kernel = "radial", cost = 1, gamma = 1/(ncol(flights_train_small) - 1), scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  5471
```

To explore the predictive power of Support Vector Machines in our classification task, we implemented a binary classification model to predict whether a flight would be delayed (1) or on time (0). Given the computational limitations of personal and school-provided computers, we constrained the model to a smaller training set of 20,000 observations and a validation set of 2,000, allowing the algorithm to converge efficiently without overloading memory. Before modeling, we performed feature engineering to improve performance and interpretability. First, the time of day variable was split into four distinct one-hot encoded categories (graveyard, morning, afternoon, night), and the day of the week was similarly one-hot encoded. As previously mentioned, we also created a new variable called Airline Delay Rank, which categorized carriers into ranked tiers based on their overall historical airline delay performance. Our outcome variable, departure delay, was converted into a binary indicator to reflect a simplified yes/no delay status. In addition, we ran two versions of the model: one using the original imbalanced training dataset and another using balanced training data, in which on-time and delayed flights were equally represented. This allowed us to evaluate the effect of class imbalance on model performance and generalization.

The first version of our SVM model used the original training partition without any balancing adjustments, meaning the data still reflected the real-world skew where delayed flights significantly outnumbered on-time ones. Despite the class imbalance, the model returned consistent in-sample and out-of-sample error rates of 13.44%, suggesting it could generalize relatively well to unseen data. This performance made it one of our stronger classification models in terms of raw predictive accuracy; however, the simplicity of the binary setup presented trade-offs. While the model could reliably distinguish between delayed and non-delayed flights, it offered no insight into delay severity, a limitation we later attempted to address using multiclass models. Overall, this early model helped us confirm that our chosen features, time of day, day of the week, airline delay rank, and late aircraft delays, were effective inputs, especially when memory and runtime were restricted. Overall, this version of the SVM served as a valuable benchmark; perhaps not the most complex, but cleanly executed and relatively robust under tight computational constraints.

```
Call:
svm(formula = DEPARTURE_DELAY ~ SCHEDULED_MORNING_DEPARTURE + SCHEDULED_AFTERNOON_DEPARTURE +
    SCHEDULED_NIGHT_DEPARTURE + MONDAY + TUESDAY + WEDNESDAY + THURSDAY + FRIDAY + SATURDAY +
    SUNDAY + LATE_AIRCRAFT_DELAY + DELAY_RANK, data = flights_train_bal_small, type = "C-classification",
    kernel = "radial", cost = 1, gamma = 1/(ncol(flights_train_bal_small) - 1), scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  15619
```

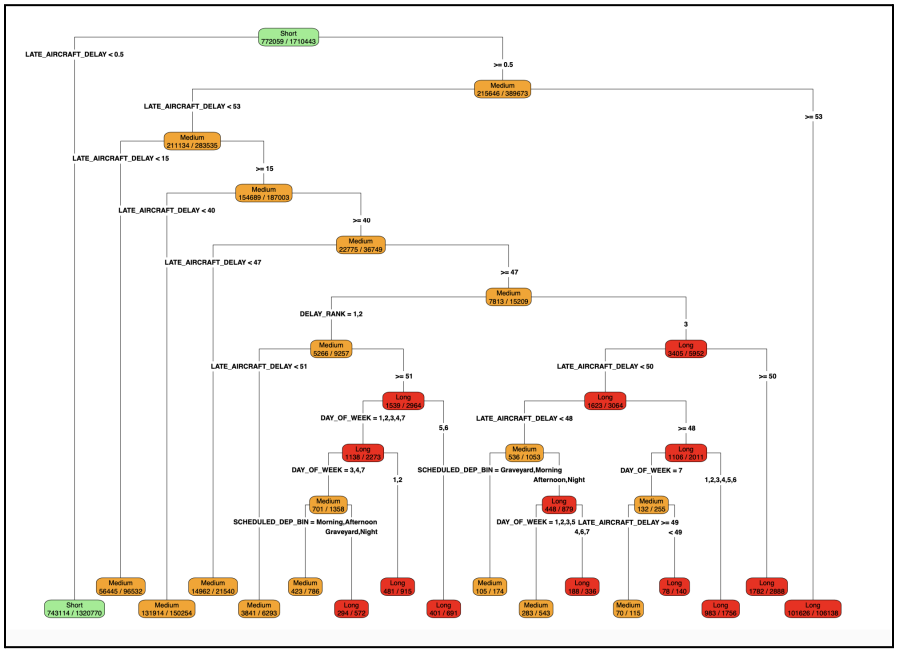
To test whether class imbalance affected our model's performance, we ran a second version of the SVM using a balanced training dataset, where the number of delayed and on-time flights was equal. All predictors remained the same as in the original model; the only difference was the composition of the training data. Surprisingly, performance declined significantly as the in-sample error jumped to 37.91%, and the out-of-sample error increased to 63.02%, making it one of our weakest models. This outcome contradicted our initial assumption that balancing the classes would improve predictive power. One possible explanation is that by forcing an artificial 50/50 distribution in the training data, the model was trained on a class structure that didn't reflect the real-world proportions seen in the validation set. As a result, it may have struggled to generalize back to a naturally imbalanced context, essentially learning a version of the problem that was less representative of reality. We also noticed a sharp increase in the number of support vectors, rising from 5,471 in the unbalanced model to 15,619 in the balanced version. In SVMs, support vectors represent the data points most influential in defining the decision boundary. A large number of support vectors often indicates a more complex decision surface, and in this case, it may reflect that the model had to work harder to separate the artificially balanced classes. This likely introduced overfitting in the training phase and contributed to the poor generalization performance on the validation set. Ultimately, this model reinforced a key lesson in predictive modeling: a technically "fairer" dataset isn't always better, especially when it distorts the underlying class distribution of the real-world problem.

Multiclass Classification Tree (CART) Models: Unpruned

Imbalanced Data Model

Building on the same foundation used for our previous models, we constructed a series of CART models to approach the delay prediction task using a multiclass output. All prior feature engineering remained unchanged, including one-hot encoding for day of week and scheduled departure time (morning, afternoon, night, graveyard), along with late aircraft delay and airline delay rank. The key difference in this modeling approach was the structure of the dependent variable. Instead of predicting a binary outcome, the CART model classified flights into four categories based on total minutes delayed: flights with zero delay were labeled on time, those

delayed by more than zero but no more than 15 minutes were considered short delays, delays greater than 15 minutes but less than or equal to 60 minutes were labeled medium delays, and anything over 60 minutes was categorized as a long delay. For the initial version of our decision tree, we focused less on optimization and more on visual interpretability. We allowed deeper splits to ensure that each predictor was visible in the tree structure, giving us an early sense of how different variables interacted and contributed to delay severity. This model served as a starting point before introducing tuning and class balancing in later iterations.



Before pruning, this initial CART model produced a highly detailed tree structure dominated by splits on late aircraft delay, which emerged as the most influential predictor. This behavior reflects CART's greedy nature as it selects the variable that yields the greatest immediate improvement in classification purity at each split (often favors continuous variables). As a result, categorical features such as day of week and scheduled departure appear deeper in the tree after the model has exhausted more impactful numerical splits. Unfortunately, the model failed to predict any on-time flights, which can likely be attributed to the imbalance in our training data, where delayed flights were far more prevalent.

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime      0      0      0      0
   Short   49250 159423  57549  16749
   Medium      0   6292  44698   8205
   Long        0      0   1598  22760

Overall Statistics

               Accuracy : 0.619
                 95% CI : (0.6174, 0.6206)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3613

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                 0.0000       0.9620        0.4304     0.47701
Specificity                 1.0000       0.3847        0.9448     0.99499
Pos Pred Value                 NaN       0.5634        0.7551     0.93440
Neg Pred Value              0.8656       0.9247        0.8075     0.92707
Prevalence                  0.1344       0.4521        0.2833     0.13018
Detection Rate              0.0000       0.4350        0.1220     0.06210
Detection Prevalence        0.0000       0.7720        0.1615     0.06646
Balanced Accuracy           0.5000       0.6734        0.6876     0.73600
```

The validation confusion matrix for this pre-tuned, imbalanced CART model shows an overall accuracy of 61.9%, which is noticeably better than the No Information Rate of 45.2%. That means the model performed better than a naive guess that always predicts “Short” delays, which were the most common class. Still, the model had some clear weaknesses. Most importantly, it failed to predict any on-time flights. The sensitivity for that class was 0%, which isn’t surprising given our imbalanced training data; there simply weren’t enough on-time examples for the model to learn from. “Short” delays were predicted well regarding sensitivity, but precision wasn’t great. The model also picked up on “Medium” and “Long” delays to some degree, though it struggled to predict them cleanly. Again, a big reason for this is that CART models tend to favor continuous variables like Late Aircraft Delay, which dominated the splits. Categorical features like day of week or departure time had less influence, likely because they didn’t split the data as effectively or cleanly. Overall, this model gives us a solid first look, but it leans heavily toward the majority class and doesn’t capture minority ones very well.

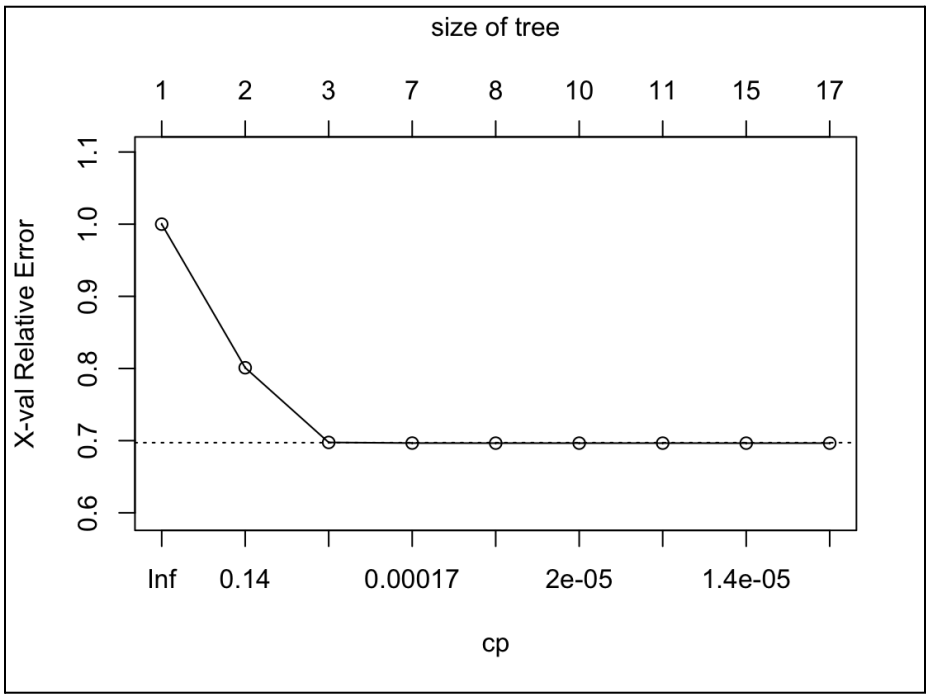
Balanced Data Model

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime  21870  55403  18770   7054
   Short   27154 102987  38152   9420
   Medium    224   7209  43014   7071
   Long        2    116   3909  24169

Overall Statistics

               Accuracy : 0.5239
                 95% CI : (0.5223, 0.5256)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3086

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                0.44406       0.6215        0.4142     0.50654
Specificity                0.74398       0.6279        0.9448     0.98737
Pos Pred Value             0.21213       0.5795        0.7478     0.85718
Neg Pred Value             0.89606       0.6678        0.8031     0.93041
Prevalence                 0.13437       0.4521        0.2833     0.13018
Detection Rate             0.05967       0.2810        0.1174     0.06594
Detection Prevalence       0.28128       0.4849        0.1569     0.07693
Balanced Accuracy          0.59402       0.6247        0.6795     0.74695
```

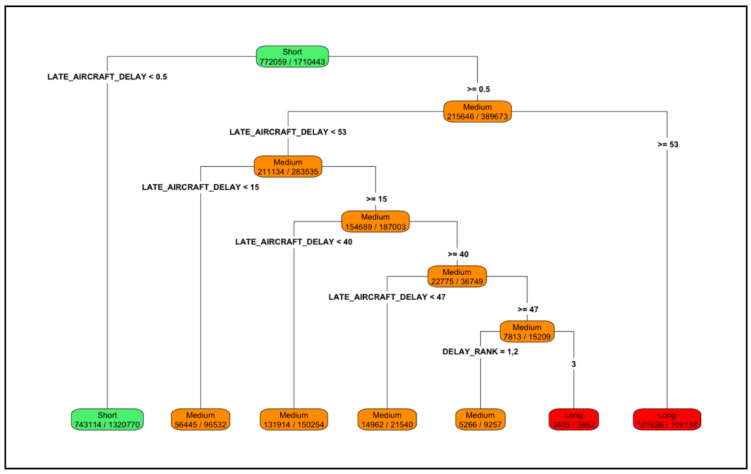
We used a balanced version of the training dataset for our second CART model to address the class imbalance we saw in the first iteration. Each delay category was downsampled to match the size of the smallest group, giving the model an equal number of observations for On Time, Short, Medium, and Long delays. After training on this balanced dataset, the model achieved a validation accuracy of 52.4%, which outperformed the No Information Rate of 45.2%. The in and out-of-sample error rates were fairly close (0.505 and 0.476, respectively), indicating that the model generalizes reasonably well and is not overfitting. The Kappa score of 0.31 suggests the model is doing noticeably better than random guessing, but there's still room for improvement. In terms of class performance, the model did best at identifying Short and Long delays, with sensitivities of 0.62 and 0.51. It had more trouble picking up On Time and Medium delays, both of which landed around the mid-0.40s in sensitivity. While the overall performance didn't drastically improve from the imbalanced version, balancing the dataset helped the model recognize underrepresented classes more consistently, especially in predicting OnTime flights.

Multiclass Classification Tree (CART) Models: Pruned

Imbalanced Data Model



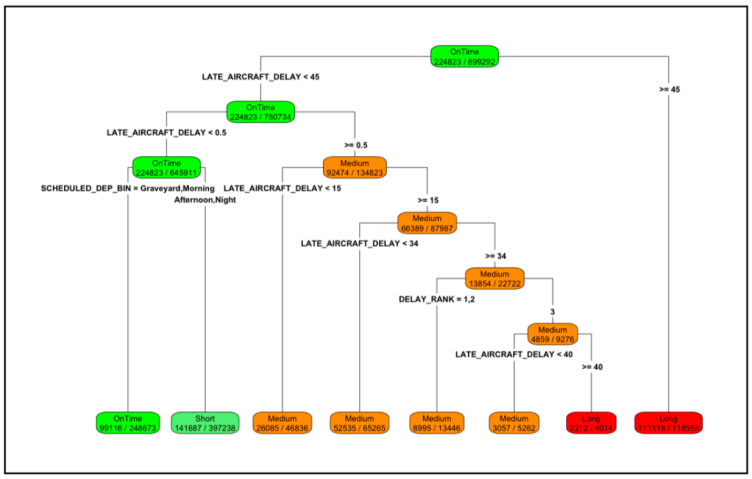
For our third CART model, we introduced pruning to improve the model's generalizability and to avoid overfitting the imbalanced training data. Using the built-in complexity parameter (CP) plot, we applied the 1-standard error rule, which helps select a simpler tree that performs nearly as well as the most complex one. We looked at the cross-validated error values (xerror) from the CP table, found the minimum value, and then selected the largest CP value where the error stayed within one standard error of that minimum. That CP value was then used to prune the tree. We chose this method because it prioritizes simplicity and generalization over memorization. While an unpruned tree typically yields higher training accuracy by fitting every nuance in the data, that often comes at the cost of poor performance on unseen data. Pruning helps strike a better balance by trimming back less valuable branches and focusing on the most essential splits for broader patterns.




```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime      0      0      0      0
   Short   49250 159423  57549  16749
   Medium      0   6292  44825   8374
   Long        0      0   1471  22591

Overall Statistics

               Accuracy : 0.6189
                 95% CI : (0.6173, 0.6205)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.361

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                 0.0000       0.9620        0.4317     0.47347
Specificity                 1.0000       0.3847        0.9442     0.99539
Pos Pred Value                 NaN       0.5634        0.7535     0.93887
Neg Pred Value              0.8656       0.9247        0.8078     0.92664
Prevalence                  0.1344       0.4521        0.2833     0.13018
Detection Rate              0.0000       0.4350        0.1223     0.06164
Detection Prevalence        0.0000       0.7720        0.1623     0.06565
Balanced Accuracy           0.5000       0.6734        0.6879     0.73443
```

After pruning the CART model trained on the imbalanced dataset, performance remained similar to the unpruned version. The overall validation accuracy stayed at 61.9%, which is still well above the No Information Rate of 45.2%. With that being said, pruning helped simplify the tree quite a bit while still holding onto its predictive ability (suggesting that the extra splits in the unpruned tree didn't add much value). As with the earlier imbalanced model, the pruned version still failed to predict any On Time flights, with a sensitivity of 0%, but that's expected given the skew in the training data. The model performed best on Short delays (sensitivity: 96.2%) and picked up Medium and Long delays to a moderate degree. While the accuracy and class-level sensitivities barely changed, pruning did reduce model complexity, making it easier to interpret and likely more generalizable in practice. Compared to the balanced CART model, this version had slightly better overall accuracy (61.9% vs. 52.4%) but worse class balance. All in all, this highlights a tradeoff between overall accuracy and class balance; the pruned model performed better on dominant classes, while the balanced model provided more equitable predictions across categories despite slightly lower accuracy.

Balanced Data Model




```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime  21906  55514  18841   7097
   Short   27344 103909  38708   9652
   Medium      0   6292  42033   6626
   Long        0      0   4263  24339

Overall Statistics

               Accuracy : 0.5244
                 95% CI : (0.5227, 0.526)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3086

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                0.44479       0.6270        0.4048      0.51010
Specificity                0.74328       0.6230        0.9508      0.98663
Pos Pred Value             0.21194       0.5785        0.7649      0.85095
Neg Pred Value             0.89610       0.6693        0.8016      0.93083
Prevalence                 0.13437       0.4521        0.2833      0.13018
Detection Rate             0.05977       0.2835        0.1147      0.06640
Detection Prevalence       0.28200       0.4900        0.1499      0.07804
Balanced Accuracy          0.59403       0.6250        0.6778      0.74837
```

Our final CART model used the balanced training data with pruning applied using the 1-standard error rule, just like in the previous setup. The goal was to keep the tree simpler while still benefiting from the more even class distribution. This version reached a validation accuracy of 52.4%, almost identical to the unpruned version trained on the same balanced data. The training and validation error rates were also fairly close (50.5% and 47.6%), which again suggests the model generalizes well and isn't overfitting. Compared to the imbalanced pruned model, this version showed slightly lower accuracy (52.4% vs. 61.9%), but it offered much better balance across classes, especially when it came to identifying On Time flights, which the imbalanced models consistently failed to detect. The sensitivity for On Time reached 0.44, and every class had some reasonable level of prediction success. While pruning didn't lead to major performance gains on the balanced data, it did help simplify the model structure, making it easier to interpret and potentially more reliable when applied to new data. Overall, this model struck one of the best trade-offs across all four: it didn't win on accuracy, but it offered the most balanced and fair predictions, with solid generalization and a simpler tree that's easier to understand and deploy. With the CART models complete, we now turn to XGBoost, a more advanced ensemble method that may be able to improve both performance and class sensitivity by combining multiple shallow trees into a stronger learner.

Multiclass Random Forest Ensemble Models

Imbalanced Data Model

```
parsnip model object

Ranger result

Call:
 ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~3,      x), num.tre
es = ~100, min.node.size = min_rows(~20, x), importance = ~"impurity",      num.thre
ads = 1, verbose = FALSE, seed = sample.int(10^5,          1), probability = TRUE)

Type:                             Probability estimation
Number of trees:                  100
Sample size:                      1710443
Number of independent variables:  4
Mtry:                             3
Target node size:                 20
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error (Brier s.):  0.3482125
```

To explore a more robust classification approach, we trained a Random Forest model using the imbalanced training dataset. Unlike CART, which builds a single tree, Random Forest builds an ensemble of multiple decision trees (in this case, 100), each trained on different random subsets of the data and predictors. The model randomly selected 3 of the 4 total predictors for each split and required at least 20 observations in each node to allow a split. The four predictors used were Late Aircraft Delay, Airline Delay Rank, Day of Week, and Scheduled Departure, which were consistent with the other models we've previously run. We used the “ranger” engine in R to run the model, which automatically uses Gini to make splits and measures variable importance based on how much each predictor helped improve the model. This setup is designed to reduce overfitting and improve prediction accuracy by averaging across many shallow trees rather than relying on a single one.

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime      0      0      0      0
   Short   49250 159494  57657  16764
   Medium      0   6220  44495   8175
   Long        0      1   1693  22775

Overall Statistics

               Accuracy : 0.6187
                 95% CI : (0.6171, 0.6203)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3607

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                 0.0000       0.9625        0.4285     0.47732
Specificity                 1.0000       0.3841        0.9452     0.99469
Pos Pred Value                 NaN       0.5633        0.7556     0.93077
Neg Pred Value              0.8656       0.9254        0.8071     0.92709
Prevalence                  0.1344       0.4521        0.2833     0.13018
Detection Rate              0.0000       0.4352        0.1214     0.06214
Detection Prevalence        0.0000       0.7726        0.1607     0.06676
Balanced Accuracy           0.5000       0.6733        0.6868     0.73600
```

The Random Forest model trained on the imbalanced dataset reached a validation accuracy of 61.9%, nearly identical to the performance of the CART models on the same data. With error rates of around 38% for both in and out-of-sample, the model generalized well. However, just like the previous imbalanced models, it failed to predict any OnTime flights while strongly favoring the Short delay category. What sets this model apart is its ability to reduce noise by averaging over many decision trees. This likely contributed to the slightly higher Kappa score of 0.36, meaning this model had the most consistent performance beyond just random guessing. Still, the class imbalance remained a challenge, with predictions skewed toward the dominant delay categories. To address this, we test the same Random Forest setup using the balanced training dataset in the next section.

Balanced Data Model

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime  21870  55403  18770   7054
   Short   27207 103214  38319   9475
   Medium    168   6794  42252   6851
   Long        5    304   4504  24334

Overall Statistics

               Accuracy : 0.5229
                 95% CI : (0.5213, 0.5246)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3073

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                0.44406       0.6228        0.4069     0.51000
Specificity                0.74398       0.6265        0.9474     0.98490
Pos Pred Value             0.21213       0.5792        0.7536     0.83487
Neg Pred Value             0.89606       0.6681        0.8016     0.93070
Prevalence                 0.13437       0.4521        0.2833     0.13018
Detection Rate             0.05967       0.2816        0.1153     0.06639
Detection Prevalence       0.28128       0.4862        0.1530     0.07952
Balanced Accuracy          0.59402       0.6247        0.6771     0.74745
```

The Random Forest model trained on the balanced dataset reached a validation accuracy of 52.3%, with training and validation error rates remaining close (50.4% and 47.7%, respectively). While this accuracy is lower than the imbalanced version, the model offered far better balance across classes, particularly in its ability to now identify On Time flights (44.4% sensitivity), which the imbalanced model missed entirely. In terms of model tuning, we tested several variations, including adjustments to the number of trees, mtry, and node size, before selecting this setup based on overall validation performance and consistency. With additional time or access to more advanced computational resources, further improvements could have come from testing alternative balancing techniques (like SMOTE or class weighting) or introducing new predictors that capture airport or weather-specific contexts. These steps might offer added nuance that even ensemble models like Random Forest can benefit from.

Multiclass XGBoost Ensemble Models

Imbalanced Data Model

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime      0      0      0      0
   Short   49250 159423  57549  16749
   Medium      0   6292  44777   8307
   Long        0      0   1519  22658

Overall Statistics

               Accuracy : 0.6189
                 95% CI : (0.6174, 0.6205)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3611

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                 0.0000       0.9620        0.4312     0.47487
Specificity                 1.0000       0.3847        0.9444     0.99524
Pos Pred Value                 NaN       0.5634        0.7541     0.93717
Neg Pred Value              0.8656       0.9247        0.8077     0.92681
Prevalence                  0.1344       0.4521        0.2833     0.13018
Detection Rate              0.0000       0.4350        0.1222     0.06182
Detection Prevalence        0.0000       0.7720        0.1620     0.06596
Balanced Accuracy           0.5000       0.6734        0.6878     0.73505
```

Our first XGBoost model, trained on the imbalanced dataset, performed similarly to previous ensemble models, reaching a validation accuracy of 61.9% and an out-of-sample error of 38.1%. As with the earlier imbalanced models, it failed to detect any OnTime flights, and the model leaned heavily toward predicting Short delays. Class-level sensitivities and balanced accuracy scores for Medium and Long delays were still relatively strong, and the Kappa score of 0.36 matched our best result so far. We tested a few different XGBoost parameter combinations (including tuning depth, learning rate, and sampling ratios), and this model provided the strongest balance between performance and training stability. However, there's still plenty of room for further optimization. Future iterations could benefit from a more systematic hyperparameter search, like grid search or Bayesian optimization, to fine-tune learning rate, tree depth, and regularization terms. Since XGBoost can be sensitive to imbalance without proper class weights or sampling adjustments, we test the model again using a balanced dataset to see if performance across categories improves.

Balanced Data Model

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime  21870  55403  18770   7054
   Short   27143 102965  38133   9403
   Medium    235   7328  43353   7201
   Long        2     19   3589  24056

Overall Statistics

               Accuracy : 0.5245
                 95% CI : (0.5229, 0.5261)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3092

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity               0.44406       0.6213        0.4175     0.50417
Specificity               0.74398       0.6281        0.9438     0.98868
Pos Pred Value            0.21213       0.5796        0.7460     0.86951
Neg Pred Value            0.89606       0.6678        0.8039     0.93018
Prevalence                0.13437       0.4521        0.2833     0.13018
Detection Rate            0.05967       0.2809        0.1183     0.06563
Detection Prevalence      0.28128       0.4847        0.1586     0.07548
Balanced Accuracy         0.59402       0.6247        0.6806     0.74642
```

Using the balanced training data, our XGBoost model reached a validation accuracy of 52.5%, with in-sample performance close behind at 50.5%. This consistency suggests a decent generalization to unseen data, and more importantly, the model delivered noticeably more even performance across all delay categories. For example, the model's sensitivity for On Time flights rose to 44.4%, addressing a key weakness from the imbalanced version. We did test a few different parameter combinations during tuning before landing on this version based on overall performance. While tuning did lead to incremental improvements, we suspect additional gains could come from more systematic optimization. For example, a grid or random search across a broader parameter range could help uncover a stronger combination. It might also help to integrate early stopping or ensemble blending techniques to avoid diminishing returns as boosting rounds increase. Lastly, and perhaps most obviously, adding more predictors could give the model additional context, particularly for harder-to-classify cases like Medium delays.

Classification Model Summary

Model Type	In-Sample Error	Out-of-Sample Error	Accuracy
Logit Regression	0.4295	0.3286	0.6414
Probit Regression	0.4296	0.3302	0.6698

SVM Binary Classification (Imbalanced Data)	0.1344	0.1344	0.8656
SVM Binary Classification (Balanced Data)	0.3791	0.6302	0.3698
Unpruned Multiclass CART (Imbalanced Data)	0. 3820373	0.3809928	0.619
Unpruned Multiclass CART (Balanced Data)	0.505	0.476	0.524
Pruned Multiclass CART (Imbalanced Data)	0.3821881	0.3811074	0.6189
Pruned Multiclass CART (Balanced Data)	0.5053831	0.4756496	0.5244
Random Forest (Imbalanced Data)	0.3816918	0.381312	0.6187
Random Forest (Balanced Data)	0.5041266	0.4770602	0.5229
XGBoost (Imbalanced Data)	0.3820893	0.3810555	0.6189
XGBoost (Balanced Data)	0.5050429	0.4754941	0.5245

Although the SVM binary classification model (trained on imbalanced data) yielded the highest overall accuracy at 86.56%, it is not directly comparable to the rest of our models due to its simplified task. Unlike the other models, which aimed to classify flight delays into four categories (OnTime, Short, Medium, Long), the SVM only handled a binary outcome (whether a flight was delayed or not). While it's a strong performer in terms of raw numbers, it doesn't meet the full complexity of our classification problem and, therefore, cannot be selected as the final model. Instead, we chose the XGBoost model trained on imbalanced data as our final model. It produced an out-of-sample error of 0.3811, tying with the pruned CART model on the same data but ultimately offering more modeling flexibility. XGBoost tends to generalize better and handle nonlinear patterns more effectively than traditional tree-based methods like CART. It also showed no signs of overfitting, as the in-sample and out-of-sample errors were nearly identical, suggesting a stable and reliable fit. While both models performed equally well on validation, XGBoost is the more robust choice for a four-class classification task.

```
Confusion Matrix and Statistics

          Actual
Predicted OnTime  Short Medium   Long
   OnTime      0      0      0      0
   Short   49243 159467  57121  16966
   Medium      0   6221  44670   8629
   Long        0      0   1536  22671

Overall Statistics

               Accuracy : 0.6188
                 95% CI : (0.6172, 0.6204)
    No Information Rate : 0.4521
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3612

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: OnTime Class: Short Class: Medium Class: Long
Sensitivity                 0.0000       0.9625        0.4323     0.46971
Specificity                 1.0000       0.3859        0.9436     0.99517
Pos Pred Value                 NaN       0.5639        0.7505     0.93655
Neg Pred Value              0.8656       0.9257        0.8089     0.92523
Prevalence                  0.1344       0.4521        0.2819     0.13169
Detection Rate              0.0000       0.4351        0.1219     0.06185
Detection Prevalence        0.0000       0.7716        0.1624     0.06604
Balanced Accuracy           0.5000       0.6742        0.6879     0.73244
```

When tested on the uncontaminated partition, our final XGBoost model (trained on imbalanced data) maintained an out-of-sample error rate of 0.3812 (accuracy of 61.89%), consistent with its validation performance. This stability across data partitions reinforces the model's reliability. While the model struggled with predicting On Time flights, it performed well in identifying Short, Medium, and Long delays, with especially strong precision and balanced accuracy in the Long category. These results suggest that although the model leans toward predicting delays, it does so in a way that meaningfully distinguishes between delay types rather than collapsing everything into one generic outcome.

One significant limitation to acknowledge is the persistent misclassification of OnTime flights. This is primarily due to the training data's inherent skew, as only about 13% of observations fell into the On Time category. While this imbalance may initially seem like a flaw, it's important to remember that the dataset comes from the U.S. Department of Transportation and reflects real-world reporting across over 1.7 million flights. In other words, the model isn't underperforming but instead surfacing a meaningful insight: flight delays may be more prevalent than most travelers realize.

From our perspective, the model's ability to distinguish types of delays (Short, Medium, and Long) is more valuable than forcing it to prioritize the least common outcome. For practical applications like staffing, resource planning, and traveler communication, being able to

differentiate between a 15-minute delay and a 90-minute one offers greater operational value than simply identifying flights that are on time. Future enhancements could explore rebalancing techniques or cost-sensitive learning to improve On Time detection. However, we believe this model reflects the true dynamics of air travel delays and provides useful predictions that align with how this data is structured in the real world.

Conclusion and Discussion

Our project began with a guiding question: *How can flight delay predictions help airlines optimize operations and improve customer experience?* To find an answer, we tested a wide range of supervised learning models on historical flight data to predict if a flight would be delayed and by how much. Along the way, we learned that answering this question isn't just about picking the most accurate model; it's about understanding the trade-offs that come with real-world data and designing features that reflect how delays actually unfold in practice.

From the start, exploratory analysis revealed meaningful trends: afternoon departures saw more delays, Saturdays were smoother, and some airports repeatedly underperformed. These patterns informed our modeling approach, such as breaking departure times into bins and introducing an airline-delay ranking. We debated incorporating origin and destination as features but ultimately left them out due to the complexity of encoding more than 30+ categories. In hindsight, creating a more scalable feature like a "route congestion index" would've allowed us to include that spatial dimension without overwhelming the model.

Model performance varied depending on the problem type and class balance. Our best regression model predicted delay length within 9.31 minutes on average. While the binary regression techniques were less accurate overall, they revealed clear and consistent patterns: earlier departures were less delay-prone, airline rank played a meaningful role, and delays tended to increase slightly as the week progressed. For classification, XGBoost (trained on the imbalanced data) stood out as the final model of choice. It achieved 61.9% accuracy, matched CART's performance, and remained stable on the uncontaminated test partition. While it did not predict On Time flights, a limitation tied to the data distribution, it excelled at distinguishing between Short, Medium, and Long delays, which are operationally more actionable for airlines. That level of detail can directly inform staffing needs, turnaround planning, and rerouting decisions in real-time.

Of course, no project is without its limitations. Our dataset, though sourced from the Department of Transportation, was from 2015 and may not reflect today's post-COVID airline operations. Additionally, some key variables like "weather delay" were too broadly defined to offer diagnostic value. It is also worth noting that while balancing the training data helped

improve fairness, it occasionally reduced the overall accuracy of our classification models. These limitations taught us that modeling is as much about design and interpretation as it is about computation.

Despite these challenges, the practical value of our findings remains strong. Airlines can use models like ours to proactively message customers about potential delays, reassign crews more efficiently, and even prepare for refund-triggering disruptions in accordance with DOT regulations. If we were to continue this work, we'd invest in more robust feature engineering (especially for route and weather data), consider further ensemble modeling strategies, and explore more advanced hyperparameter tuning techniques.

Ultimately, the value of our models went beyond accuracy metrics. While some were better suited for prediction than others, each revealed something useful about how delays unfold and where airlines might intervene. By translating delay data into decision insights, airlines can gain the tools necessary to manage uncertainty more strategically. In an industry where every minute matters, small insights can lead to big operational wins, and our models help make those insights possible.