Problem Statement

“RTD has seen a steady decline in ridership on both its trains and buses over the last few years, dropping from more than 103 million boardings in 2015 to fewer than 98 million last year.*” Additionally, the light rail has faced severe delays and cancellation in 2019, including those caused by outages, operator shortages, weather, and maintenance issues. Of the trains surveyed by RTD from December 2018 through November 2019, 83% of the station’s departures occurred after they were scheduled. In fact, CBS4 reports that in November 2019, RTD began surveying the public to see if service cuts or delays and cancellations are preferred in the wake of the driver shortage.

With decreased ridership, reliability, and workforce, RTD faces a triple-pronged problem that will continue to escalate moving into 2020. RTD is considering a new pricing program and has multiple line expansion operations underway. However, one key item to track and improve across all these efforts is consistent and timely adherence to train schedules.

Purpose of Analysis and Model

To help RTD in this effort in providing riders with consistent reliability, a model will be built to predict trains’ departures from each station on its scheduled line. Knowing a station’s scheduled departure, the minutes delayed until predicted departure and ultimately the forecasted departure time will allow prospective riders to plan accordingly. With wait times for a single line ranging from 15 minutes to over 30 minutes, it’s critical that riders board their expected train.

A model that can predict any future train’s expected departure has both B2C and B2B opportunities. The obvious usage of such a model can be seen with a direct to consumer model, wherein RTD can provide riders with the output of such a model. Currently, RTD has 2 independent offerings that seek to help riders plan trips using their services.

1. Rider Alerts (https://www.rtd-denver.com/app/alerts) on RTD’s public domain provides a listing of service lines affected by scheduled or known delays/cancellations.
2. Trip Planner (https://www.rtd-denver.com/app/plan) also on RTD’s public domain provides an interactive planning interface allowing visitors to select a starting and ending destinations as well as departure or arrival date and time.

Both of these services fill a large gap for RTD’s riders. However, there is room for improvement. Currently the Trip Planner does not consider detours, cancellations, and other special circumstances and encourages riders to view the Rider Alerts for those delays/cancellations that were scheduled/known. Instead, the Trip Planner considers scheduled services with adjustments based on real time vehicle location information.

* Denver Post: https://www.denverpost.com/2019/11/12/rtd-fares-regional-n-line/

Thus, a model that considers scheduled service, special circumstances/events, and leverages historic data to forecast service in the future could be used to improve current trip planning features offered by RTD. In a continued effort to increase ridership, the model can also be provided or licensed to local vendors who's websites provide directions via RTD. With a model that can predict future train service, patrons of other commercial establishments can better plan their logistics more appropriately using RTD.

Data Landscape and Overview

To begin the construction of this model, light rail data spanning December 2018 through November 2019 provided by RTD will be used. This data details service lines' arrival, departure, and scheduled departure at each station. RTD currently considers departure time compared to scheduled departure time when evaluating for train delays.

There are some known gaps in the provided RTD data that will impact the comprehensiveness of the model constructed. These gaps are listed below.

- RTD leverages Automated Passenger Counters (APCs) to collect the provided data. Approximately 30 percent of RTD's light rail vehicles have APCs installed. All of the commuter rail vehicles have APCs installed; however, not all APCs report every day for various reasons.
- Snow days are determined by examining the arrival times of buses at Union Station and Civic Center station. If more than 30% of the buses are more than 5 minutes late on that day, then the date is eliminated from the data set.
- Since RTD evaluates performance in regards to departure time, most end terminal stops do not feature a performance evaluation.

In addition to the provided RTD data, special occurrences and activity must be considered to further improve the Trip Planner offerings. The below areas are being considered as additional inputs into the model:

- Local hourly and daily weather from visualcrossings.com for 'Denver', 'Aurora', 'Greenwood Village', 'Centennial', 'Lone Tree', 'Westminster', 'Englewood', 'Littleton', 'Golden', 'Lakewood', 'Meridian', 'Wheat Ridge', 'Arvada'.
- Start game times of regular season games hosted by the Broncos, Nuggets, Rockies, and Avalanche from https://www.sports-reference.com/.
    - Preseason and postseason games will be considered where applicable
    - End game times are not currently available and will not be considered

Approach and Methodology

To identify which input variables should be considered for the resulting model, the correlation between independent input variables and departure delays will be measured. Various regression models will be used to determine the best fit and correlation coefficient for each

variable. Once a selection of input variables have been made, a predictive machine learning model will be constructed, trained, and tested in its ability to predict the continuous output variable of number of minutes delayed in future train departures.

Deliverables

The final Deliverable will be a jupyter notebook consisting of instructional walkthrough and corresponding code to be executed at each step in the analysis and model creation, from data ingestion to model output and assessment. Along with the Jupyter Notebook, a powerpoint deck will be created that details the problem being solved, analysis, and recommended solution.

Data Wrangling Effort

Library and Initial File Imports

For data import retrieval and manipulation, the following libraries were imported:pandas, numpy, datetime, and timezone. For initial discovery through visualization, the following libraries were imported: matplotlib.pyplot and seaborn.

Two files were initial imported: 1) station departure records of light rail trains and 2) station identifier translations. Each file was imported as a csv with a ‘;’ (semicolon) delimiter.

Stop Identifier Cleaning

To leverage stop names used by the public, the train data was merged with the stop name lookup file to bring the stop name into the station departure data. Some STOP IDs do not exist in the lookup file and thus do not have stop names. For these stations, the MAIN_CROSS_STREET field will be substituted for the STOPNAME.

To complete these translations, a dataframe was created of relevant and unique STOP IDs and their corresponding STOPNAMEs. This file was then merged into the train data. A separate list of STOP IDs were created that are only available in the train data, by making use of the indicator parameter on the merge method. These stops’’ MAIN_CROSS_STREET value were used to replace the STOPNAME in place in the train data.

Removal of Unnecessary Fields

The following fields were removed as they only feature nulls or a consistent value across the entire dataset: ‘ON_DETOUR', 'SL_COUNT', 'SM_COUNT', 'AVL_COUNT',' SL_AVAILABLE', 'SM_AVAILABLE', 'AVL_AVAILABLE', 'APC_AVAILABLE'.

Creating Consistent Formatting in Datetime Fields

There are 3 native datetime fields in the train data: 'TIME_ACTUAL_ARRIVE', 'TIME ACTUAL DEPART', ‘TIME_SCHEDULED’. The fields represent when the train actually arrived at the recorded station, when it actually departed, and when it was supposed to depart, respectively. Please note that expected arrival is not native in the data.

TIME_ACTUAL_ARRIVE and 'TIME ACTUAL DEPART were set to December 31, 1899 as defaulted by Oracle. Their timestamps were appropriately set. TIME_SCHEDULED carried both an accurate date and time of the scheduled departure. Thus, after transforming all fields to datetime objects, the date from TIME SCHEDULED was merged with the time fields of TIME_ACTUAL_ARRIVE and TIME ACTUAL DEPART to create accurate datetime fields, ARRIVAL and DEPARTURE, respectively.

However, some train arrivals and departures straddle midnight, meaning their ARRIVAL, DEPARTURE, and TIME_SCHEDULED may occur on different days. Thus, a field was created for both ARRIVAL and DEPARTURE, indicating if there was a 12 hour (noted as 43200 seconds) difference between the respective time field and TIME_SCHEDULED. The DEPARTURE and ARRIVAL day stamps of these records were then adjusted accordingly.

Additionally, the field TIME_TO_SCH_DEPART was added that creates a range from ARRIVAL to TIME_SCHEDULED (departure) in seconds to illustrate trains that arrive after they were supposed to have already departed.

Fill Empty County Field for Weather Merge

Hourly weather by county can be obtained to complement train data. However, some stations have NaN COUNTY information and ‘Douglas’ county is not available from https://www.ncdc.noaa.gov/cdo-web/.

Sky Ridge Station, Lone Tree City Center Station, and Ridgegate Parkway Station are in Douglas County but close to the Arapahoe County edge. These stations’ COUNTY value was replaced with ‘Arapahoe’.

A unique dataframe of STOPNAME with NaN corresponding COUNTY values was created and saved to a local csv file. These stations were then researched to find their corresponding counties and read back in as a new dataframe. The value of the merged County field was then used to update the LOOKUP_COUNTY field and a final list of 4 distinct LOOKUP_COUNTY values was confirmed: 'Denver', 'Adams', 'Arapahoe', 'Jefferson'.

Rockies Data Cleaning

The 2019 Rockies game results were obtained from sports-reference.com and saved to an excel file. However, the data did not include start times. For start times, the original 2019 schedule published by the Denver Post was used. Both files were read in as data frames and datetime objects transformed accordingly.

There were 4 Date values in the games file that were set to NaN after transforming the field to a datetime objects. These 4 values correspond to double header games played on ‘2019-07-15’ and ‘2019-07-24’. These dates were set manually after being discovered and researched.

Away games were removed from both files and the All-Star Break record was removed from the times file by dropping the rows corresponding to the applicable indexes. Indexes were then reset.

For additional validation for future merging of times and games, the unique list of opponent abbreviation (‘Opp’) ,i.e. LAD, from the games file was merged with the corresponding opponent definition (‘Opponent’), i.e. Los Angeles Dodgers, from the times file. A dictionary was created with these opponent fields and used to map to a new field in the times file labeled as Opp_Cd.

Finally, the times and games files were merged into a single file using Date and Opp from the games file with Sch_Date (renamed from Date) and Opp_Cd from the times file. Further validation was performed and adjustments made to correct for the following anomalies:

- Game on May 8th was canceled
- Cancelled game was replaced with game on August 26th starting at 3:10pm
- Double-header on July 15th started at 12:10pm.

Broncos Data Cleaning

Broncos data for the 2018 and 2019 seasons were obtained from sports-reference.com as excel files. The files include both date and time of kickoff for each game played by the Broncos. They were both read in as dataframes, combined, and away games and bye weeks removed.

Nuggets Data Cleaning

Nuggets data for the 2018-2019 and 2019-2020 seasons were obtained from sports-reference.com as excel files. The files include both date and start time of each game played by the Nuggets. They were both read in as dataframes, combined, and away games removed.

Avalanche Data Cleaning

Avalanche data for the 2018-2019 and 2019-2020 seasons were obtained from Altitude TV Network as excel files. The files include both date and start time of each game played by the Avalanche. They were both read in as dataframes, combined, and away games removed.

Setting Game Times to 24-Hour Clock

The train dataframe features times based on a 24-hour clock. For future merging, sport game start times had to be converted to the same format.

The Rockies Time field was converted to a datetime.time value after manually fixing the value set as 1pm as it was inconsistent with other formats.

The Broncos timestamps were set according to US/Eastern. After merging the date with the time as a datetime object, the timezone library was used to make the datetime aware of its time zone and subsequently convert to US/Mountain time.

Lastly, the Nuggets and Avalanche games were both set to datetime.time objects.

Merging Games into Train Data

For each day in the train data, there can only be up to one game played by the Broncos, Nuggets, and Avalanche. Thus, each one was merged into the train data, incrementally, by joining on the respective Date columns.

However, there was a single day in which the Rockies played 2 games. Thus, 2 data elements were created to merge into the train data - time of first game and time of second game. To create these columns, the rockies data was grouped by Date and Opp and the first method was used to grab the first game on each day. The grouped dataframe was then merged once again to the original rockies dataframe grouped according to the last game played on each Date and Opp combination. Duplicative times across the first and second game times were set to blank spaces and then filled with NaN using a regex expression within a replace method. The day of

the double header was then used to confirm the appropriate result. Finally, the consolidated rockies dataframe was then merged into the train data similar to the sport merges above.

Re-Calculating Dwell Time

DWELL TIME, while native in the original train data, was not always completed. Thus, WAIT TIME CALC was created by identifying the number of seconds between the actual departure and arrival of each train at each station.

However, this produced around ~260 rows in which the WAIT TIME CALC value was negative. These negative rows occurred when the date stamp from TIME SCHEDULED was used to replace the default date in TIME ACTUAL ARRIVAL to produce the ARRIVAL value. An inaccurately negative WAIT TIME CALC was produced when the actual arrival was shortly before midnight and the departure was immediately afterwards. To correct these mis-transformations, the day of the ARRIVAl values was reduced by 1 day for records in which the WAIT TIME CALC was less than -60 seconds. Please note that there is a single record in which the actual arrival of the train is 1 second after the departure. This record remains unaffected for now.

Merging in Revised Weather Data

Weather from NOAA was scarce and featured many data type conflicts across the county fields sourced. As a result, new weather data was sourced from Visual Crossing (visualcrossing.com) and spot checked for its accuracy. Unlike NOAA, Visual Crossing features hourly weather by city. As a result, a similar exercise was performed to ensure each record in the train data had a value for CITY.

Unfortunately, the native data featured 2 conflicting values for CITY for stations 'Red Rocks Community College Station' and 'Sheridan Station'. These records' CITY field were set to 'Lakewood' and 'Denver', respectively. Additionally, missing CITY values had to be researched and set for certain STOPNAMEs.

The weather data from Visual Crossing was then merged into the train data using pandas' merge_asof method, setting a timedelta tolerance of 1 day, meaning the difference between the recorded hourly weather and train DEPARTURE must not exceed a day. However, since weather was sourced by hour for the entire date range for each city, this tolerance is likely exhaustive.

The sourced weather data featured a column WEATHER TYPE that contained a listing of weather descriptions, such as Drizzle, Fog, Heavy Snow, etc. These listings were in no particular order and featured combinations of 22 unique descriptors. To create a cleaner version of this information, individual fields were added for each of the 22 descriptors, representing a

flag for if a given record included that particular type of weather. For example, a row with a value of 1 in the ‘Heavy Snow’ column indicates that heavy snow was recorded at the time of the DEPARTURE.

Denoting End of Line Stations

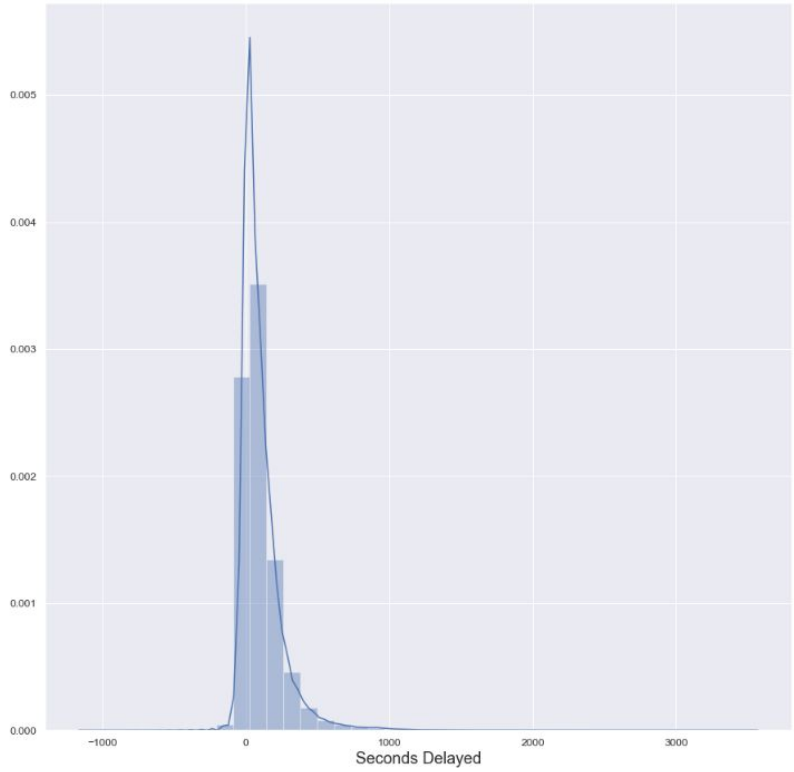
While sorting trains departures by SORT_ORDER, the provided identifier for ordering stations in each line, the following key insights were gathered:

1. Not all of the provided lines’ stations are included in the dataset
2. Some lines’ final station are missing in the dataset
3. DEPARTURE_DELAY is highly skewed at the end of the line as timeliness is assessed by departure, which does not concern passengers as final stations.

To combat numbers 2 and 3 above, each line’s stations were explored both within the dataset and online. After completion of this research, an END_OF_LINE flag was added to the train data to be set to 1 if the record features a station at the end of the corresponding route/line. This process was done for each unique combination of BRANCH (line) and DIRECTION_NAME.

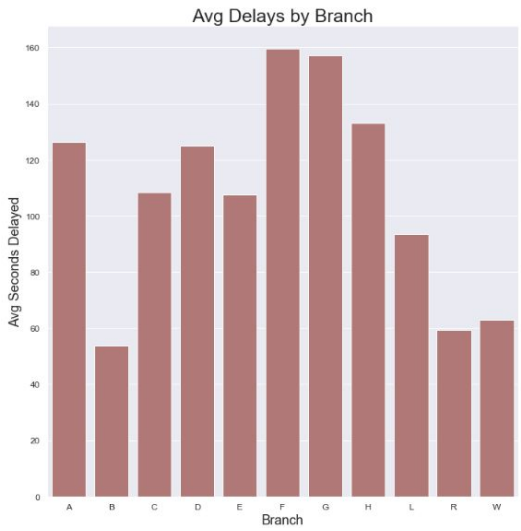
Data Exploration

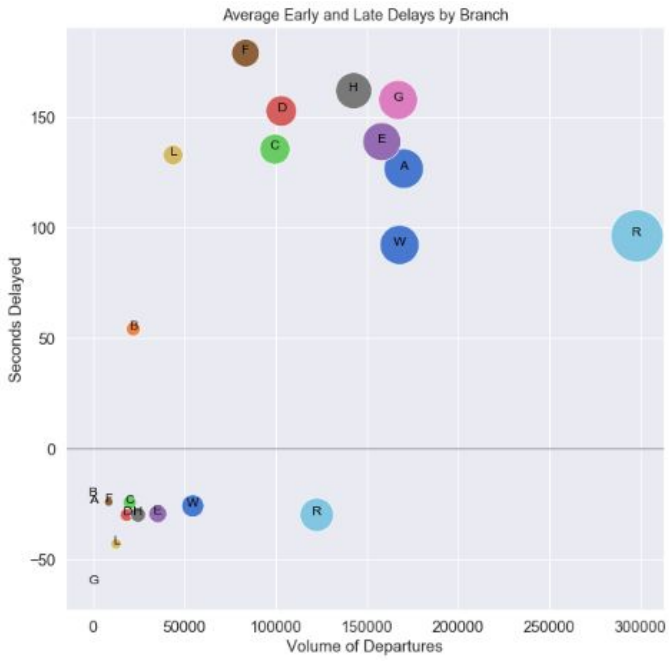
Follow successful ingestion of the RTD light rail data, several very evident insights came to light. First and foremost, 83% of trains are departing after their scheduled departure. Furthermore, while there is a substantial volume of early departures, the average departure is still occuring 1 minute and 42 seconds after its scheduled time. Below is a brief overview of some of the variables explored in pursuit of seeing what may cause these RTD delays. Sources of information and variables include RTD train data, local sporting events, and hourly weather.



Train Line Breakdown

Evaluating departure delays by line (also known as branch internally to RTD), shows substantial variance in average delays. The best performing line (by departure delays) is line B, with an average delay of just under a minute. Alternatively, line F averages a delay of nearly 3x that of line B.





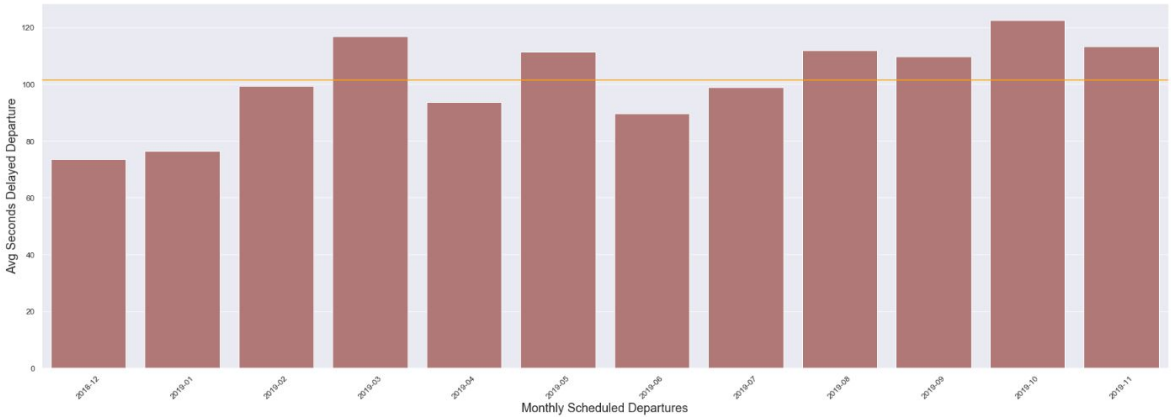
While lines F, G, and H show the highest average delays, respectively, F does not feature the highest volume of delays. On the contrary, G and H shows a high number of departures, thus extrapolating the impact of their delays.

Line R seems to feature the most departures with the average delayed departure around a minute and a half and the average on-time/early departure roughly 30 seconds before scheduled.

A one-way F test indicated a statistical significance (p-value ~ 0.00) in the difference in average departure delay among the different lines.

Exploration into Seasonality

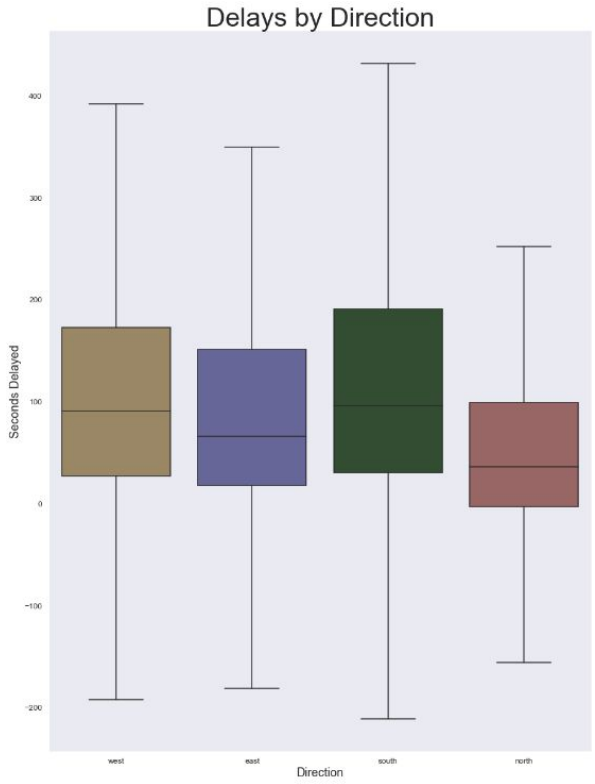
While seasonal trends cannot be predicted with only a year's worth of data, there are some spikes and climbs seen across December 2018 through November 2019. Both averaging delays using monthly snapshots and rolling averages (weekly, monthly, and quarterly) show a rather rapid climb between January to March and a spike in October 2019.

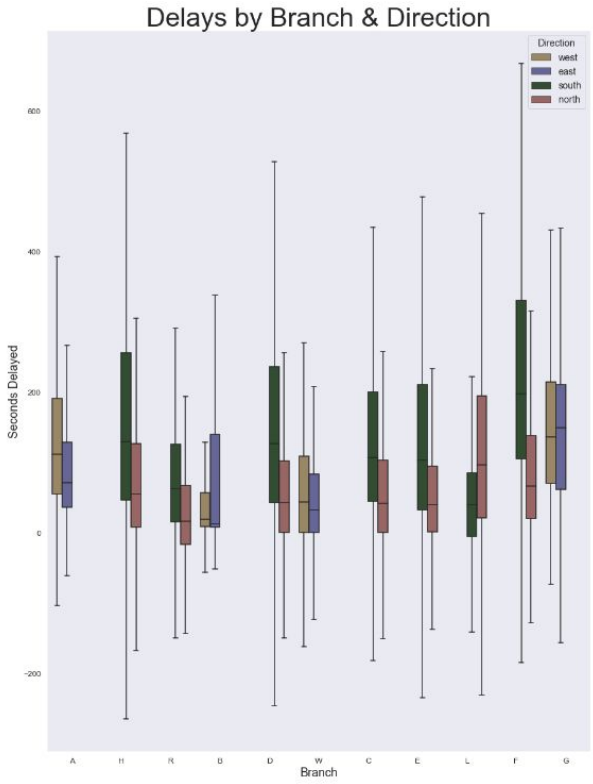


The initial climb may in fact coincide with weather, so there is reason to explore weather's effect on delays further. The secondary spike in October however, may correlate to the driver shortage that became widely published in the final quarter of 2019, as it caused many train delays and cancellations.

Comparison of Direction, Time Period, and Station

In pursuit of additional variables for modeling consideration, several additional pieces of RTD data were compared. Interestingly, southbound trains tend to be more delayed while northbound trains perform better than the other 3 directions. Furthermore, this contrast is evident across all train lines with the exception of line L, wherein southbound trains outperform northbound trains.

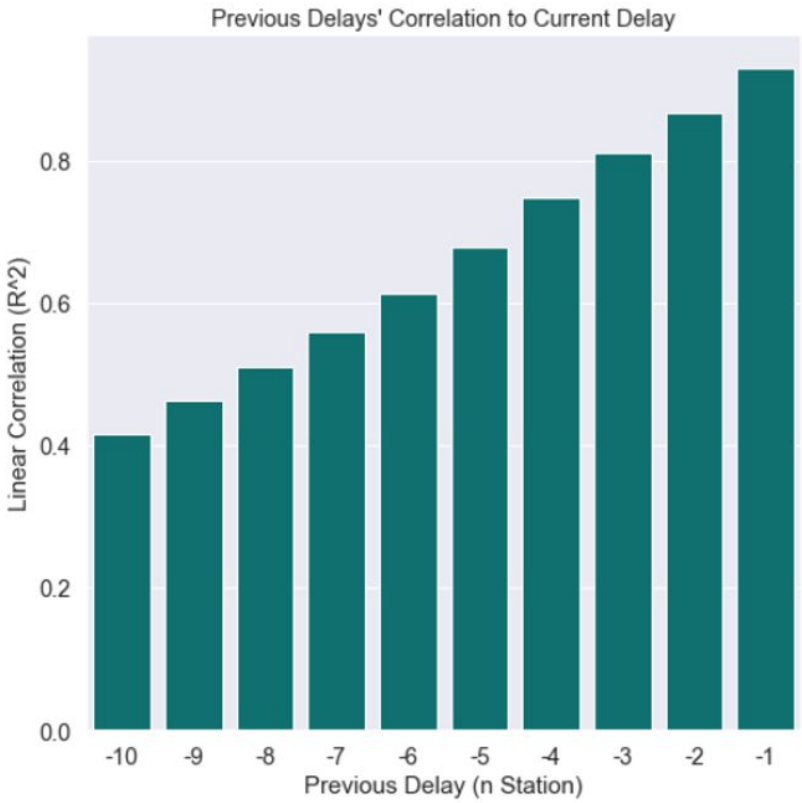




Comparing service (time) periods, i.e. morning rush hour and late evening, did not present any immediate findings but the service period classification should be considered for further analysis. In contrast, comparing departure delays across stations showed significant variance, presenting itself as possibly a significant predictive variable.

Impact from Prior Delays

As to be expected, delayed departures at prior stations will inevitably have an effect on current and future delays. However, what is interesting is that these delays do not continue in a flat, nor increasing manner. Instead, the linear correlation of prior delays decreases from ~0.85 as one progresses further into a train's previous departures. In other words, one may hypothesize that train operators are able to make up time as the train progresses through its schedule.



Local Sporting Events Influence

To better understand the effect local sporting events may have on train delays, the average delay on game days were compared to days without games. Specifically, the average delay of Rockies game days in Denver were compared to that of days during which the Rockies did not play in Denver. The same snapshot analysis was performed for the Avalanche, Nuggets, and Broncos.

The average comparison did not show any immediate insight for the 4 teams, with the exception of the Broncos. Initial views showed a considerable increase in delays on Broncos game days. However, since most of these games are played on Sundays, comparing Broncos Sunday game days with Sundays during which the Broncos did not have a home game revealed the opposite - trains perform slightly worse on Broncos Sunday game days.

Weather Exploration

Multiple weather variables are to be considered in their effect and predictive abilities in determining train delays. To start, linear correlations were obtained between departure delays and snow depth, precipitation, and wind speed. While all p-values are near 0, the correlations

are very small: 0.1, (0.01), 0.01, respectively. Further exploration into weather conditions and the overlap of various measurements is recommended.

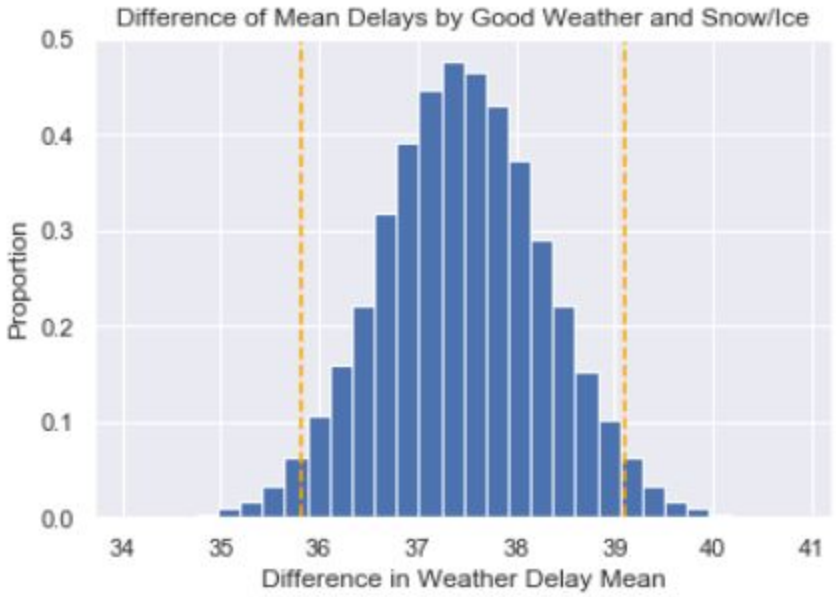
Inferential Statistics Analysis

Similar to the prior exploratory data analysis phase, brief statistical analysis was conducted using the combined RTD light rail, local sporting event, and weather dataset. Beginning with the previously cleaned dataset, inferential statistics using bootstrap and Bayesian statistics were completed. These exercises are described below.

Difference of Delays between Good and Snowy Weather

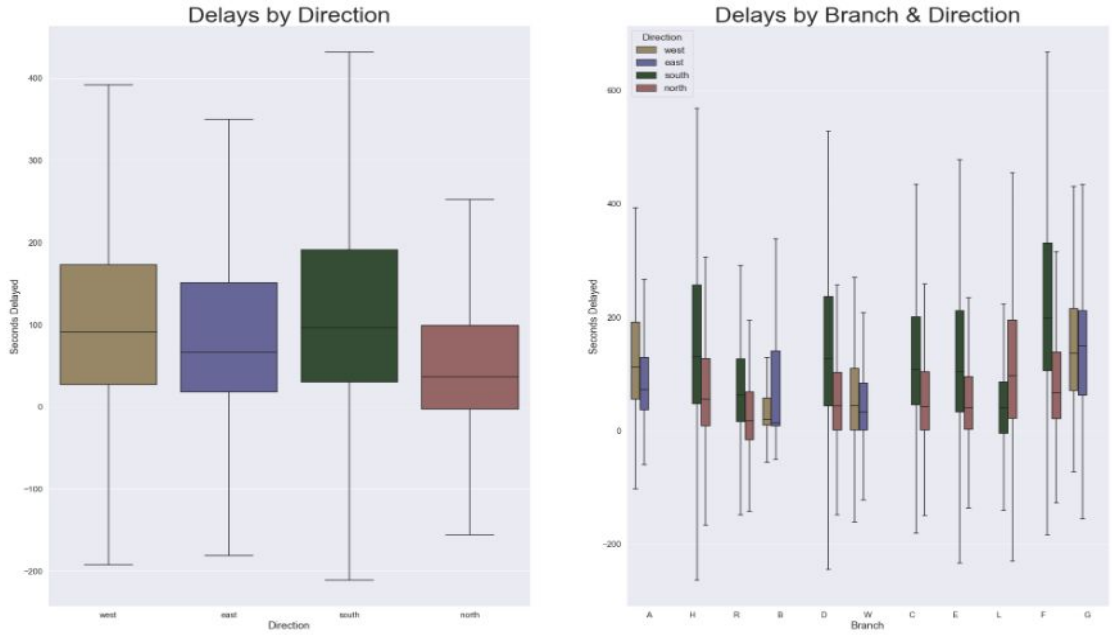
Bootstrap inferential statistics was used to determine if one can be confident that the average delay on a snowy or icy day will be different than a “clear” day, specifically a day without snow or ice at the time of departure. To begin, delays were segmented by those that have a weather type corresponding to snow, ice, freezing rain, or hail and those that have none of these weather characteristics.

For each resulting series of delays, 50,000 series of the same length were generated by using a random selection (with replacement) from the empirical data. From each randomly generated series, a mean was captured. The 50,000 resulting “snow day” means were compared to those of the “clear days”. This produced a 95% confidence interval in the difference of means (in seconds) of [35.82728009, 39.10296803]. More importantly, this confidence interval does not near 0, indicating that “snow days” have statistically higher mean delays than non-snowy days.



Isolate Southbound Train Performance

One of the more intriguing findings from the initial exploratory data analysis was that southbound trains seem to underperform compared to other directions.

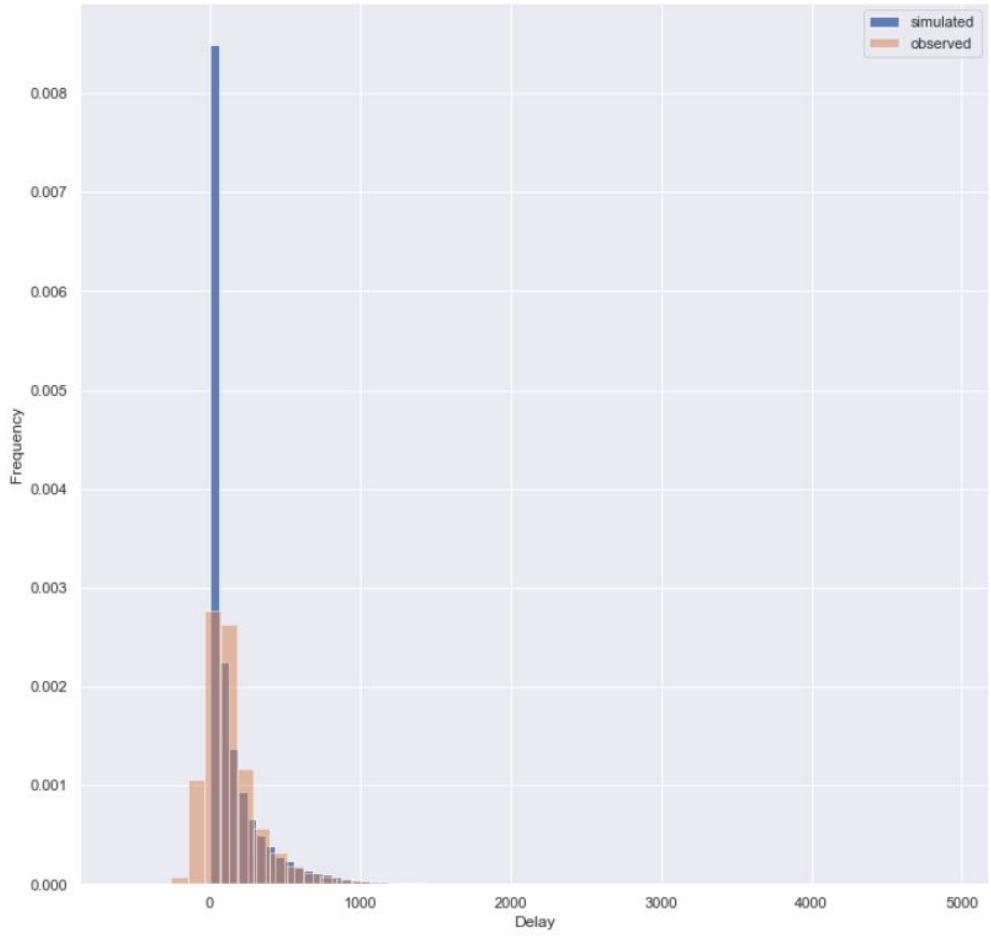


To test this hypothesis, permutation methodology was used to compare the average delays of southbound trains with non-southbound trains. Specifically, the delays of southbound trains were combined with delays of non-southbound trains and randomly sorted. Two new arrays were created to mirror the original lengths of the southbound and non-southbound delays. Lastly, the difference (called bootstrap replicates) of these 2 randomly generated arrays was captured.

Identifying the portion of the bootstrap replicates (using the above permutation method) that are greater than or equal to the difference found in the empirical data results in a 0% p-value. Such a low p-value indicates that the null hypothesis of southbound and non-southbound trains having the same average delay should be rejected.

Simulating Gamma Delay Distribution

Initial histogram view of delays suggested a right tail and possibly gamma-like distribution. Thus, a simulation was conducted using a gamma distribution and derived alpha and beta parameters was created to compare the simulated gamma output to the possibly gamma empirical distribution. While the fit is not perfect, it is close.



Predictive Modeling

The data gathered throughout this effort included information that would be available several days before and a train departure and those that would only be available only minutes before a train station departure. The first category (referred to here as “long term”), which includes the majority of the data gathered, includes local sporting events and to some degree, weather, as weather forecasts permit. The second category (referred to here as “real-time”) exclusively refers to the train departure delays at immediately prior stations. For example, the E line going south on any given day will visit the Belleview Station before the Orchard Station. Thus, the delay at the Belleview Station will only be known minute(s) before arriving at the Orchard Station. However, while not linear, there will inevitably be some degree of correlation between the 2, as will be revealed below.

Model Overview

The delineation of data through their “long term” or “real-time” accessibility, as described above, was carried through to separate model evaluation avenues. The data available 1+ days in advance can be used to predict more severe delays due to local sporting events and weather forecasts. Similar data updated to the moment, combined with the real-time, can be used to predict more granular delays to provide riders with knowledge that they have an extra few minutes without a risk of missing a train’s departure. In a sense, their prior knowledge obtained through standard train schedules can be augmented in the moment.

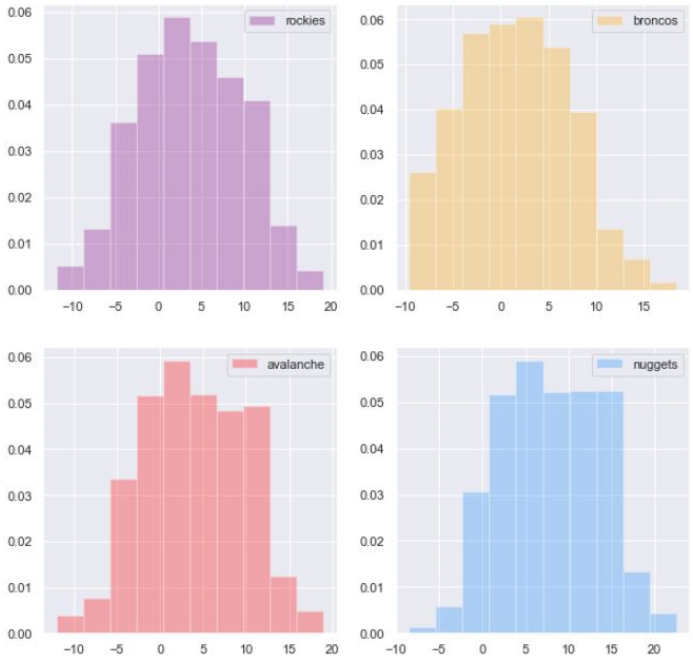
Several supervised learning algorithms were used. A starting point was established with a lasso regression algorithm with a very small regularization constant. Subsequently, a random forest ensemble algorithm was used for both models with hyper-parameter tuning for max depth. Lastly, based on results from the random forest model, the “real-time” model dataset was also used with an extreme gradient boosting algorithm with more thorough hyper-parameter tuning.

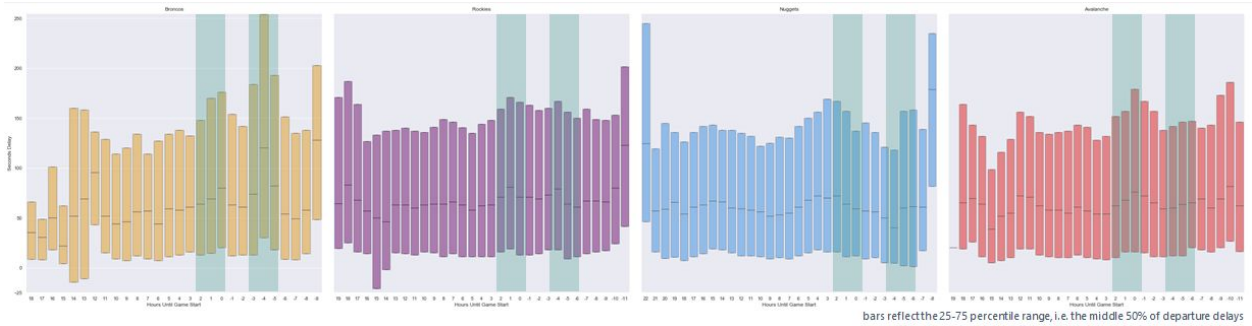
Final Data Preparation

To begin final data preparation for model assessment, the prior dataset was read in and reviewed. Since RTD supports delays only based on departures, stations at the end of each respective route cannot be assessed; thus these stops were removed.

While prior efforts included measuring the correlation between immediately prior delays and the current delay, these data elements were never added. A new function was built to add these lag features for prior delays up to the 4th prior delay for each departure. These additions are the “real-time” data elements.

Additional fields were added to each record corresponding to the hours until each local sporting event began. This feature includes both negative and positive figures as a train departure can happen before or after a game begins. Days in which a respective team did not host a local game are filled with 100,000 as 0 would indicate train departures occurring very close to game start, which could be highly misleading.





Trending the middle 50% of delays near game times shows possible upticks in the 2 hours leading up to the beginning of Broncos and Avalanche games. A similar but less obvious pattern is seen with Rockies games. Interestingly, there is also a spike around where Broncos games would likely end, approximately 3-5 hours after kickoff.

Next, a multitude of strategies were used to address both missing weather data and data not conducive for predictive modeling. Based on the native weather descriptions gathered earlier, ranges of ordinal values were constructed for rain, snow, and ice conditions. These ranges are shown below.

Rain Description	Rank
Light Drizzle	1
Drizzle	2
Heavy Drizzle	3
Mist	4
Light Rain	5
Rain	6
Heavy Rain	7
Rain Showers	8
Thunderstorm	9

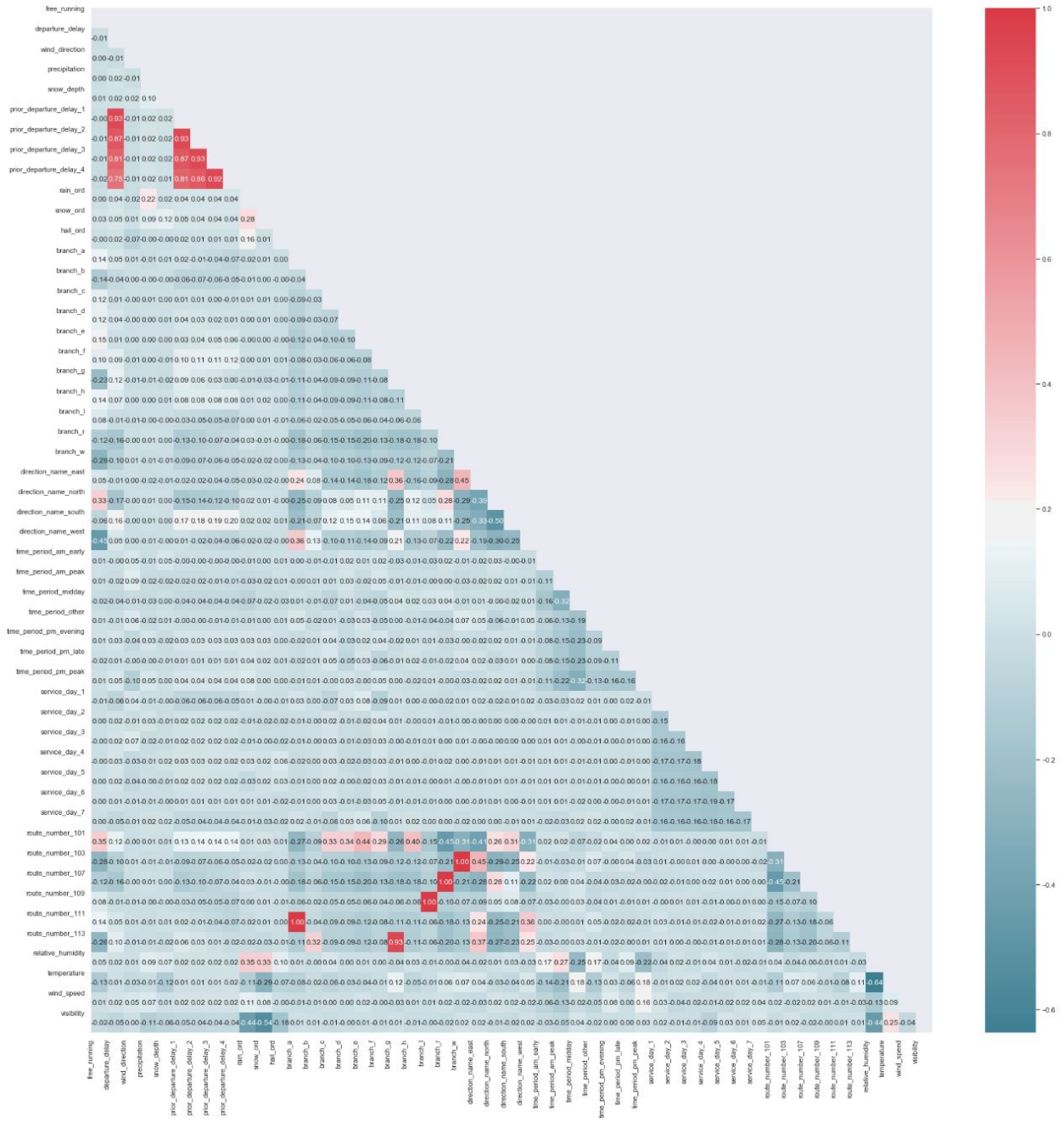
Snow Description	Rank
Light Snow	1
Light Rain and Snow	2
Snow	3
Blowing or Drifting Snow	4
Heavy Snow	5

Ice Description	Rank
Light Freezing Drizzle/Freezing Rain	1
Light Freezing Rain	2
Freezing Drizzle/Freezing Rain	3
Hail Showers	4

Other quantitative weather fields featuring null values were filled using other records. Null wind direction, which features wind speed, was filled with the average value of the field. Null precipitation and snow depth fields were filled with 0. Relative Humidity, Temperature, Wind Speed, and Visibility were filled with the nearest values in time, with thresholds of 5 hours, 3 hours, 4 hours, and 3 hours, respectively.

Dummy indicator fields were created for 'STOPNAME', 'BRANCH', 'DIRECTION_NAME', 'TIME_PERIOD', 'SERVICE_DAY', 'CITY', 'LOOKUP_COUNTY', 'ROUTE_NUMBER' as well as the month of the scheduled departure. Null values in the free_running column were set to 0.

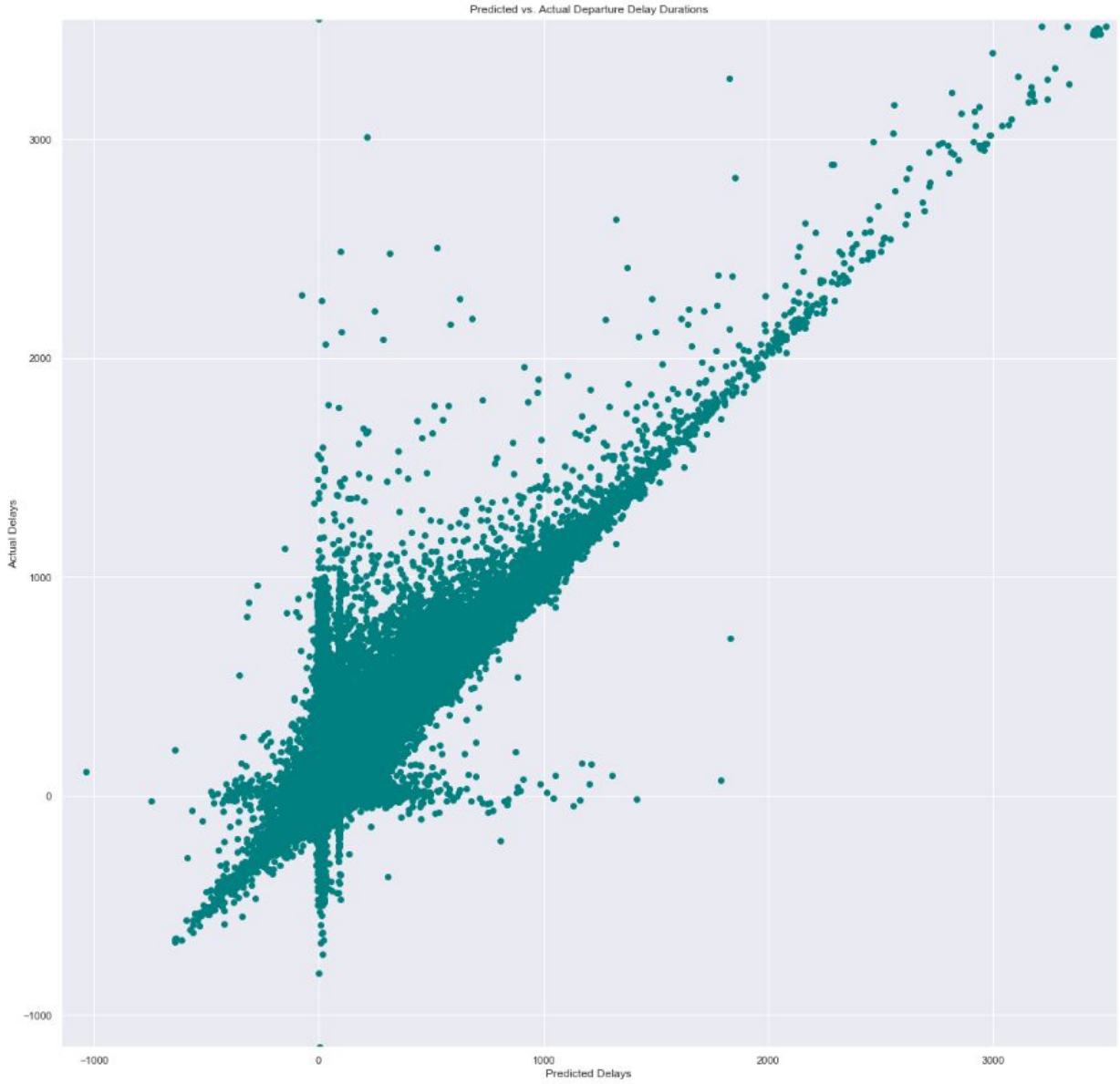
Finally, correlation factors among the features were also assessed to ensure there were not any instances of unknown correlation and possibility for feature reduction. There were some expected areas of correlations, such as among the lag features and various combinations of routes, cities, counties, and stations. However, no features were removed as the location-based correlations were not consistent across all combinations.



Lasso Linear Regression

LassoCV was used to define an appropriate level of regularization, which resulted in a very low constant of ~0.0004. This recommendation of low regularization should be considered in later models, such as extreme gradient boosting. Using the suggested level of regularization, lasso regression regression was trained on a validation set (not used for hyperparameter tuning) and predictions were made on a final test/holdout set. Reasonable R-squared and mean squared error results were produced of 0.89 and 2375, respectively. These were used as baselines in more advanced algorithms in the following sections.

Furthermore, plotting the predicted and actual values shows an obvious 45 degree trend in the data, indicating there are no unexpected spreads of residuals. There is more noticeable error between 0 and 2000 seconds delay wherein the actual delay was higher than predicted. This could be the case due to unforeseen sudden train delays not present in the data, such as mechanical issues.



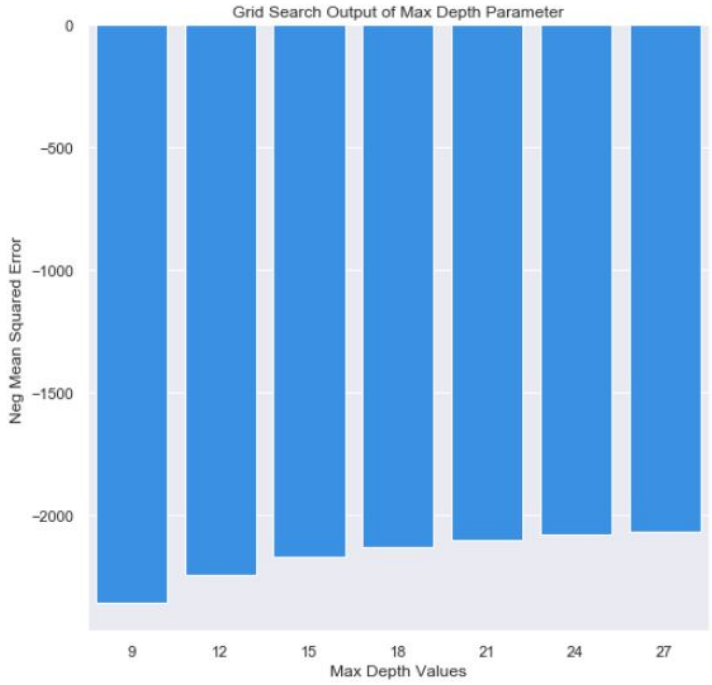
Random Forest: Model Creation and Evaluation

With roughly ~ 1.7 million records in the entire remaining dataset, it was split into a training, validation, and testing set similar to the Lasso Regression approach. Specifically, the test set was 70% of the entire dataset, and the remaining 30% was split into 20% train and 80%

validation. The training set was used for initial grid search to select an appropriate max depth hyperparameter while the validation set was used for a secondary measurement of over-fitting. A max depth range between 9 and 30 was evaluated. Lastly, the model was re-trained on the entire training and validation set and performance measured against the holdout, or test set.

Random Forest was used as a general testing ground for both the “long term” and “real-time” and a similar approach was used for both. However, the “real-time” model was first to gain an understanding of ceiling performance as it should inevitably perform better than the “long term” model.

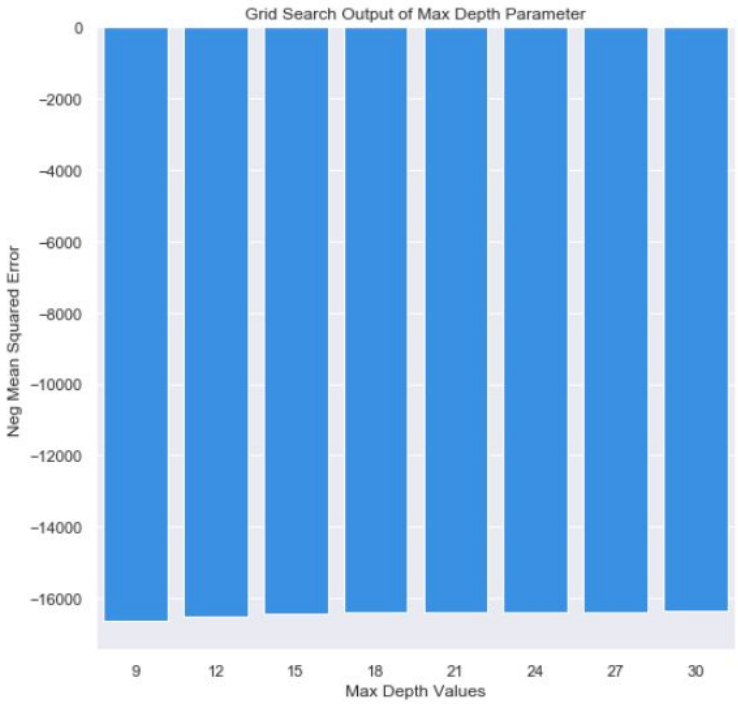
The “long term” model gridsearch indicated that a max depth of 27 was best, using the negative mean squared error as a scoring mechanism. However, this is a rather high max_depth that risks over-fitting. Thus, a follow up evaluation was conducted to compare the prediction on the training set with the prediction on the validation set. This did in fact prove a decent amount of over-fitting was occurring.



As the gridsearch output shows, there is minimal improvement from a max_depth of 21 onward. Thus, once again the model was trained on the training set and predictions made from the validation set with a max_depth of 21. Interestingly, despite the over-fitting of a max_depth of 27, its generalization on the validation set was superior. Thus, a max_depth of 27 was selected moving forward for the “real-time” model.

In a similar fashion, the “long term” model favored a higher max_depth recommendation of ~30. However, again this risks drastic over-fitting. Furthermore, the difference in performance measurements among the gridsearch max_depth values tests were minimal. Thus, instead of a max_depth of 30, which would drastically increase training time, max_depths of both 9 and 18

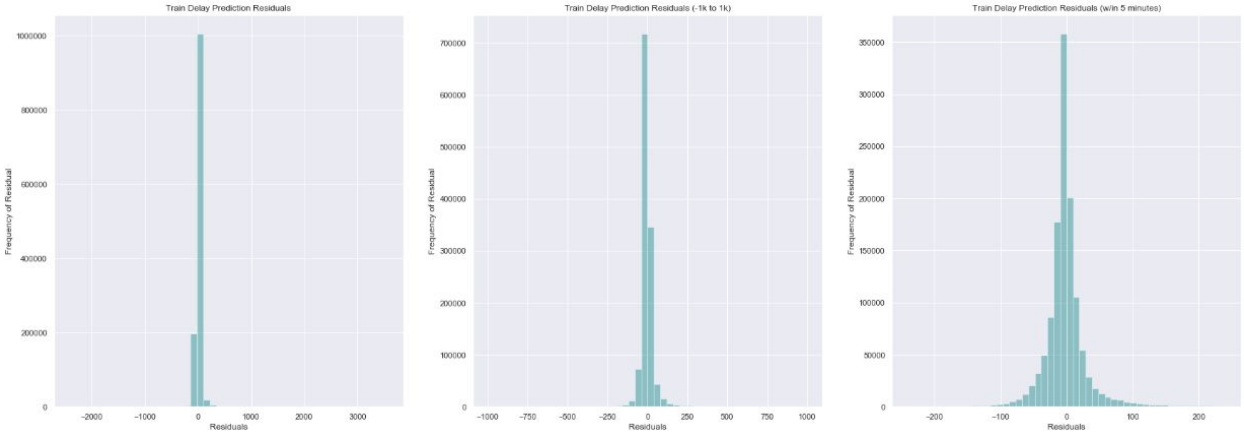
were used to compare generalization performance on the validation set. A max_depth of 18 performed best and was chosen for continued assessment.

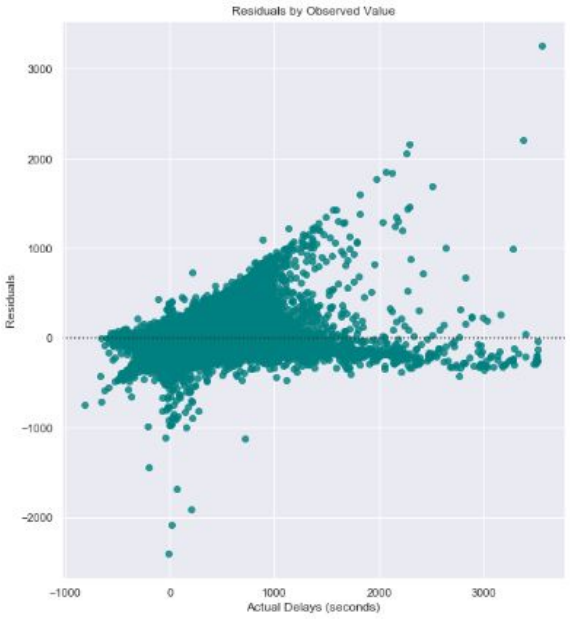


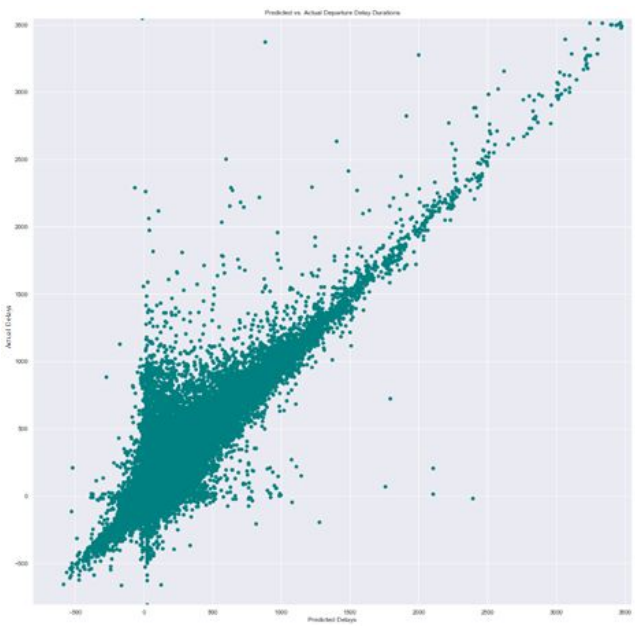
Random Forest: Model Performance

Real-Time Model

Evaluating the residuals across the spectrum of departure delays, shows a vast majority of the error within 2 min difference, or 240 seconds of the actual delay. As to be expected, a vast majority of the more extreme prediction errors are wherein the actual delays are much greater than the predicted delays. Unforeseen circumstances, i.e. mechanical issues, not captured in the data may be responsible for large delays.



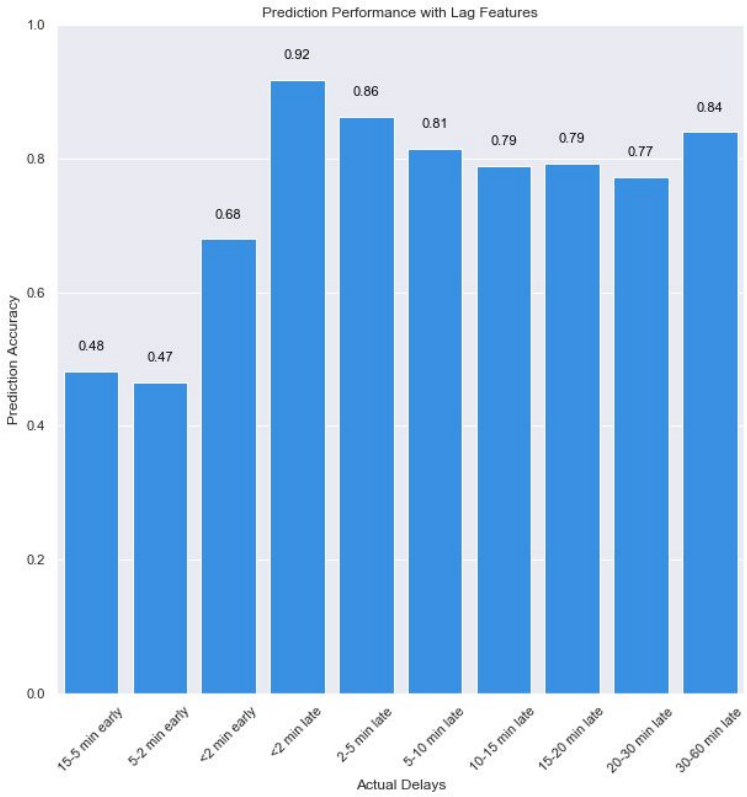




To further assess real-world performance and applicability, the actual delays of the test set were cut into buckets, as depicted below.

Time Bucket	Seconds Range
'15-5 min early'	-900 - -300
'5-2 min early'	-300 - -120
'<2 min early'	-120 - 0
'<2 min late'	0 - 120
'2-5 min late’	120 - 300
‘5-10 min late'	300 - 600
'10-15 min late’	600 - 900
'15-20 min late'	900 - 1200
'20-30 min late'	1200 - 1800
'30-60 min late'	1800 - 3600

The percentage of above time buckets for which the same time bucket was predicted corresponds to the model’s ability to estimate a delay with a given range. The percentage accuracy of matching the prediction time bucket to the actual time bucket is depicted below for range.



At first glance, train departures leaving within 2 minutes of their scheduled time are correctly marked as doing so 92% of the time, which is expected to be the highest as this corresponds to nearly on-time departures. However, the departures delayed between 2 and 10 minutes are correctly predicted 81-86% of the time. Since this particular model incorporates the lag features, these minor but noticeable delays are useful in the moment when riders are hoping to catch the next train. Developing a probability distribution for a train's delay to fall into one or multiple of these time buckets could be helpful for riders as well. Since a train's situation changes on a minute basis, updating these probabilities and alerting riders of these changes and risks for delays could also help to update riders with expected timelines as they develop.

Also worth noting is the time sensitivity for the "immediate model". Trains will progress on their respective routes on a minute basis, thus the time to compute and update predictions is critical. Furthermore, random forest can take considerable time as compared to other tree-based models, such as gradient boosting. Comparing prediction run times of newly generated departure records across potential hyperparameter values would be beneficial from a production standpoint.

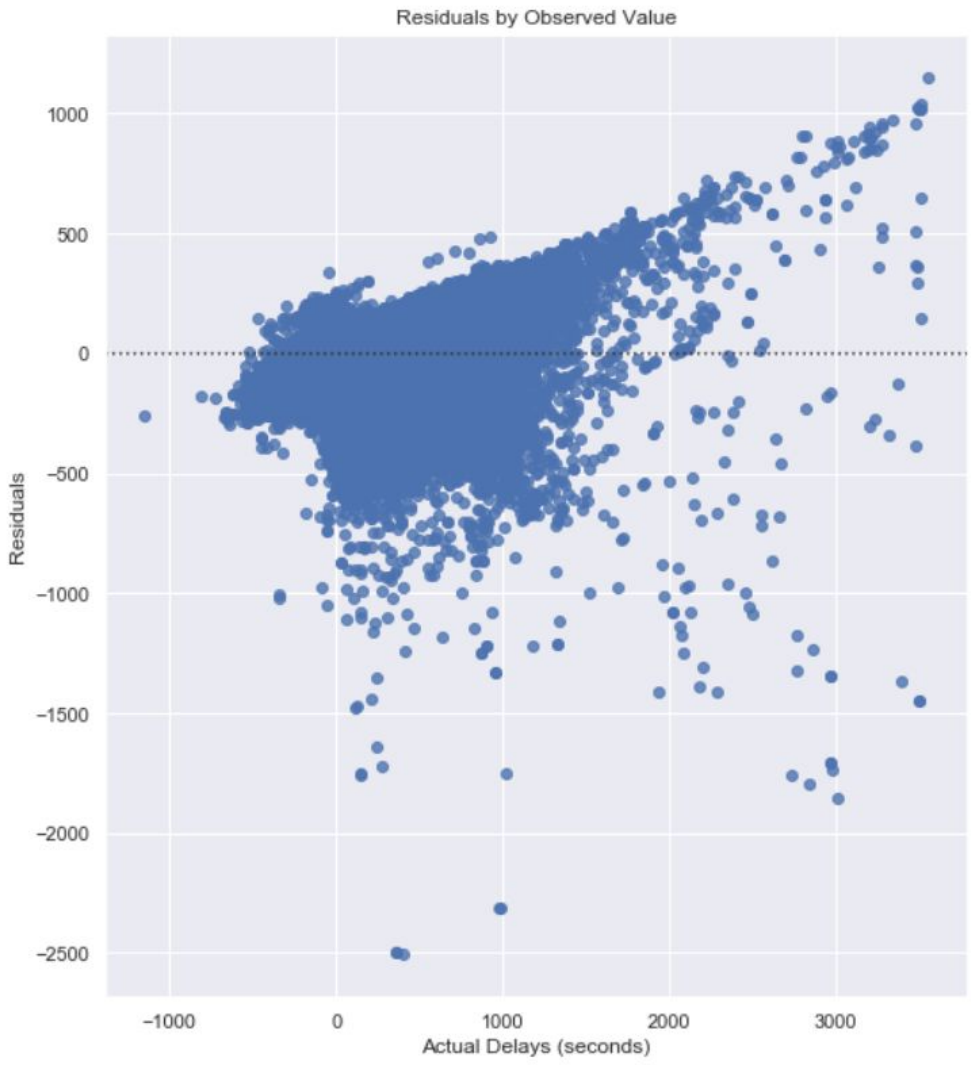
Based on the consolidated RTD data, the maximum number of train departures occurred between 2019-08-30 16:13:00 and 2019-08-30 16:13:59, which 18 departures in less than 60 seconds. Evaluating the difference in prediction runtime between a max_depth of 27 and 18 with these 18 records indicated very similar results confidence intervals. Thus, comparative

performance among these max_depth hyperparameters may not be of immediate concern if simply running new predictions between trains.

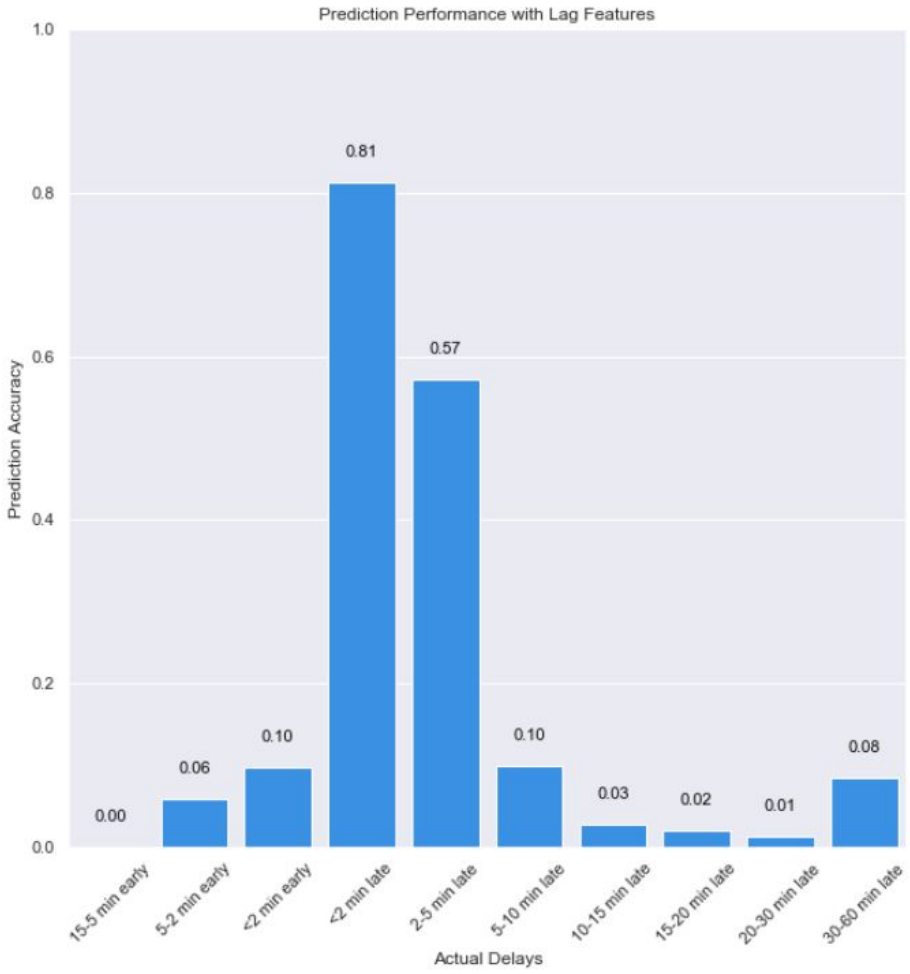
Long Term Model

Once again, a similar strategy was taken to evaluate the long term model. However, without lag features a large emphasis is placed on drastic delays. In essence the question at hand is can a confident recommendation to change a rider's transportation plan be provided solely based on past train performance, local sporting events, and local weather forecast.

The bulk of the prediction error seems to mirror that of the "real-time" model with a slightly larger range for the "long term" model. However, as shown in the below scatterplot, the residuals show a slightly more negative weight skew than the "real-time" model. Without lag features to show otherwise, this could be a result of train operators compensating for potential delays.



Evaluation of the time bucket categorizations reveals a stark answer to our original question - “can a confident recommendation to change a rider’s transportation plan be provided solely based on past train performance, local sporting events, and local weather forecast.” As the below plot depicts, delays ranging from 5 minutes to 60 minutes can be predicted between 1 to 10% of the time, which is not very effective. While the approach to the current input data could be revised, new models selected, and hyperparameters reviewed, predicting large delays one or more days out likely will require richer and more abundant data.



Extreme Gradient Boosting: Model Creation and Evaluation

Since the “real-time” model with the inclusion of lag features returned rather promising results with minor hyperparameter tuning, another iteration was conducted using extreme gradient boosting. With far more hyperparameters available to tune, a smaller random sample (10,000) was used for various cross validation efforts during the tuning process.

Ultimately, max_depth, min_child_weight, gamma, colsample_bytree, subsample, reg_alpha, n_estimators, and learning rate were all iteratively tuned using 3-fold cross validation. The optimal values for each hyperparameter were as follows:

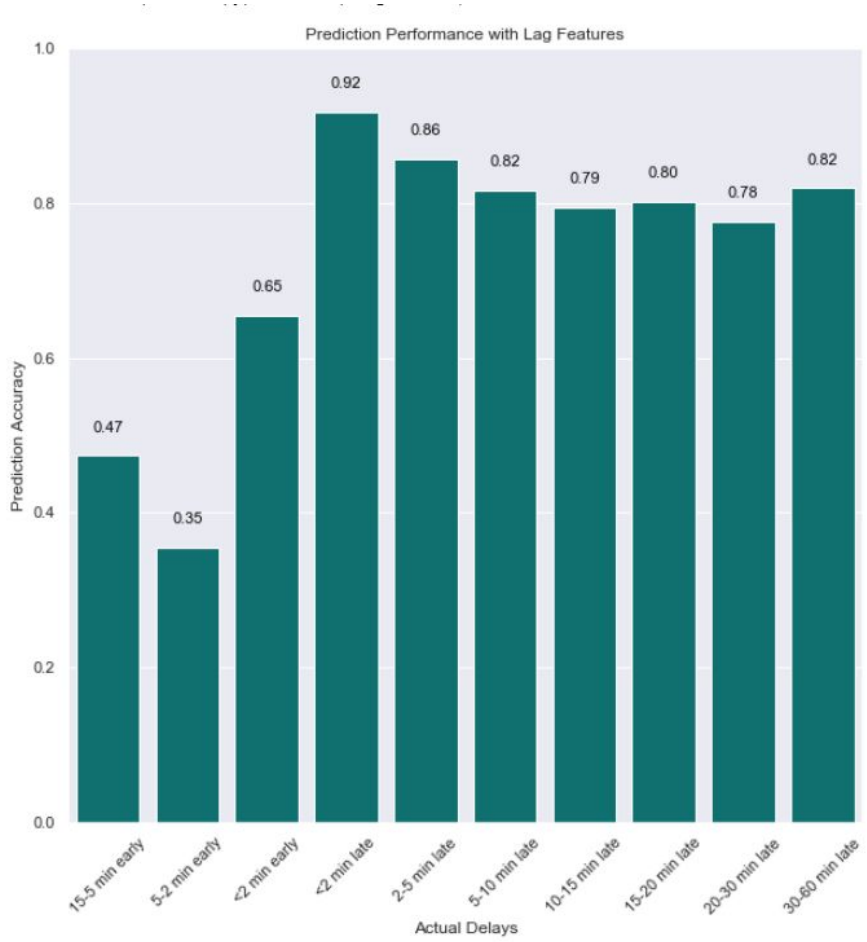
Time Bucket	Seconds Range
max_depth	3
min_child_weight	4
gamma	0.0
colsample_bytree	0.85
subsample	0.85
reg_alpha	1e-05
learning_rate	0.1
n_estimators	1000

As first shown suggested in the linear regression exercise, the regularization constant here is quite small. Tuning indicated a lower number of trees, 550, with a learning rate of 0.1. However, once a larger dataset was introduced, generalization performance was inferior to the random forest model using 1,000 trees. Thus, increasing the extreme gradient boosting model's number of trees back to 1000 rendered results more on par with the random forest model with far faster results.

Extreme Gradient Boosting: Model Performance

Once again plotting predicted against actual shows a thorough line at 45 degrees with a window of variance around 0 to 1000 seconds delayed. Additionally, evaluating performance across the previously defined buckets shows very similar results with very slight improvement and degradations in various delay ranges.

Specifically, comparing the random forest model's ability to correctly predict time ranges compared to that of the extreme gradient boosting shows that random forest performed better in the 30-60 minute and 2-5 minutes early ranges. Other areas show very slight changes of +/- 1% that are likely not very significant.



Opportunities for Improvement

As with any data analysis or modeling effort, there is room for improving both the quality of data and approach taken for this effort. Arguably most important would be the quality of the RTD data obtained. Unfortunately, the light rail data used in this effort was sans a handful of stations and routes for various reasons. The data also did not contain arrival times which could have added another dimension of evaluating timeliness.

Furthermore, data spanning multiple years could aid in an evaluation of seasonality. This is increasingly important as the tail end of the data received coincided with an employee shortage at RTD that was well documented in the local news. Alongside the employee shortage, passenger ridership and train cancellations, which again plagued the local service area, could have added more rich dimensions to the data gathered.

While evaluating the effect of local sporting events was insightful, other highly-attended local events within the service areas would again increase the richness of the analysis. Additionally, understanding extra infrastructure, i.e. bus routes, set in place for large-scale events could have

made for a more robust analysis. Similarly, more comprehensive weather measurements across the stations would have reduced the widespread need of estimations.

While random forest and extreme gradient boosting produced reasonable results with the “real-time” model, one could argue that a more advanced algorithm could better consume time-based dimensions and their effect. Afterall, trains run according to a relatively linear schedule. However, the productionalization aspect would need to be considered before considering more advanced implementations.

Similarly, a variety of alternative features could have been implemented using the time series data. For example, instead of including up to 4 prior delay durations as lag features, the weights of a linear regression model could have been used to assess all lagged delays. Further additional opportunities here include time-based window features, which could be particularly enlightening in the months experiencing employee shortages.

Riders are not tracked in a congruent fashion as trains and rider data is not available to the public. However, having estimated ridership for each train departure would allow one to determine if capacity has an affect on delays. Furthermore, it could help to identify high impact areas based on rider volume.

Including a spatial analysis of routes between stations would help to estimate the impact of the distances and identify possible areas of high variance, such as streetlights or queuing spots.