Will a scheduled ride arrive on-time?

Capstone project submitted for
the Springboard.com
Intermediate Data Science Course
November 2018
Student: Bob Newstadt
Mentor: Hobson Lane

Abstract

Data science models were built to predict, 1 hour in advance of the scheduled start of a trip, when the vehicle would arrive at the pickup location relative to the scheduled start time. Algorithm selection and feature engineering improved the accuracy of the models. Information leakage in building the data models was investigated. The final model explained 22% of the variance in lateness.

Furthermore, the models identified 2 principal causes of predictable lateness. The primary correlate to lateness was the driver; some drivers are more consistently on-time than others. The second case is when a driver gets replaced at the last minute.

Interventions and additional investigations are recommended based on this analysis.

Background

On-time arrival is an important metric for a transportation service to optimize. 25% of negative customer comments mention the word "late" and another 19% mention time-related issues. "Driver" and "Late" are the top 2 most frequently used words in negative comments for this service. Customers value on-time arrival. The business would like to reduce lateness to improve customer satisfaction and reduce support costs.

Rides are scheduled in advance to start at a specific time and location. The driver must check-in 60 to 90 minutes before the scheduled start of the ride. The check-in period is an ideal time to alert the driver about the likelihood of arriving late to the pick up location. The driver is already interacting with the driver app at that time. An informed driver could take actions to improve the probability of arriving on-time.

There are many hypotheses for what is causing lateness: Is lateness an issue for a few drivers? Is lateness mostly correlated with new drivers? Is lateness associated with specific locations which are hard to find or where parking is difficult? Is the cause of lateness traffic congestion at certain times? Building predictive models can expose the factors which have the most influence on lateness.

Problem Statement

Based on data known at the driver check-in time, 60 to 90 minutes before the scheduled start of the ride, when will the vehicle arrive at the pickup location? The prediction target is the number of seconds *relative* to the scheduled start time of the ride. This normalizes the prediction around zero which represents arriving exactly on-time.

Building a regression model to predict a continuous value (such as seconds late) was thought to be better than building a classification model (to predict early, on-time, and late classes). Classes can be created for later business use based on the regression predictions.

Data Set

A transportation service provider, who wishes to remain anonymous, granted permission to use their data for this project. The data set will not be available to the public but the results of this project are public.

The data has over 200K rows with 74 attributes per row. Each row is a trip booked in advance with this service. Trips become complete or canceled. Complete trips delivered one or more passengers from one or more origin locations to one or more destinations. Canceled trips were terminated before any passengers was picked up.

The 74 attributes are described in this google sheet.

TABLE 1: The number of attributes of each type in the data set.

boolean	datetime	duration	factor	quantity	string categories	string category	identifier	identifiers	string identifier
10	11	2	1	12	1	13	18	3	3

There are a variety of attribute types in this dataset. 20 attributes are related to geographic locations. 11 attributes are datetimes. 4 attributes are multi-valued, not scalar.

Steps have been taken to protect the identities and personal information of the trip organizers, drivers, and passengers. Names, addresses, and phone numbers do not appear in this data set. The latitudes, longitudes and distance values have been reduced in precision. Free-form text fields are not included.

Research Procedure

A series of predictive models were built targeting the dependent variable: the lateness of the ride arriving to the pickup location. This was measured as the number of seconds relative to the scheduled start time of the ride. Zero seconds is exactly on-time. 300 seconds would be 5 minutes late. -600 seconds would be 10 minutes early. The outliers have been truncated to +/- 30 minutes.

Fun fact: the average arrival time is 8.5 minutes early.

FIGURE 1: Distribution of relative arrival times.

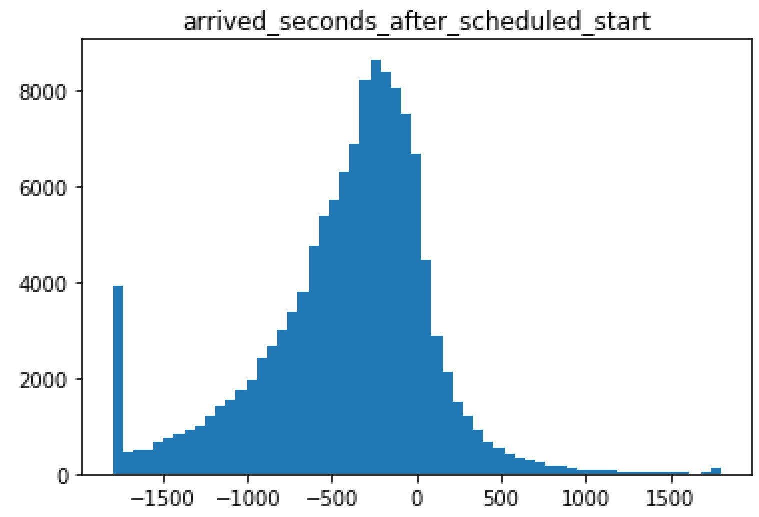


Table 2 is an excerpt of the hyper-parameter tuning table. The full table is available here. Each row is one trial and shows important attributes of the model building process and test results. The hyper-parameter tuning table guides the trial-and-error model optimization process and allows review of what worked well and what did not.

TABLE 2: Hyperparameter Tuning Table (Excerpt)

Trial	Prep Notes	# Features	Model Notes	Predictive Model	Model Tuning	Training Set # obs	Test Set # obs	explained variance score	median absolute error minutes
1	native numeric features only, some big outliers	28	test set = training set	LinearRegression	default	125675	125675	N/A	N/A
2	native numeric features only, some big outliers	28	80/20 train/test	LinearRegression	default	100540	25135	-67.956	35.32
3	native numeric features only, some big outliers	28	normalize=True	LinearRegression	normalize=True	100540	25135	-67.956	35.32
4	added bool features as 1,0 features	34		LinearRegression	normalize=True	100540	25135	-67.985	35.37
5	added 10 duration features measured in seconds	44		LinearRegression	normalize=True	100540	25135	-0.023	5.53
6	added 6*7=42 date properties	86		LinearRegression	normalize=True	100540	25135	-0.027	5.58
7	truncate y outliers	86		LinearRegression	normalize=True	100540	25135	0.045	4.77
8	create one-hot encoded features for some date features.	361		LinearRegression	normalize=True	100540	25135	-7.02432E+22	4.68
9	Reduce one-hot date features.	215		LinearRegression	normalize=True	100540	25135	0.048	4.75

10	Make date attributes integers.	172		LinearRegression	normalize=True	100540	25135	0.047	4.76
11	Misc one-hot features river gender, platform, metro, region.	213		LinearRegression	normalize=True	100540	25135	0.051	4.73
12	add previous ride in 1hr	214		LinearRegression	normalize=True	100540	25135	0.052	4.72
13	same	214		Ridge	alpha = .75	100540	25135	0.052	4.73
14	same	214	try stochastic gradient descent regressor	SGDRegressor	max_iter=10000	100540	25135	-4.3771E+40	9.89676E+20
15	same	214	try random forest regressor	RandomForestRegressor	max_depth=2, random_state=808, n_estimators=100	100540	25135	0.041	4.76
16	same	214	random forest with deeper trees	RandomForestRegressor	max_depth=4, random_state=808, n_estimators=100	100540	25135	0.065	4.67
17	same	214	random forest with deeper trees and more trees	RandomForestRegressor	max_depth=4, random_state=808, n_estimators=200	100540	25135	0.065	4.67
18	same	214	gradient boosting regressor	GradientBoostingRegressor	max_depth=4, random_state=808, n_estimators=200	100540	25135	0.198	4.16
19	added feature avg_prior_arrived_late_seconds	215	gradient boosting regressor	GradientBoostingRegressor	max_depth=4, random_state=808, n_estimators=200	100540	25135	0.259	3.87
20	same	215	test for leakage train on older data and test on newer data	GradientBoostingRegressor	max_depth=4, random_state=808, n_estimators=200	100198	25477	0.18	4.18
21	added feature avg_prior_arrived_late_seconds_to_origin_location	216	continue to use time series test set and test new feature	GradientBoostingRegressor	max_depth=4, random_state=808, n_estimators=200	100198	25477	0.186	4.09
22	same	209	Remove all IDs from the feature set to test the hypothesis that resulting predictive	GradientBoostingRegressor	max_depth=4, random_state=808, n_estimators=200	100198	25477	0.217	3.97

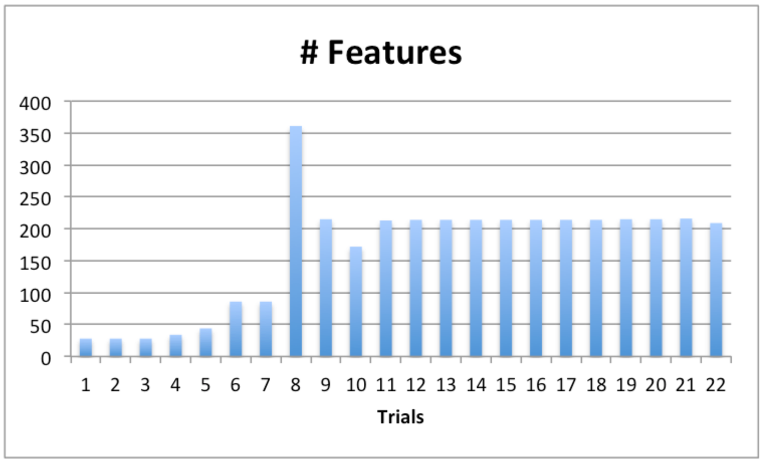
			model will generalize better, perform better on new data.						

Algorithm Selection

A simple linear regression model was built first to establish a baseline for comparison. This algorithm finds optimal weights for a linear model in the feature space. It is fast to train. Ridge regression, a linear model which penalizes model complexity, was built in an attempt to deal with the highly correlated feature set. Stochastic gradient descent regression did not help for this problem. Random forests were trotted out. This ensemble method creates many short decision trees and averages their results. The final models use a gradient-boosting regressor. The gradient-boosting regressor had the best out-of-sample performance. This algorithm builds decision trees focusing more on observations which are not well predicted as it progresses.

Feature Engineering

FIGURE 2: Number of features prepared for each trial



Only numeric features were used initially. Non-numeric features had to be treated in some way to generate numeric features that the machine learning algorithms could use. Different types of features were treated and added in bulk.

In trial #8 the number of features increased by 4X when date attributes were transformed into many binary or “one hot” features. This appeared to overwhelm the machine learning (ML) algorithms with noise. Many of those binary features were given a zero weight by the model. Half of those features were removed in trial #9. During the final trials some specific features were engineered based on features that the ML algorithms were finding helpful.

Model Performance

FIGURE 3: Median Absolute Error (MAE)

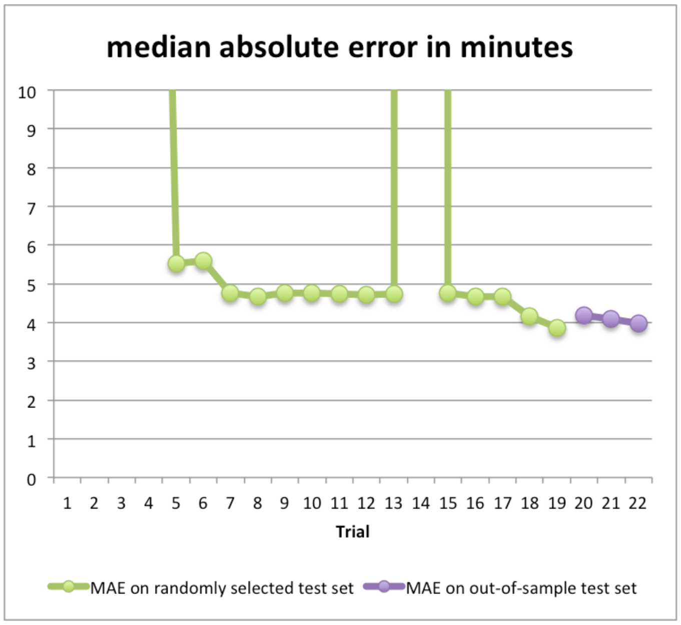
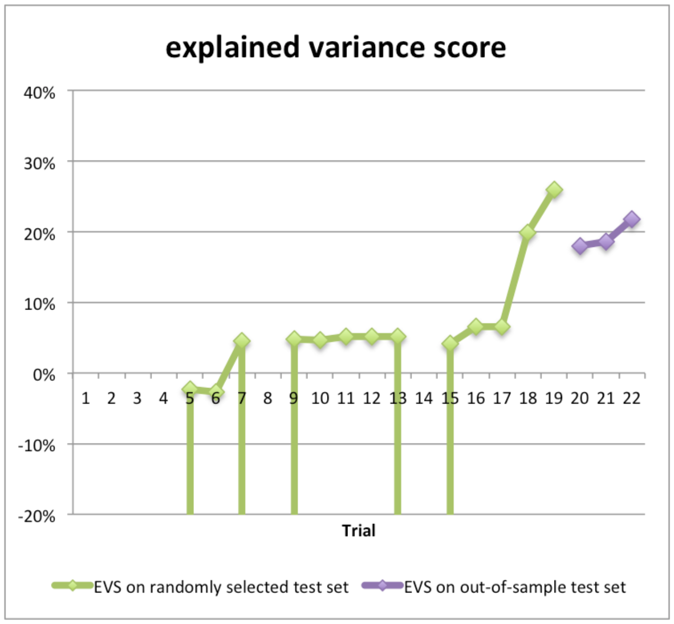


FIGURE 4: Explained Variance Score (EVS)



The metrics selected for performance monitoring were “median absolute error in minutes” (MAE) where lower is better and “explained variance score” (EVS) where higher is better.

The initial linear models had MAE in the 30 minute range. This is off the chart. Terrible. A major improvement happened in trial #5 when duration features were added. MAE was reduced in the final model to below 4 minutes. That means half of the trips arrived within 4 minutes of when they were predicted to arrive.

The “explained variance score” follows an interesting trajectory. The EVS values below 0% were failed trials. The initial trial used all the data as the training set and the test set. Over 30% of the variance was explained by that model. This provides an upper bound, not an independent measure of performance.

Trials #2 to #19 used a randomly selected 20% of the data for testing and the remaining 80% for training. Trial #18 saw significant EVS improvement switching to a gradient boosting regressor algorithm. Trial #19 improved EVS further using a feature engineered to expose the driver’s past on-time performance.

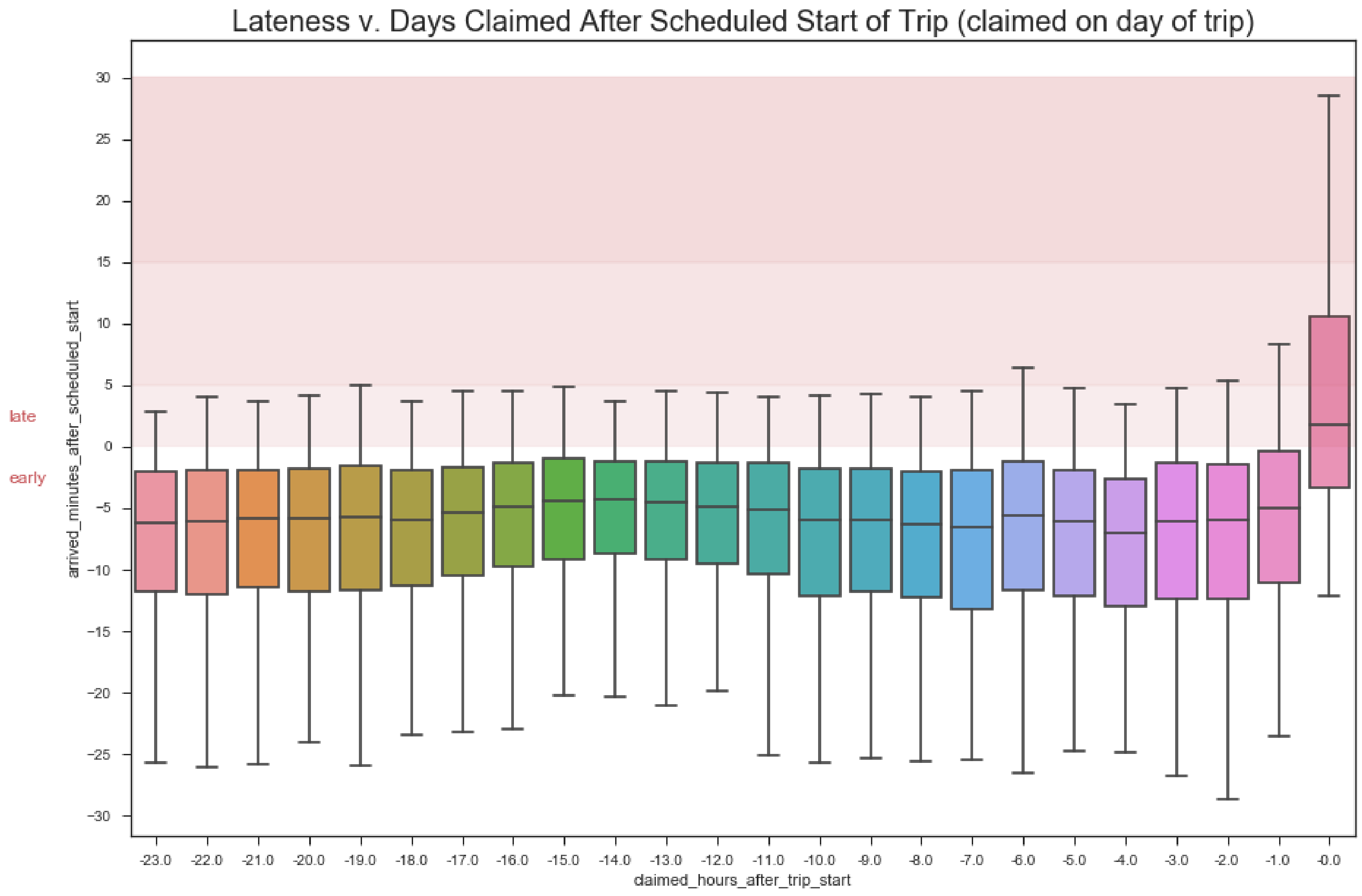
The remaining tests used the latest 20% of the observations as the test set instead of a random test set. The lower performance reflects the added difficulty of predicting on-time arrival in an out-of-sample period. Trial #21 improved EVS using a feature engineered to expose the past on-time performance to the origin location. Trial #22 removed all the ID features, notably driver_id and origin_location_id, forcing the model to use other features which would generalize better. As a result, EVS improved an additional 3 percentage points.

Data Science Learnings

Let the models tell us what is important.

We inverted the standard approach of meticulously investigating each feature. We thought it was inefficient to perform exploratory data analysis (EDA) on all 74 features up front. Instead we built predictive models which selected useful features and suggested additional features. Feature exploration was reserved for features that the models found useful.

FIGURE 5: Trips claimed within 1 hour of scheduled start can arrive extremely late (rightmost box).



Try simple algorithms first to estable a baseline.

A baseline was established using simple linear models which were fast to train and easy to interpret. Other algorithms were used once most of the features had been added to the training set. The final models used gradient boosting decision trees. Some algorithms have several parameters to tune. Better results could have achieved with additional parameter tuning.

Pick 1 or 2 meaningful metrics to optimize.

I was advised to keep a hyper-parameter tuning log but not to optimize all measures at once. Several measures can be used as “guardrails” to observe when the model is failing to perform as intended in certain ways. Studying the tuning log can inform intuition and focus further trials.

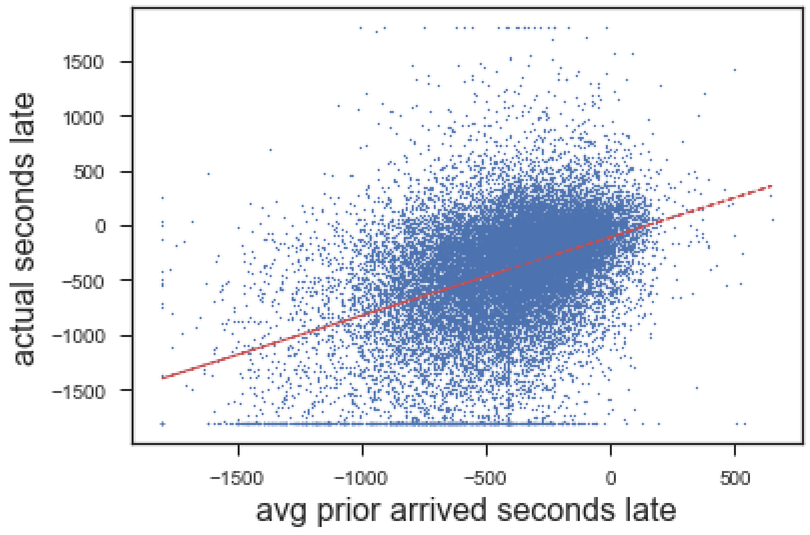
Contextual features helped.

The training set contained features where were measured after the selected prediction time. An example would be the actual arrival time. The model should not peak at those feature values. It would be like having a sports almanac from the future to bet on known game outcomes. The “features from the future” were stripped from the training and test sets during a preparation and cleaning phase to prevent their use.

It took a while to appreciate that “future” features from prior trips could be legitimately used by the model. The “future” features of prior trips were in the past from the perspective of that trip. For example it’s legitimate to use the arrival time of the driver’s previous trip as a feature. Trips are meaningfully ordered in time, unlike say iris measurements, or cat photos in other datasets.

Based on this insight new features were engineered using data from prior observations (trips). In this data set the same drivers and locations appear across multiple trips. Both the average lateness of the 10 prior trips driven by the same driver and the average lateness of the last 10 trips to the same pickup location helped improve the lateness prediction.

FIGURE 5: Positive correlation between driver’s average prior lateness and actual lateness



The randomly selected test set overstated model performance.

A randomly selected test set can be highly correlated to the training set in a way that is not predictive of data in a different time frame. A more robust way to split the data is to select a cutoff time. Use the observations before the cutoff time as the training set and use the observations after that time as the test set. In this project the switch to an out-of-sample test set resulted in lower performance which we feel gives more realistic results.

Driver replacement cases leaked information.

It’s cheating to use any data known after the prediction time to compute the prediction. The expected prediction time, an hour before the ride, was not enforced carefully enough in the case where the driver was replaced within the last hour. The second most important feature used by the model, claimed_before_trip_starts_seconds, is the duration between the claim time and the scheduled start time. Using that claim time prior to when it had happened is the problem.

It would have been more appropriate to use the claim time of the current driver at the time of the prediction if that had been available in the data set. Another alternative is to drop the observations (trips) where the claim time of the final driver was within an hour of the scheduled start.

This blunder highlights that driver replacement is correlated with lateness. Predicting driver replacement was beyond the scope of this project.

The model has a slight selection bias because of last-minute cancellations.

The model to predict lateness is trained on trips where the driver actually arrived. Not all trips arrive. Trips canceled after check-in were excluded from the training set and test sets. The data shows less than 5% of trips get canceled after check-in. This amount of bias seems acceptable.

Some features generalize less well than others.

Random Forest and Gradient Boosting Regressor algorithms yield “importances”, features which are most useful to predict the target variable. The models continued to use driver_id and location_id so heavily that they were in the top 10 most important features.

Removing the IDs from the training set forced the model to use other features which generalized better. Performance measured on the training set went down but performance measured on the test set went up. Figures 5 and 6 show the top 10 most important features before and after the ID features were removed.

FIGURE 5: Trial #21 Model Importances

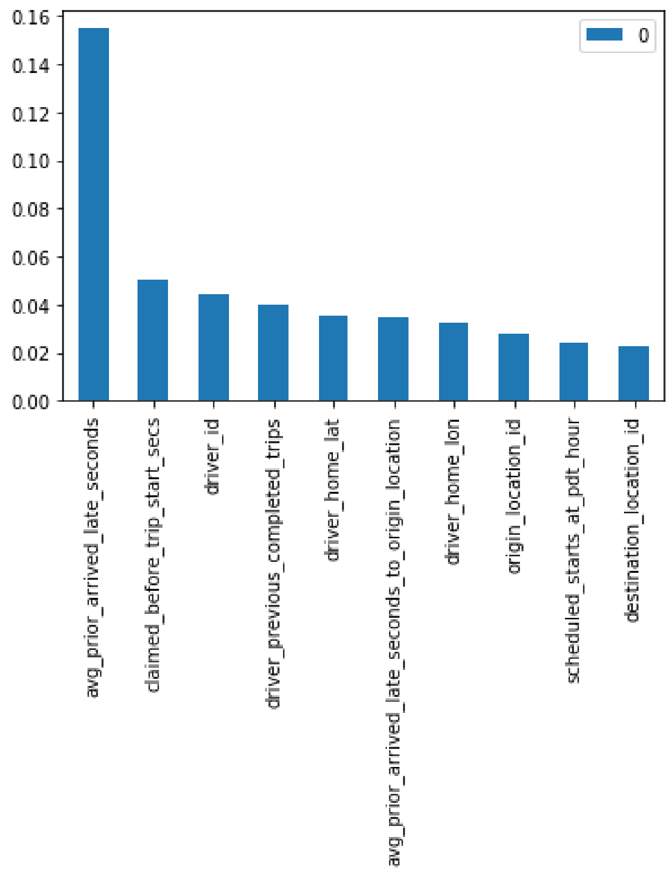
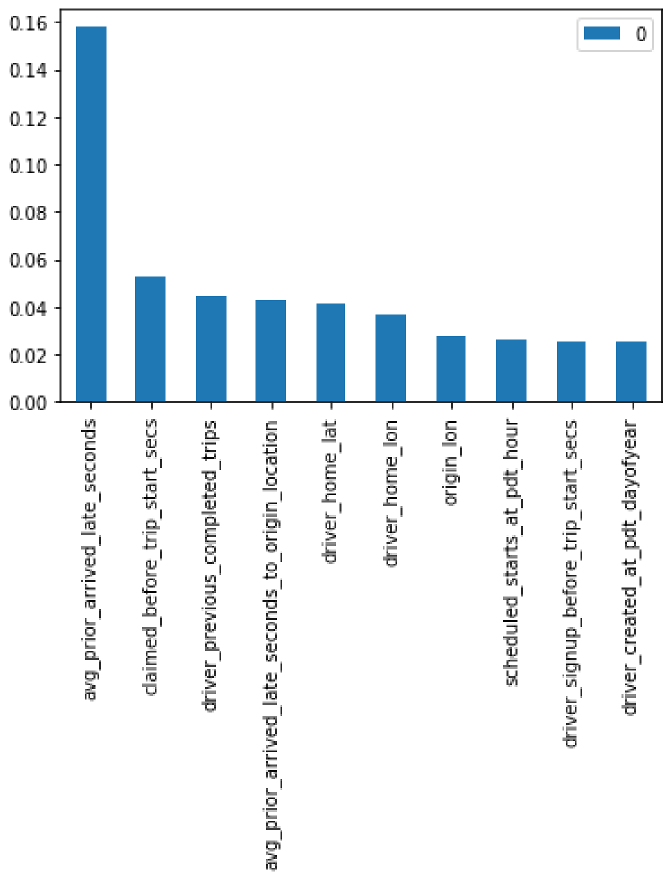


FIGURE 6: Trial #22 Model Importances



Conclusions

Data science models were built to predict, 1 hour in advance of the scheduled start of a trip, when the vehicle would arrive at the pickup location relative to the scheduled start time. The eventual prediction accuracy was low.

Even so, analysis of the features found most useful by the models had several benefits:

1) This analysis identified opportunities to engineer new features. The data set can be considered a time series. The same drivers and locations are present on many trips. This inspired the engineering of contextual features to expose the patterns around specific drivers and locations. The features were computed from the attributes of prior trips.
2) This analysis identified cases which could be addressed in more direct ways. The chronic lateness of individual drivers can be addressed through clear and timely feedback or improved training. The phenomenon of replacement drivers can be further investigated to address its root causes.
3) The analysis revealed information leaks. Leakage happens when data which would not be available at the time of prediction was used to build or test the model. Careful data preparation can deal with that blunder.

Some features will not generalize well to new data such as IDs. In a trial where ID features were removed from the training set the prediction performance improved on the out-of-sample test set.

Business Recommendations

The machine learning models identified 2 principal causes of predictable lateness: the driver and driver replacement.

The Driver

The main factor in lateness is the driver. Many of the most important features reflected the driver's past performance, the driver's experience, and the driver's location.

The most effective way to reduce lateness would be to help drivers manage their schedule. The original hypothesis of this project was to inform the driver at check-in time when to leave. The user experience challenge is that the driver may not depart from the location from which the best time-to-leave is predicted.

Driver training improvements may also help. Help drivers effectively deal with any confusion about the pickup location and understand the business impact of being late.

Driver replacement

The models discovered that drivers who claim rides in the hour before the scheduled start are more likely to be late. Many of these claims are last-minute driver replacements. A process to replace the driver is initiated when an estimated-to-arrive-late alert is fired based on locations and traffic conditions. Sometimes the replacement happens too late for the new driver to get to the pickup location on-time. The recommendation is to investigate the root causes of driver replacement. Help drivers avoid being replaced. Training, feedback, and incentives should be considered.

Acknowledgements

This project would have been a jumble of code without the clarity provided by Springboard mentor Hobson Lane. His guidance was invaluable. I also wish to thank the transportation service company, who wishes to remain anonymous, for providing access to the trip data.

References

Jupyter notebooks in github repository: https://github.com/b-i-bob/trips_springboard_data_science

Hyperparamter tuning log in github repository
https://github.com/b-i-bob/trips_springboard_data_science/blob/master/Hyperparameter%20tuning%20log%20for%20Trips.xlsx

https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

Gail and Zemeckis, "Back 2 the Future Part II" (1989)