Is your NYC Bus on-time?

Thomas Briggs / February 2019

The Data

Source:

- NYC Bus data from this Kaggle kernel: https://www.kaggle.com/stoney71/new-york-city-transport-statistics

Details:

- Trip Data from October 2017
- We will extract the outcome variable from the scheduled time of arrival at stop, expected time of arrival at stop and distance from stop.
- Size: ~7,000,000 Rows

Questions and Objectives

What Constitutes an on-time bus?

- According to the New York City guidelines, a bus is on-time if it is less than 5 minutes late and less than 1 minute early

Who benefits from the resulting information

- Having a predictive model for the NYC Bus system can help optimize the system to perform more effectively then it currently is.

Cleaning the Data

- Removed delays over 60 minutes
- Split all time data into a numerical variable
- Dropped null values (there were a few rows where there was no expected arrival time)
- Extracted Day of Week from date
- When all was said and done, the dataset ends up with about 6 million rows
- Original shape: (6865841, 17); Final Shape: (5715166, 17)

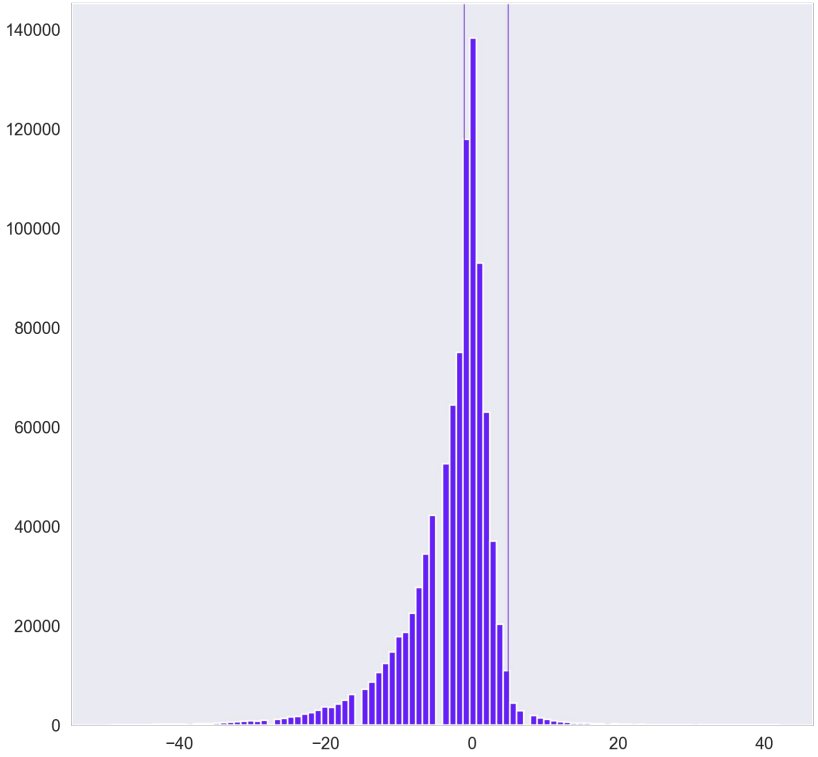
Creating the Outcome Variable

	ScheduledArrivalTime	ExpectedArrivalTime	delay
1	2333.0	4.0	2329.0
2	2351.0	4.0	2347.0
4	2347.0	4.0	2343.0
6	2334.0	4.0	2330.0
9	2353.0	4.0	2349.0
11	2336.0	3.0	2333.0
13	2359.0	4.0	2355.0
14	2319.0	4.0	2315.0
15	2349.0	3.0	2346.0
16	2346.0	3.0	2343.0

- To create the delay variable, I subtracted the scheduled time by the arrival time, which was fairly straightforward.

- As shown to the left, the one hiccup was when a scheduled arrival time was at the end of a day and the expected arrival was very early on the next day. To fix, I had to subtract 2360 from the 2000+ delays

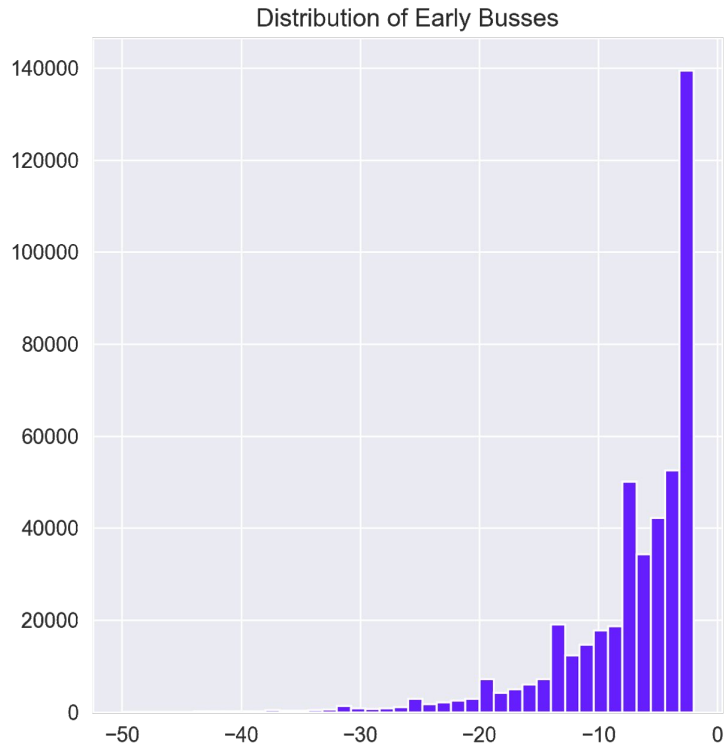
Distribution of Target

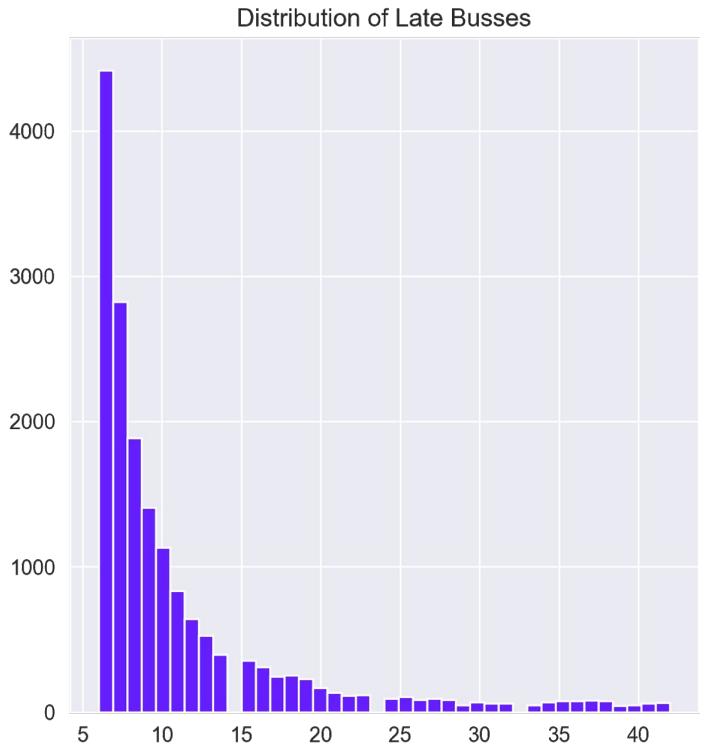


Lines added to show the window within which a bus in on-time.

Data is normally distributed but busses tend to be early at a much larger rate than they are late.

Most commonly, busses are on-time


Distribution of Early Busses
140000
120000
100000
80000
60000
40000
20000
0
−50
−40
−30
−20
−10
0


Distribution of Late Busses
4000
3000
2000
1000
0
5
10
15
20
25
30
35
40

Delays by Borough

	Borough	Count	Average Delay	Standard Deviation
1	Manhattan	197084.0	-3.300679	6.831106
2	Queens	178719.0	-2.989369	6.172698
3	Brooklyn	292638.0	-2.949258	6.369048
3	Bronx	215727.0	-2.700311	6.088328
5	Staten Island	65175.0	-2.277514	5.454414

Manhattan on average is the least on-time by a pretty fair margin compared to the other boroughs. Each have a fairly large amount of variance.

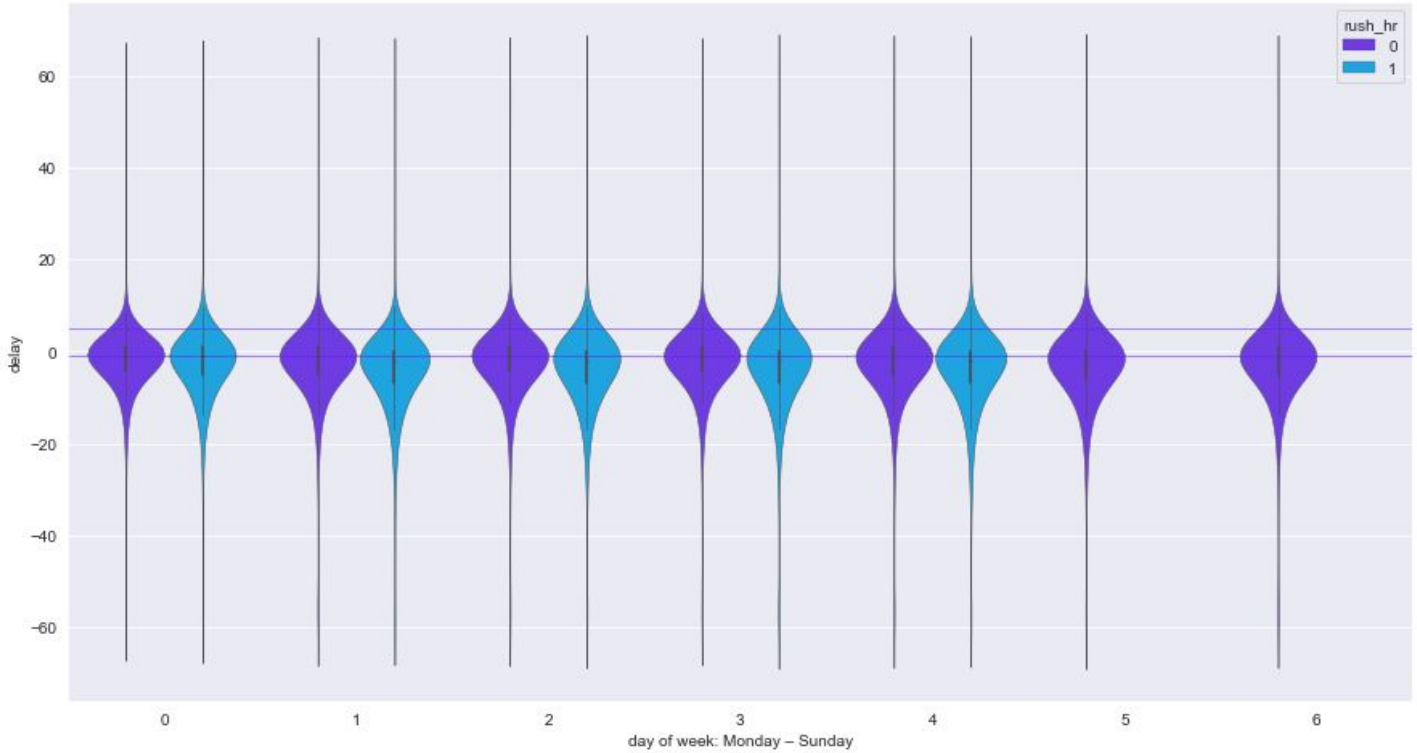
Top 10 most Delayed Bus Lines

	Published Line Name	Average Delay
1	S86	-7.906977
2	S84	-7.296703
3	M1	-5.911177
3	Q56	-5.649912
5	S81	-5.622449
6	B35	-5.528730
7	M5	-5.217507
8	Bx32	-4.975241
9	M2	-4.947130
10	M7	-4.877533

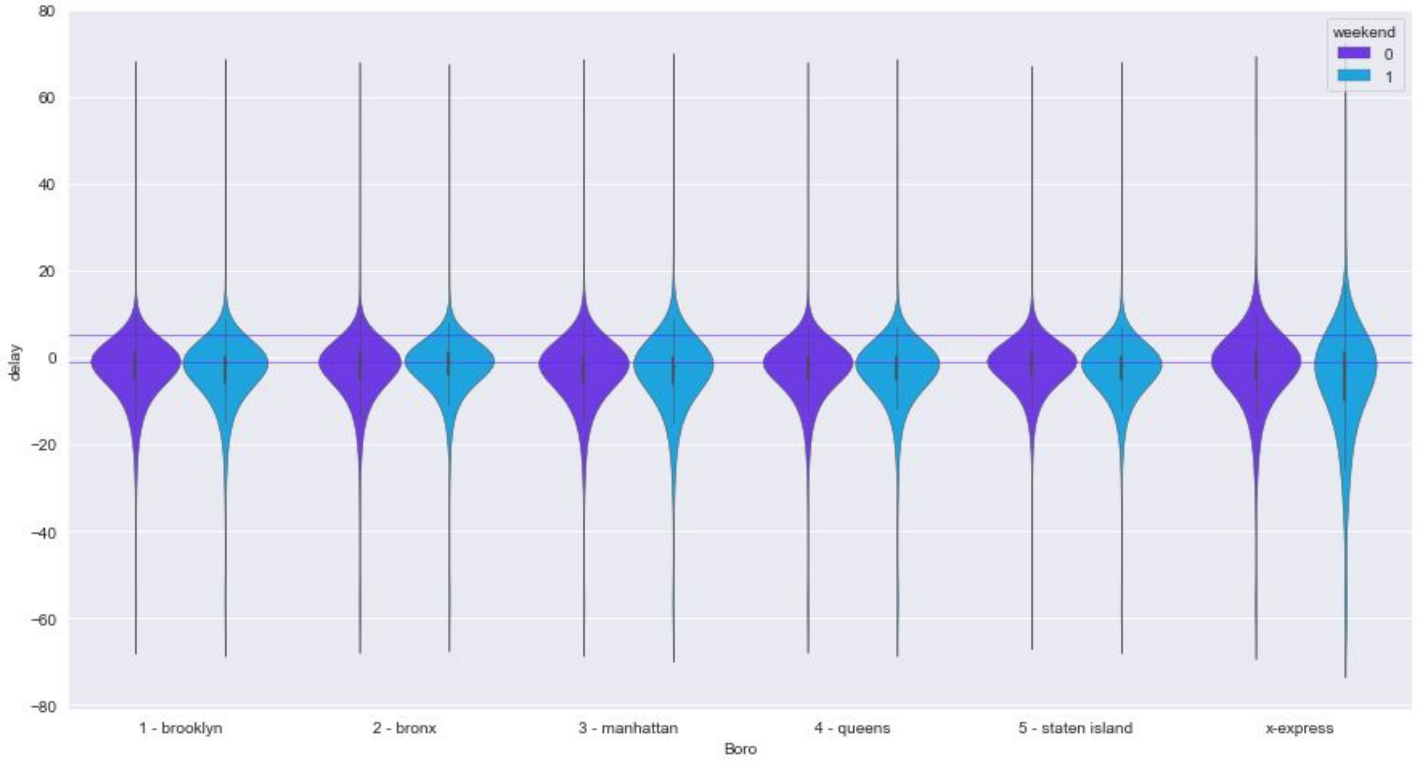
Staten Island Island and Manhattan have the most in the top ten. Interesting that Staten Island has the worst average offenders but the lowest overall average.

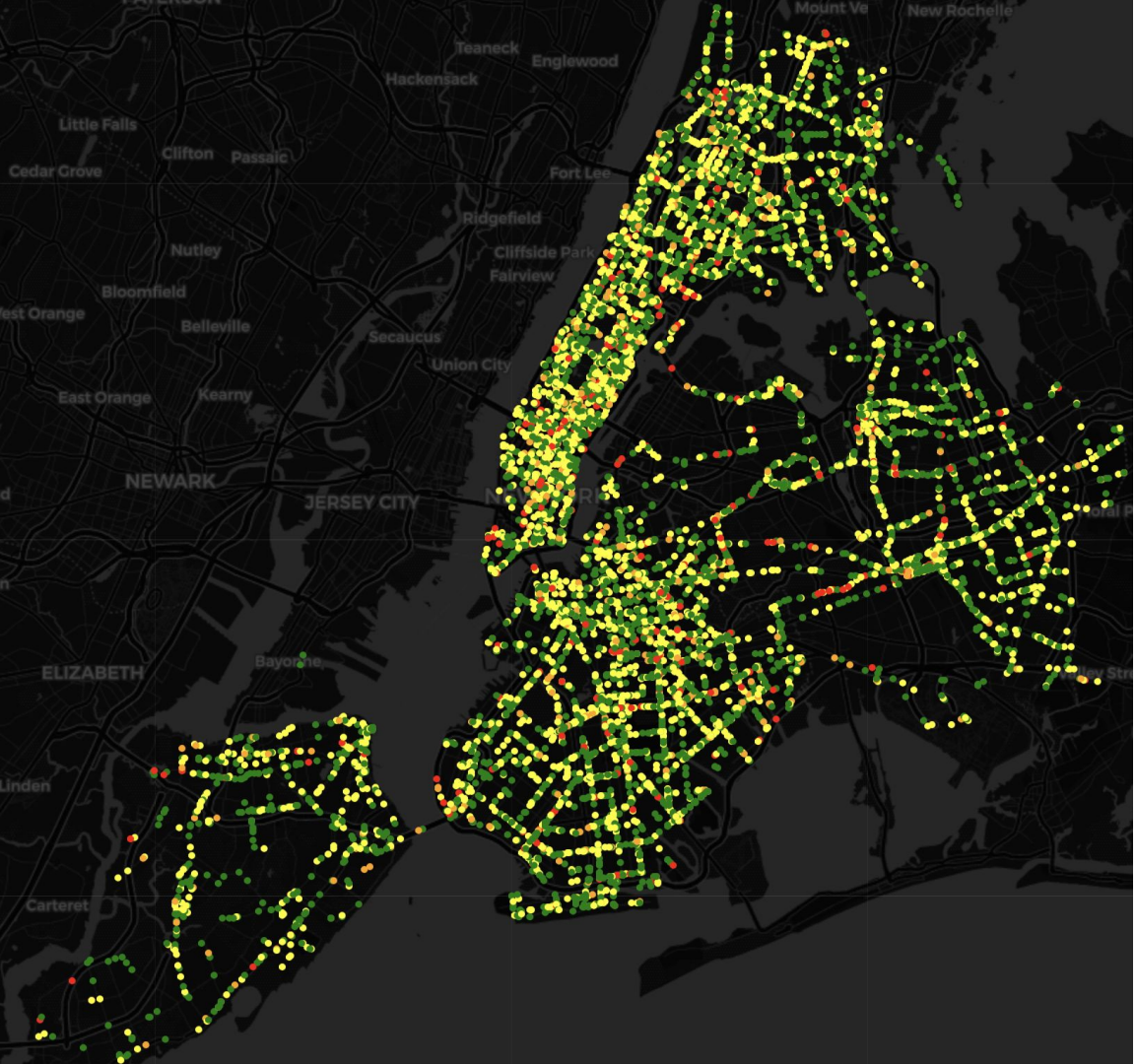
As a feature, I pulled out what would be the historical average delay time for each bus.

Does Rush Hour have any effect?



Does the weekend have any effect?





Vehicle Location & delay

This is a sample of 10,000 busses from throughout the month plotted.

The gap of service in northern queens is interesting.

As the busses get closer to the center of the city, the delays become more frequent.

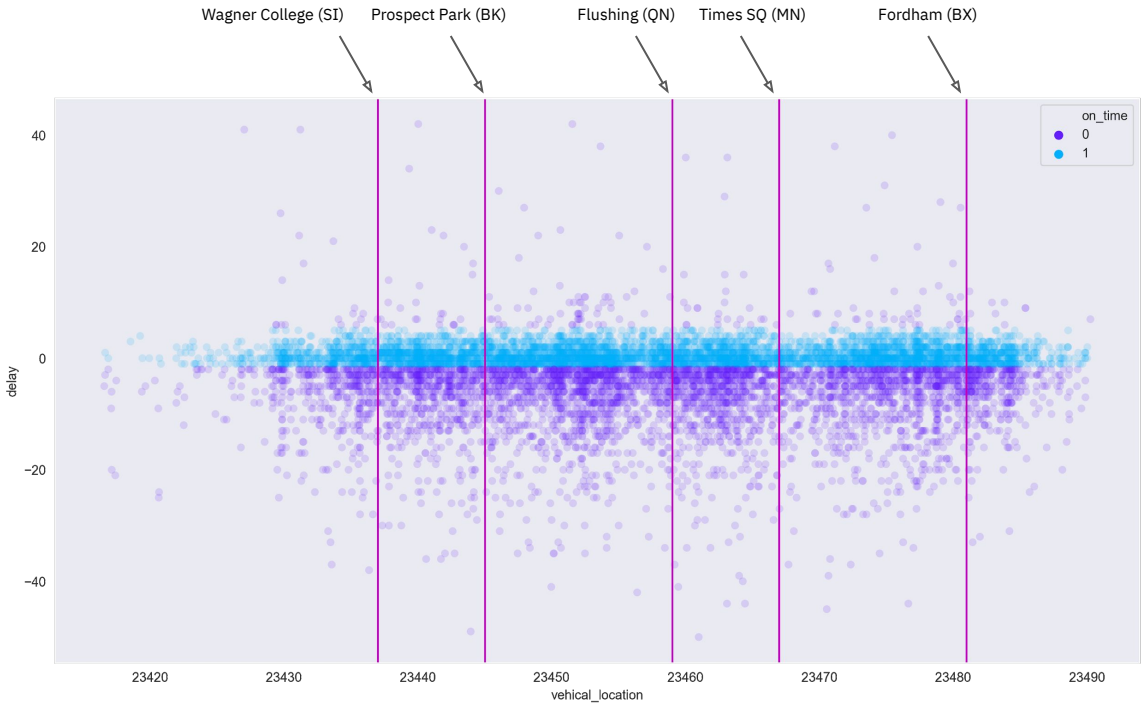
Lower Manhattan and Downtown Brooklyn

Looking closer into the map, it becomes clear that a bus on a main road is more likely to be delayed. (Fulton, Atlantic, Broadway, 14th St, all shown here)

Manhattan & Uptown

Manhattan is pretty congested and again shows that the farther away we get from the center, delays seem to reduce.

Vehicle Location & delay (sample of 10,000)



The map visualizes pretty well just how often a bus isn't on-time, but fails to visualize if they're early or late.

This uses a variable I created that takes Lat/Long and turns it into a single number.

https://stackoverflow.com/questions/4637031/geospatial-indexing-with-redis-sinatra-for-a-facebook-app

Other Features Engineered

- **Line avg delay –** The historical average delay for the line
- **Line delay std –** The historical average standard deviation for the line
- **Veh avg delay –** The historical average delay for the specific vehicle

- **Route length –** Used GeoPy to calculate the distance between the beginning and end of the route
- **Distance Into Route –** Used Geopy to calculate the bus's current location from the start location
- **Percent Into Route –** Used Route length and Distance into route to create a percentage.

Supervised Learning

Models used:

- Logistic Regression
- Naive Bayes
- K Nearest Neighbors
- Decision Tree
- Gradient Boosted Decision Tree (with smaller sample size)

SVM was abandoned due to taking way too long to run.

	index	Attribute	F Score	P Value	Support
0	16	line_avg_delay	16948.231462	0.000000e+00	True
1	25	distance_into_route	14627.996248	0.000000e+00	True
2	17	line_delay_std	13329.779371	0.000000e+00	True
3	20	veh_avg_delay	12857.922009	0.000000e+00	True
4	10	ScheduledArrivalTime	11648.756011	0.000000e+00	True
5	0	RecordedAtTime	7599.470048	0.000000e+00	True
6	18	rush_hr	3911.988626	0.000000e+00	True
7	24	route_length	1640.624392	0.000000e+00	True
8	2	OriginLat	567.421540	2.224031e-125	True
9	23	origin_location	561.402999	4.524595e-124	True
10	9	DistanceFromStop	535.402068	2.035189e-118	True
11	11	day_of_week	373.194212	3.915008e-83	True
12	8	VehicleLocation.Longitude	243.223436	7.919390e-55	True
13	7	VehicleLocation.Latitude	242.592945	1.086738e-54	True
14	21	vehical_location	238.842188	7.140691e-54	True
15	14	sbs	233.487058	1.049989e-52	True
16	5	DestinationLong	192.063365	1.138197e-43	True
17	3	OriginLong	184.311943	5.596737e-42	True
18	1	DirectionRef	178.119876	1.257790e-40	True
19	12	boro_code	156.823587	5.630497e-36	True
20	6	VehicleRef	101.595794	6.827868e-24	True
21	15	bus_code	70.924194	3.717235e-17	True
22	19	weekend	67.304738	2.329075e-16	True
23	26	percent_into_route	25.748085	3.890859e-07	True

Select K Best

Used Select K Best to pull out features with a p-value under .05

Logistic Regression

Parameters: C: 0.5, max_iter: 100, Penalty: l1

	Precision	**Recall**	**F1**	**Support**
Not On-Time	0.60	0.60	0.56	117483
On-Time	0.59	0.63	0.61	119853

Duration: 0:03:21.655056

Naive Bayes

Parameters: NA

	Precision	Recall	F1	Support
Not On-Time	0.58	0.54	0.56	117483
On-Time	0.58	0.61	0.59	119853

Duration: 0:00:03.674113

K Nearest Neighbors

Parameters: n_neighbors=50

	Precision	Recall	F1	Support
Not On-Time	0.62	0.62	0.62	117483
On-Time	0.62	0.63	0.62	119853

Duration: 4:10:54.545705

Decision Tree

Parameters: n_neighbors=50

	Precision	Recall	F1	Support
Not On-Time	0.61	0.58	0.59	117483
On-Time	0.61	0.63	0.62	119853

Duration: 0:00:08.457417

Gradient Boosted Decision Tree (Round 1)

Parameters: loss='deviance', max_depth=8 , n_estimators=50

	Precision	Recall	F1	Support
Not On-Time	0.76	0.68	0.72	117483
On-Time	0.72	0.78	0.75	119853

Duration: 1:10:42.868392

Gradient Boosted Decision Tree Grid Search CV

Because there was some clear promise with the gradient boosted decision tree the first time I ran it through a round of Grid Search CV to see if I could tune the model.

Parameter Grid:

n_estimators: 50,100,150,200

Max_depth: 4,6,8,10

Loss: Deviance, exponential

Gradient Boosted Decision Tree (Round 2)

Parameters: loss='deviance', max_depth=10 , n_estimators=200

Cross Validation: 0.9178593, 0.91602646, 0.9170371, 0.91912978, 0.9191368

	Precision	**Recall**	**F1**	**Support**
Not On-Time	0.96	0.88	0.92	117483
On-Time	0.89	0.96	0.93	119853

Duration: 9:34:20.368787

Conclusions

- It is clear that the gradient boosted decision tree was the best performer among the different models ran.
- This came at a pretty large performance cost.
- The other models performed very poorly, with a bit more tuning, they probably could be improved upon but probably not to the levels which the gradient boosted decision tree was.

Suggestions for Further Study

- Run the models on the entire dataset
- See how the model performs on different months other than October 2017
- Possibly look deeper into creating more location-based features or explorations. Such as a choropleth map based upon zipcode or congressional district.

Thank you!

Any Questions?