SILICON VALLEY DATA SCIENCE

TRAINSPOTTING AND PREDICTING TRAIN DELAYS

DataEDGE 2016

May 5, 2016

Chloe Mawer, Daniel Margala, Jeffrey Yau

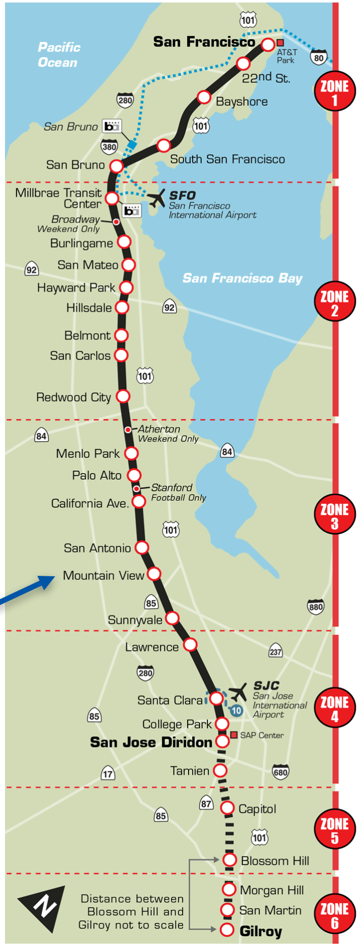
Silicon Valley Data Science is a boutique consulting firm focused on helping companies transform their businesses through data strategy, data engineering, and data science.

OUR PEOPLE

Caltrain

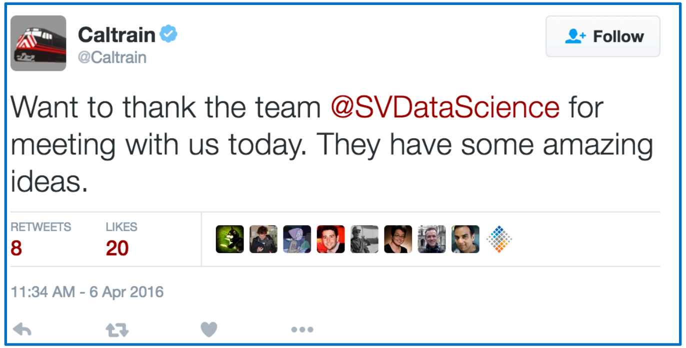
- Commuter rail between San Francisco and San Mateo and Santa Clara counties ~30 stations
- Track crosses streets at several points
- 2015 weekday ridership was 58,245 boardings daily
- On-time performance is about 92%
- No reliable real-time status information
- **Past real-time API outages from days to months**

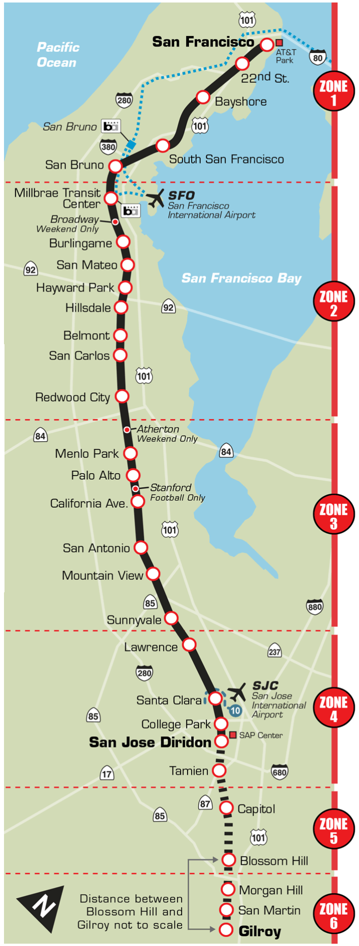
SVDS



- 3 types of train: local, baby bullet, express
- 3 types of schedule: weekday, weekend, “special”
- 1 train per direction per hour off-peak, up to 5 per direction per hour during peak hours
- 92 unique trains per weekday, 36 on weekends

Caltrain Project Background





Engineering and Data Science Aspects of the Project

1. The Data Engineering Aspect
   - Architecture built data ingestion and feeding information back to the data platform and mobile app
2. The Data Science Aspect
   - Classification on Train Video Streams
   - Classification on Train Audio Streams
   - Tweeter Sentiment Analysis and Topic Modeling
   - Analysis of empirical patterns of train delays
3. Modeling Train Delays
   - Departure data sampled at the 1-minute intervals we scraped from Caltrain website
   - Real-time GPS information

Trainspotting

Analyzing video streams to identify trains and their directions

Trainspotting Presentation and Demo

http://cmawer.github.io/trainspotting

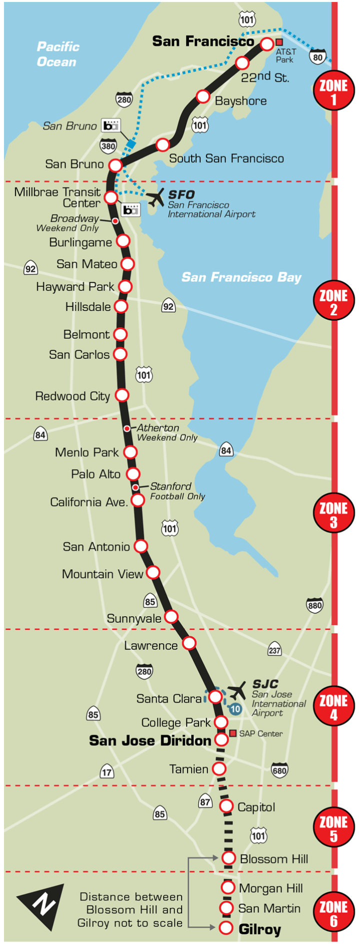
The data science

Train Departure Time Prediction

Caltrain Schedule

Southbound Service

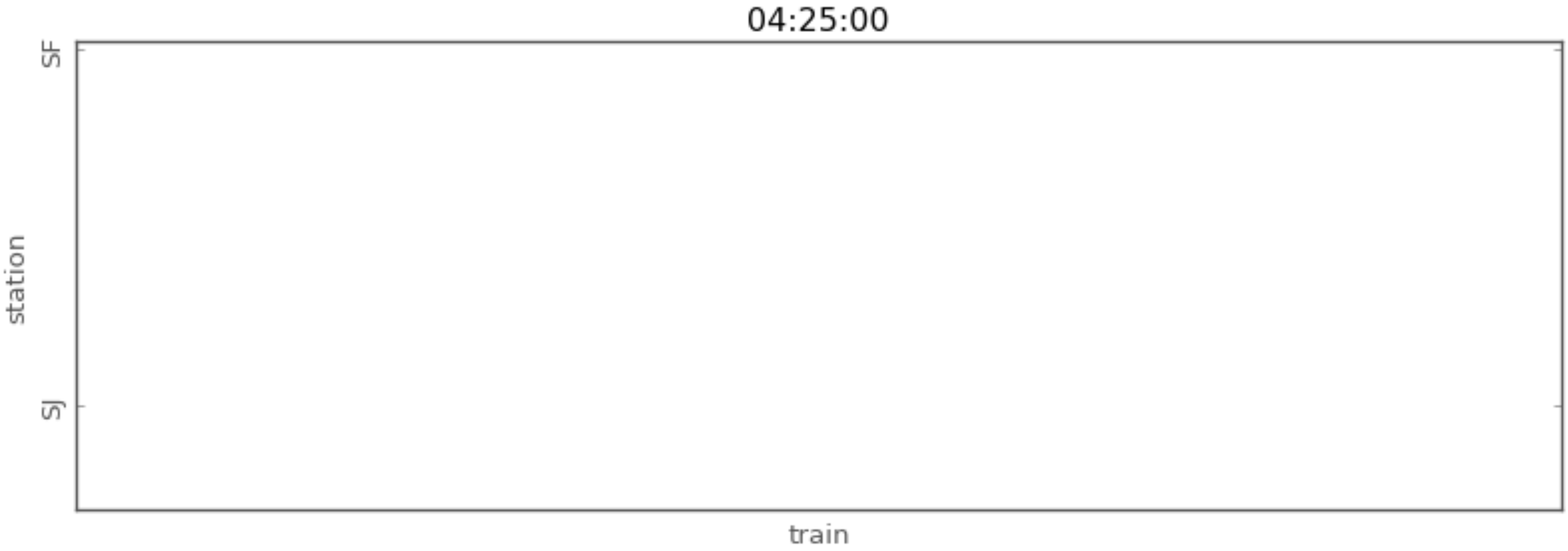
Zone	Southbound Train No.	102	104	206	208	210	312	314	216	218	220	322	324	226	228	230	332	134	236	138	142
1	San Francisco	4:55	5:25	6:11	6:24	6:44	6:57	7:14	7:19	7:24	7:44	7:57	8:14	8:19	8:24	8:44	8:57	9:07	9:37	10:07	11:07
1	22nd Street	5:00	5:30	6:16	6:29	6:49	7:02	7:19	7:24	7:29	7:49	8:02	8:19	8:24	8:29	8:49	9:02	9:12	—	10:12	11:12
1	Bayshore	5:05	5:35	—	6:34	—	—	—	—	7:34	—	—	—	—	8:34	—	—	9:17	—	10:17	11:17
1	So. San Francisco	5:11	5:41	—	6:40	—	—	—	—	7:40	—	—	—	—	8:40	—	—	9:23	—	10:23	11:23
1	San Bruno	5:15	5:45	—	6:44	—	—	—	7:35	7:44	—	—	—	8:35	8:44	—	—	9:27	9:51	10:27	11:27
2	Millbrae	5:19	5:49	6:29	6:48	7:01	7:15	7:32	—	7:48	8:01	8:15	8:32	—	8:48	9:01	9:15	9:31	9:55	10:31	11:31
2	Burlingame	5:23	5:53	6:33	6:52	—	—	—	7:40	7:52	—	—	—	8:40	8:52	—	—	9:35	9:59	10:35	11:35
2	San Mateo	5:28	5:58	6:38	6:57	7:09	—	—	7:46	7:57	8:09	—	—	8:46	8:57	9:09	—	9:40	10:04	10:40	11:40
2	Hayward Park	5:31	6:01	—	7:00	—	—	—	—	8:00	—	—	—	—	9:00	—	—	9:43	—	10:43	11:43
2	Hillsdale	5:34	6:04	6:42	7:03	—	—	7:42	7:50	8:03	—	—	8:42	8:50	9:03	—	—	9:46	10:08	10:46	11:46
2	Belmont	5:37	6:07	—	7:06	—	—	—	—	8:06	—	—	—	—	9:06	—	—	9:49	10:11	10:49	11:49
2	San Carlos	5:40	6:10	6:46	7:09	7:15	—	—	7:54	8:09	8:15	—	—	8:54	9:09	9:15	—	9:52	10:14	10:52	11:52
2	Redwood City	5:45	6:15	6:51	7:14	7:20	7:30	—	—	8:14	8:20	8:30	—	—	9:14	9:20	9:30	9:57	10:19	10:57	11:57
3	Menlo Park	5:50	6:20	6:56	—	7:25	7:35	—	8:02	—	8:25	8:35	—	9:02	—	9:25	9:35	10:02	10:24	11:02	**12:02**
3	Palo Alto	5:53	6:23	6:59	7:20	7:28	7:38	7:53	8:05	8:20	8:28	8:38	8:53	9:05	9:20	9:28	9:38	10:05	10:27	11:05	**12:05**
3	California Ave	5:57	6:27	7:03	—	7:32	—	—	—	—	8:32	—	—	—	—	9:32	—	10:09	10:31	11:09	**12:09**
3	San Antonio	6:01	6:31	—	—	7:36	—	—	—	—	8:36	—	—	—	—	9:36	—	10:13	10:35	11:13	**12:13**
3	Mountain View	6:05	6:35	7:09	—	7:40	7:46	8:00	8:13	—	8:40	8:46	9:00	9:13	—	9:40	9:46	10:17	10:39	11:17	**12:17**
3	Sunnyvale	6:10	6:40	—	—	7:45	—	—	—	—	8:45	—	—	—	—	9:45	—	10:22	10:44	11:22	**12:22**
4	Lawrence	6:14	6:44	7:14	—	7:51+	—	—	8:20	—	8:51+	—	—	9:20	—	9:51+	—	10:26	10:48	11:26	**12:26**
4	Santa Clara	6:19	6:49	—	7:36	7:58+	—	—	—	8:36	8:58+	—	—	—	9:36	9:58+	—	10:31	10:53	11:31	**12:31**
4	College Park	—	—	—	—	8:01+	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
4	San Jose Diridon	6:28	6:58	7:26	7:45	8:08	8:00	8:15	8:32	8:45	9:07	9:00	9:15	9:32	9:45	10:07	10:00	10:40	11:02	11:40	**12:40**
4	Tamien	—	7:05	—	7:52	8:15	—	—	—	8:52	9:14	—	—	—	9:52	10:14	—	—	11:09	—	—
5	Capitol																				
5	Blossom Hill																				
6	Morgan Hill																				
6	San Martin																				
6	Gilroy																				



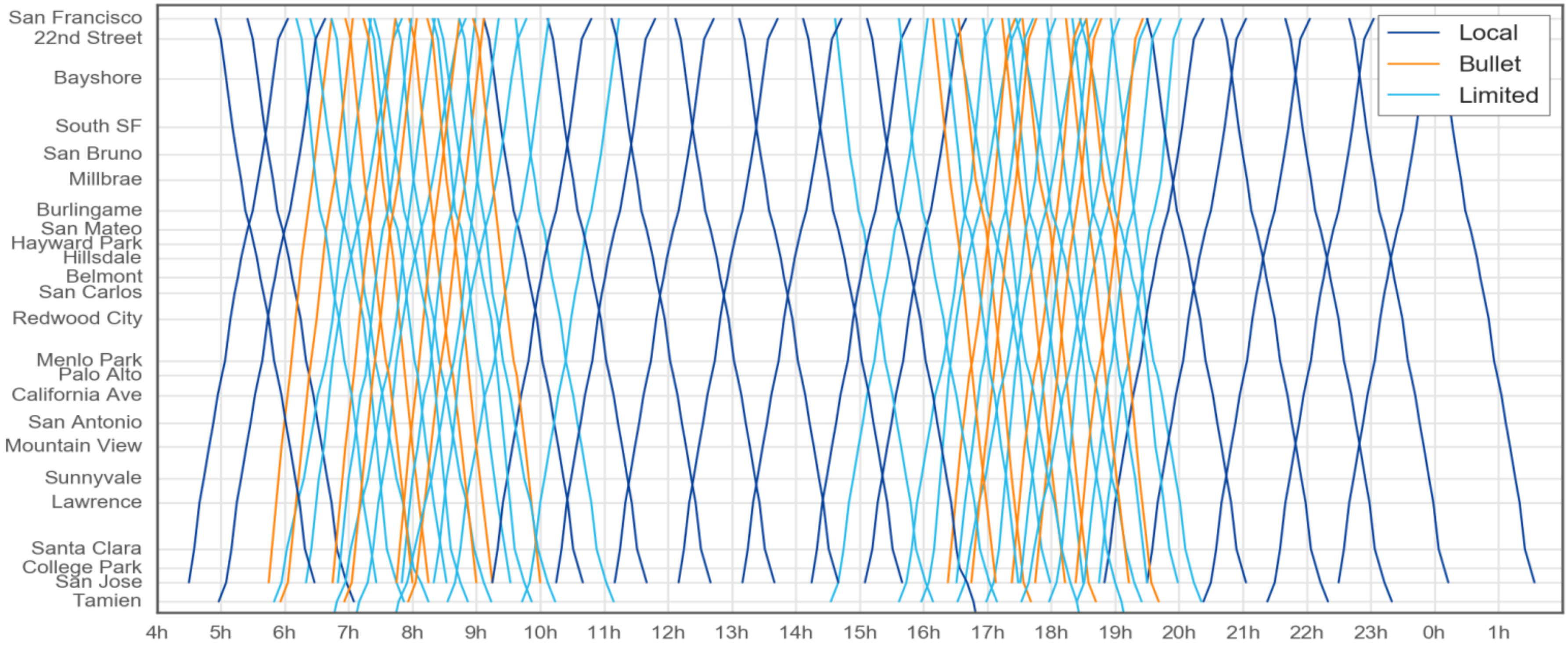
Caltrain Schedule Zoom-In

Zone	Southbound Train No.	102	104	206	208	210	312	314	216
1	San Francisco	4:55	5:25	6:06	6:24	6:44	6:56	7:12	7:19
1	22nd Street	5:00	5:30	6:11	6:29	6:50	7:02	7:18	7:25
1	Bayshore	5:05	5:35	—	6:35	—	—	—	—
1	So. San Francisco	5:11	5:41	—	6:41	—	—	—	—
1	San Bruno	5:15	5:45	—	6:44	—	—	—	7:37
2	Millbrae	5:19	5:49	6:24	6:49	7:02	7:17	7:32	—
2	Burlingame	5:23	5:53	6:28	6:53		—	—	7:44
2	San Mateo	5:28	5:58	6:32	6:56	7:09	—	—	7:48
2	Hayward Park	5:31	6:01	—	7:00	—	—	—	—
2	Hillsdale	5:34	6:04	6:36	7:04	—	—	7:42	7:52
2	Belmont	5:37	6:07		7:07	—	—	—	—

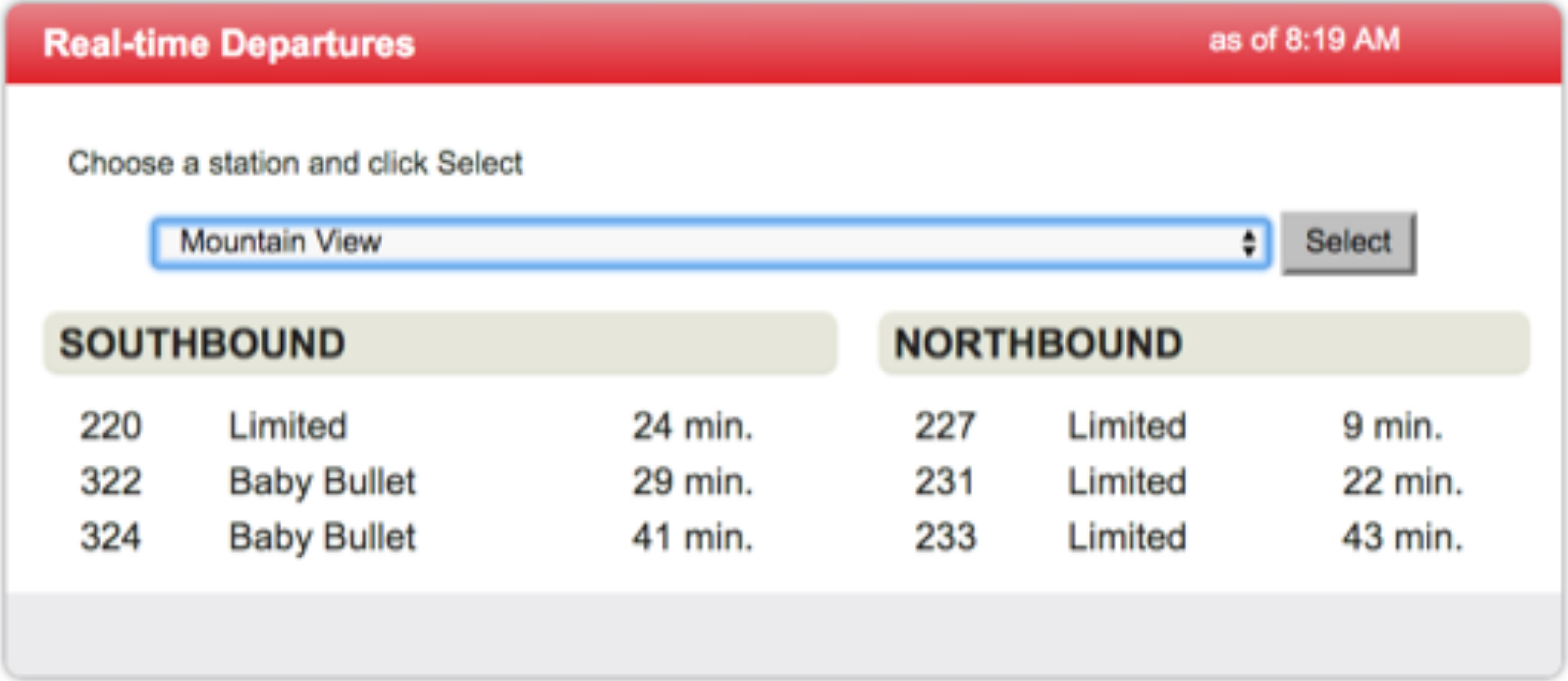
Caltrain Schedule: Animation



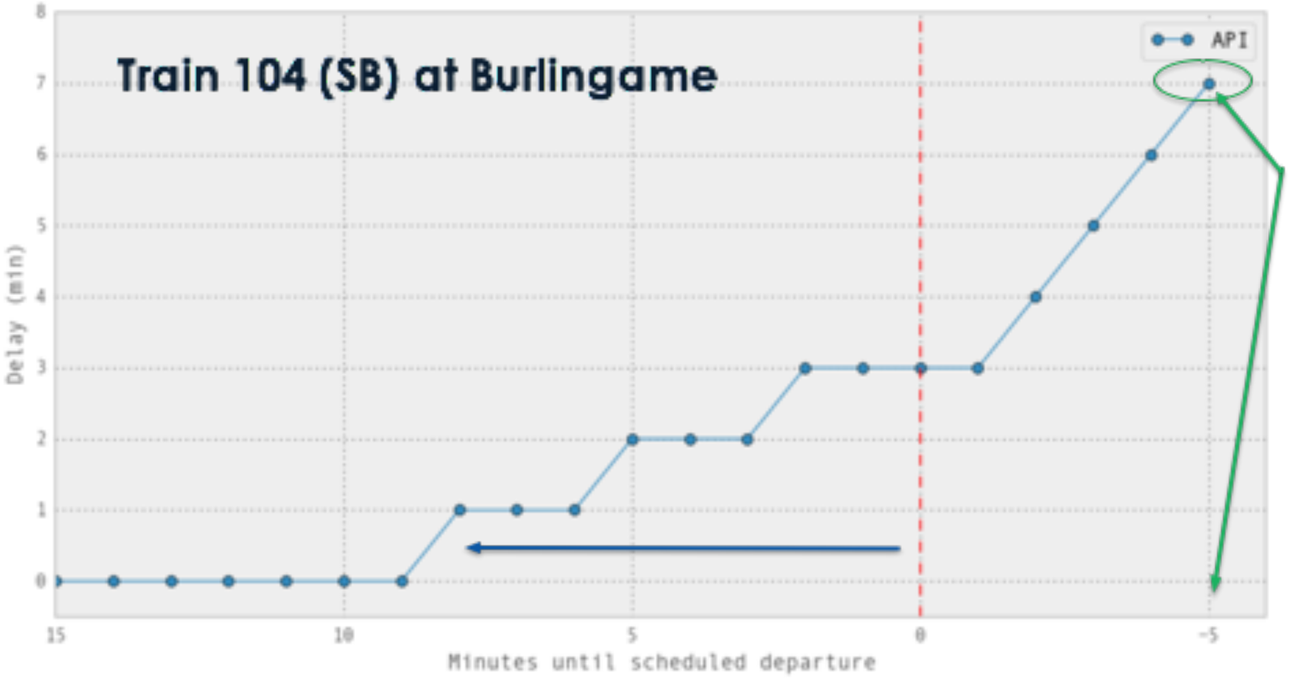
Caltrain Schedule: Marey Graph



Observations: Real Time “API”

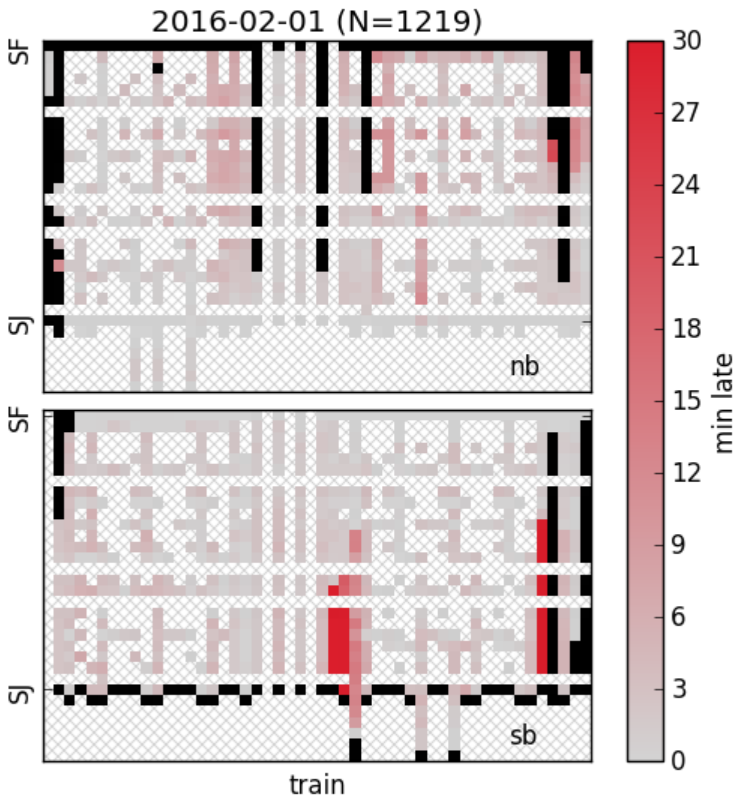


Observations: Caltrain's Real Time "API"

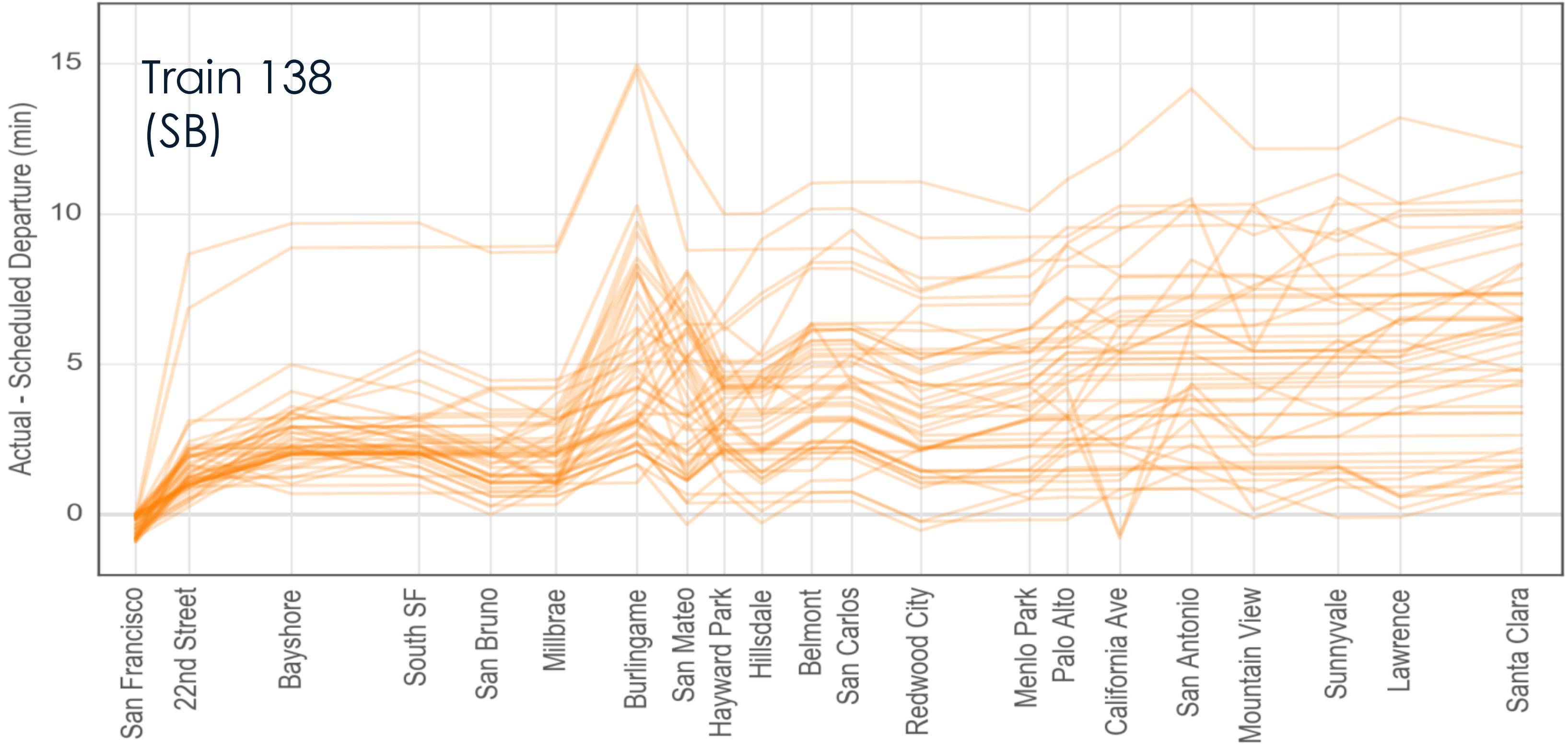


Observations: Dataset Overview

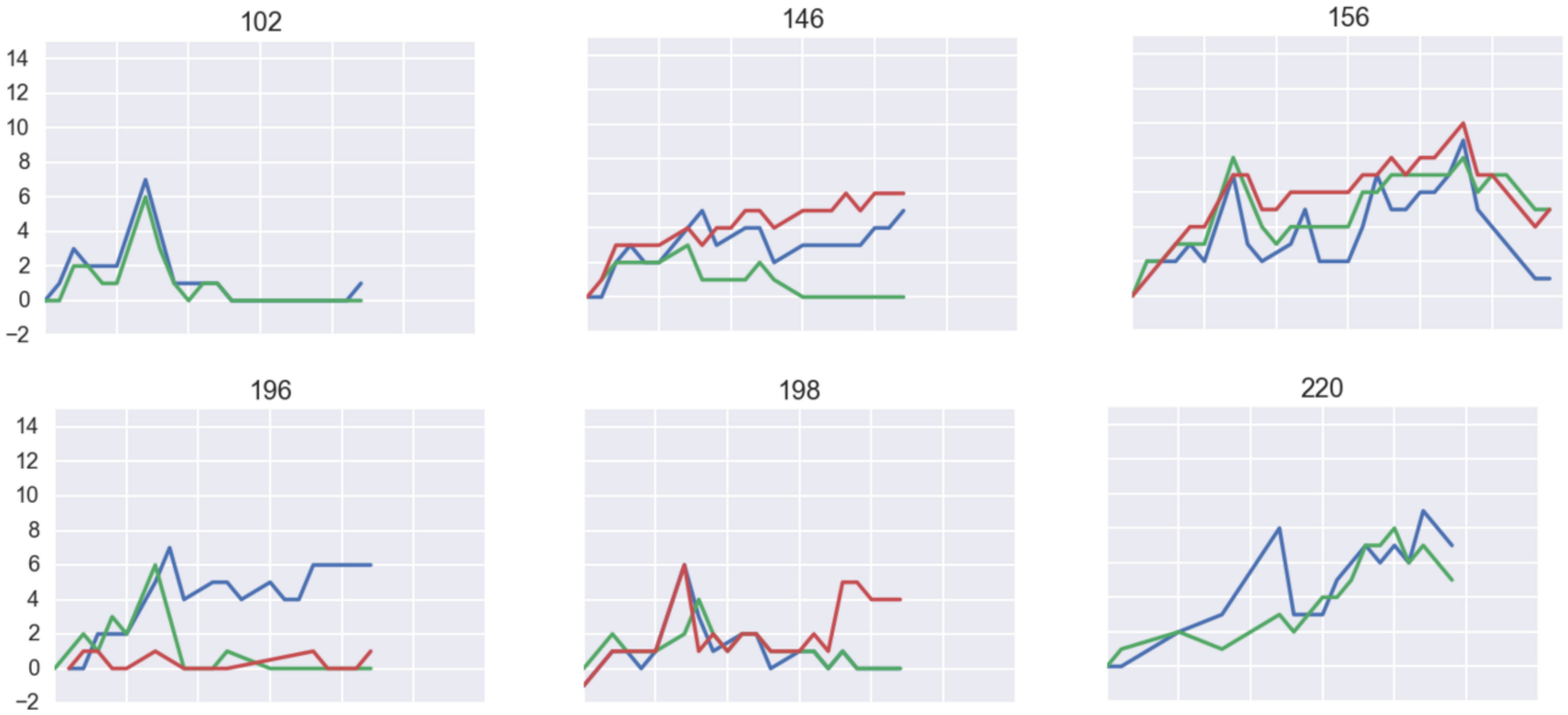
- ~80 days of observation (including weekends/holiday)
- Often missing >10% per day
- Departures are mostly "on time" (within 3 minutes)
- When significant delays occur, they often appear suddenly



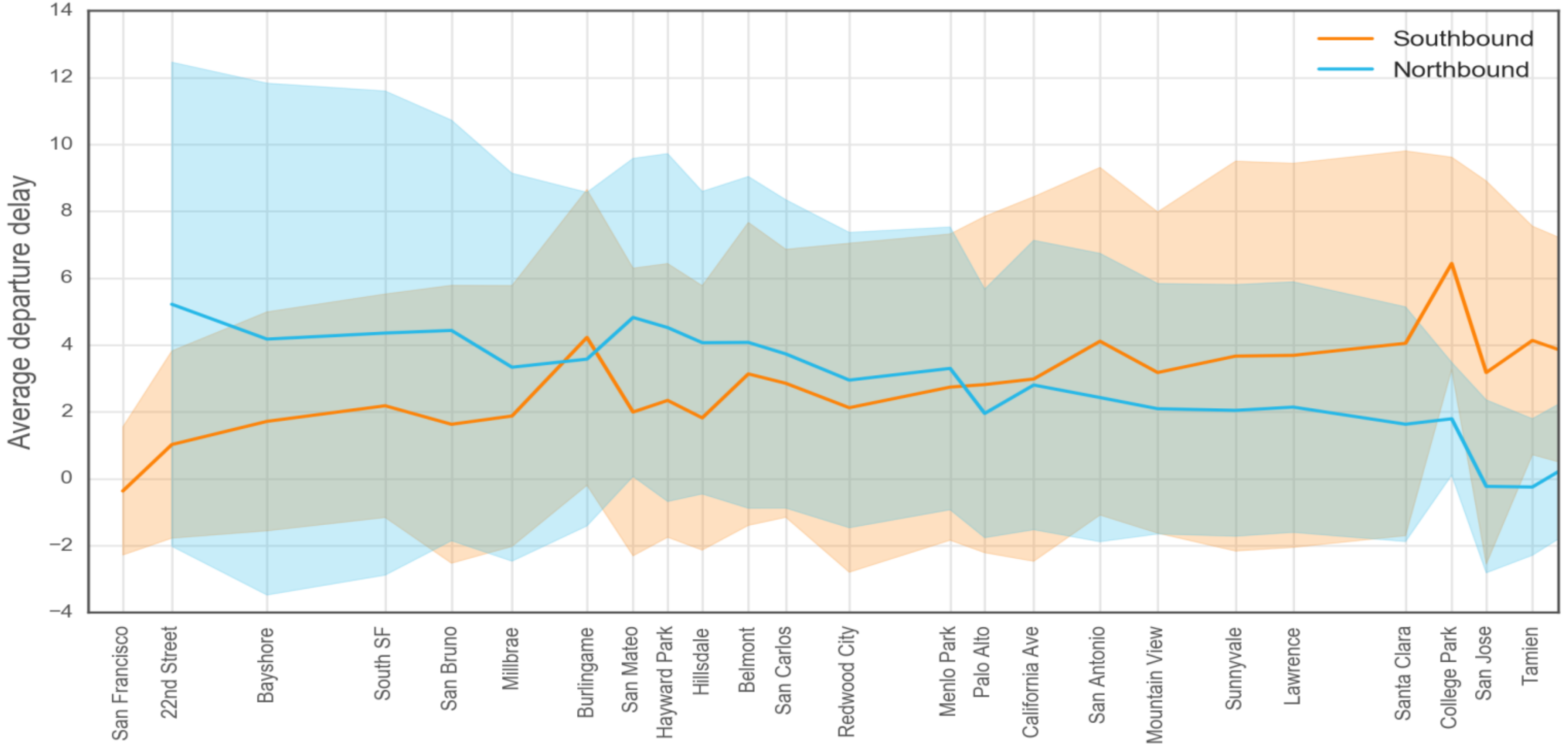
Observations: Single Train Delay History



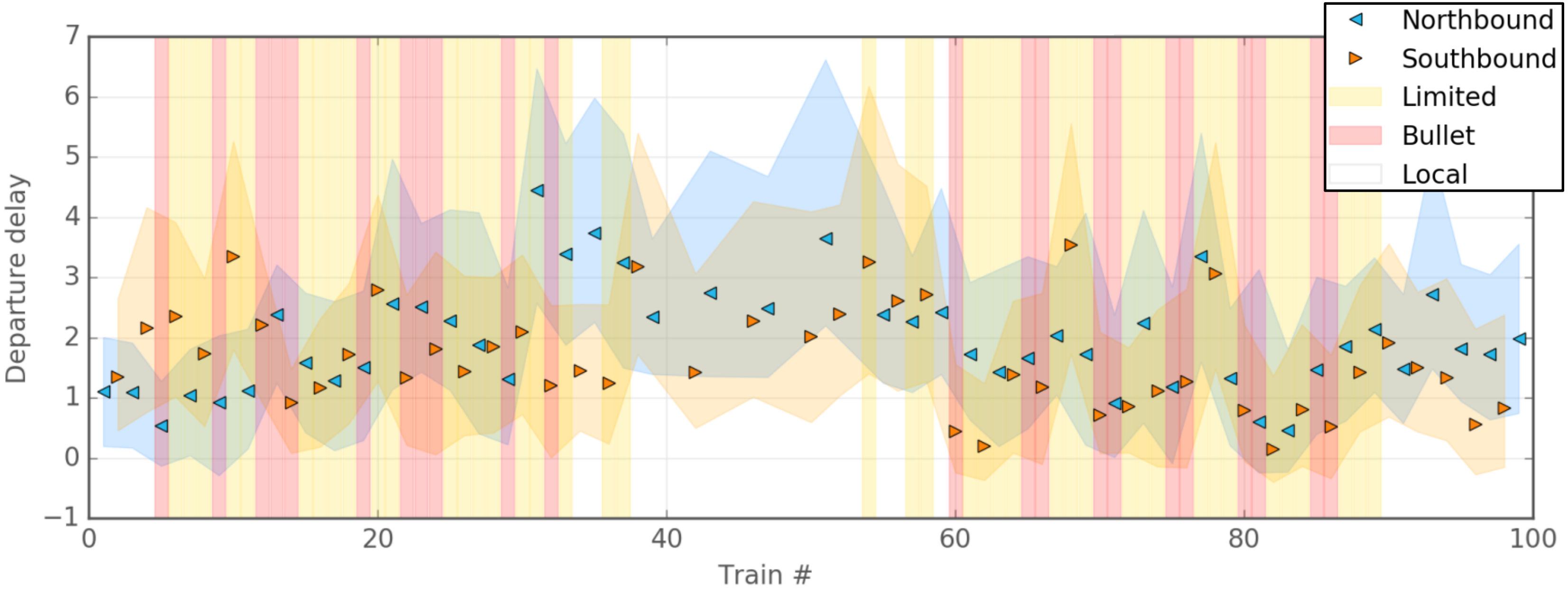
Observations: Delays vary greatly by trains



Observations: Delays Vary Greatly by Station

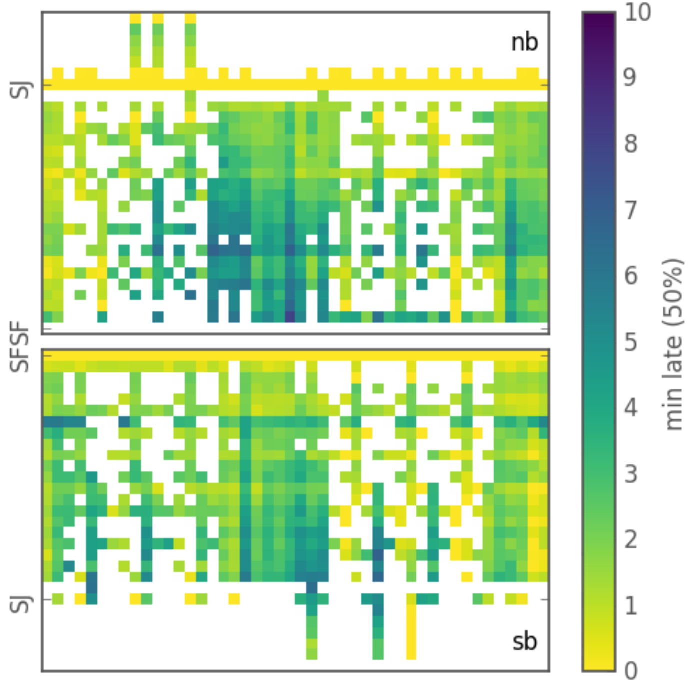


Observations: Median Departure Delays Vary by Train (Type, Direction)

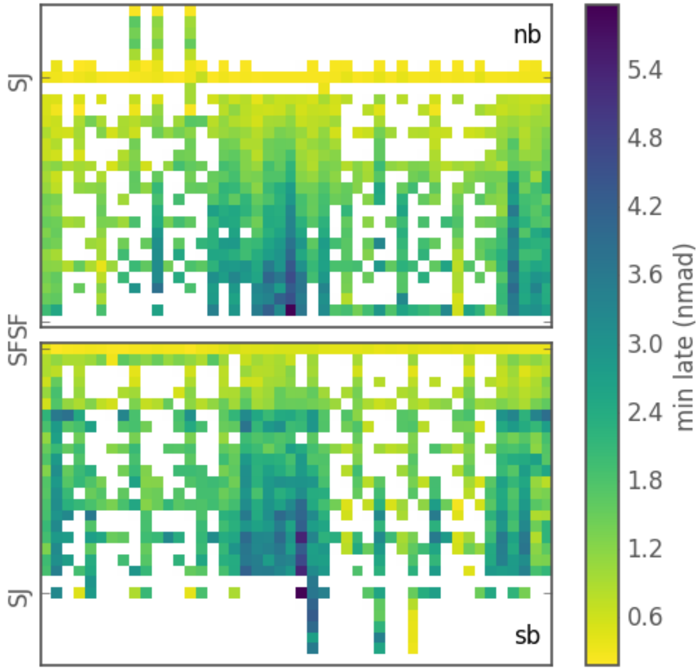


Observations: Dataset Summary (1)

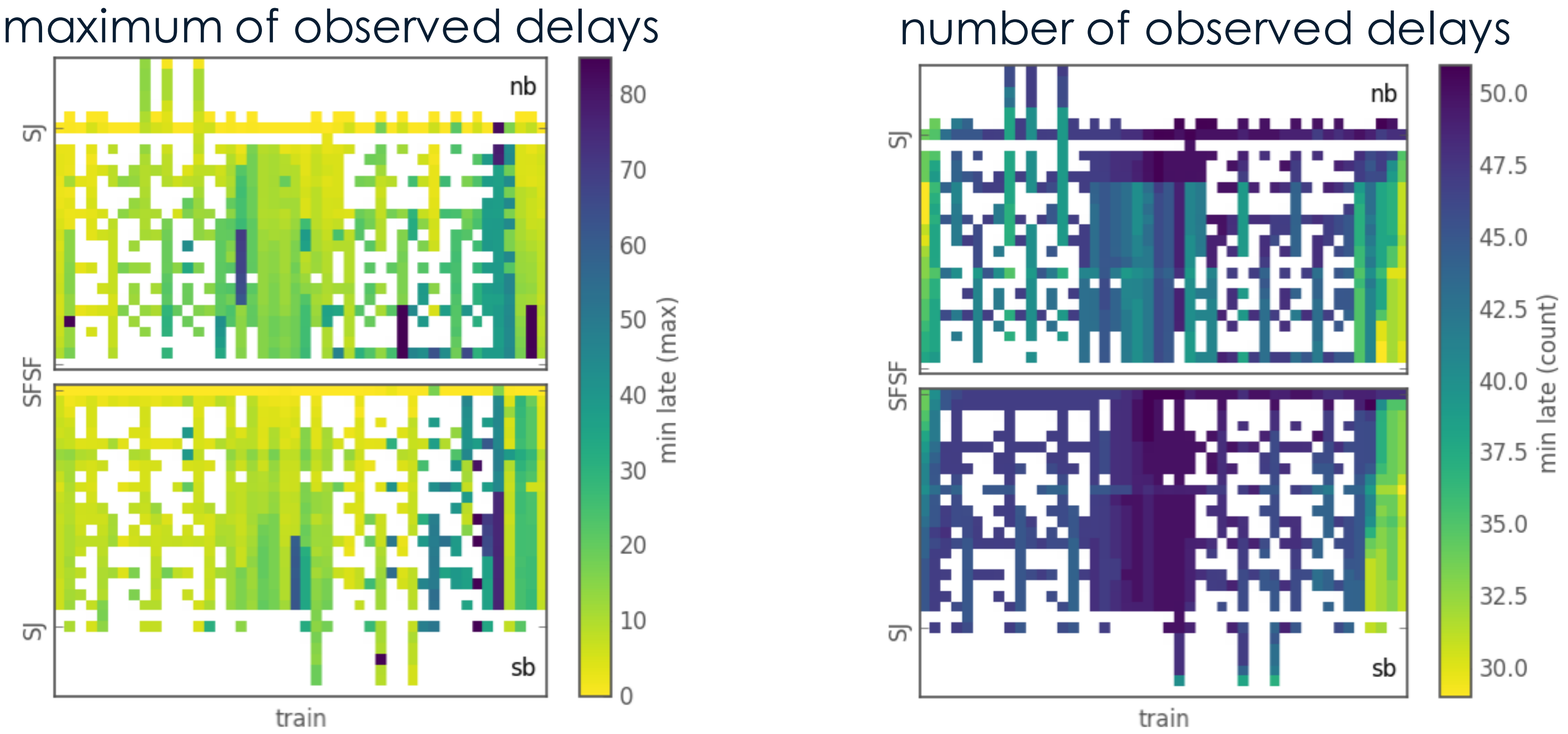
median of observed delays



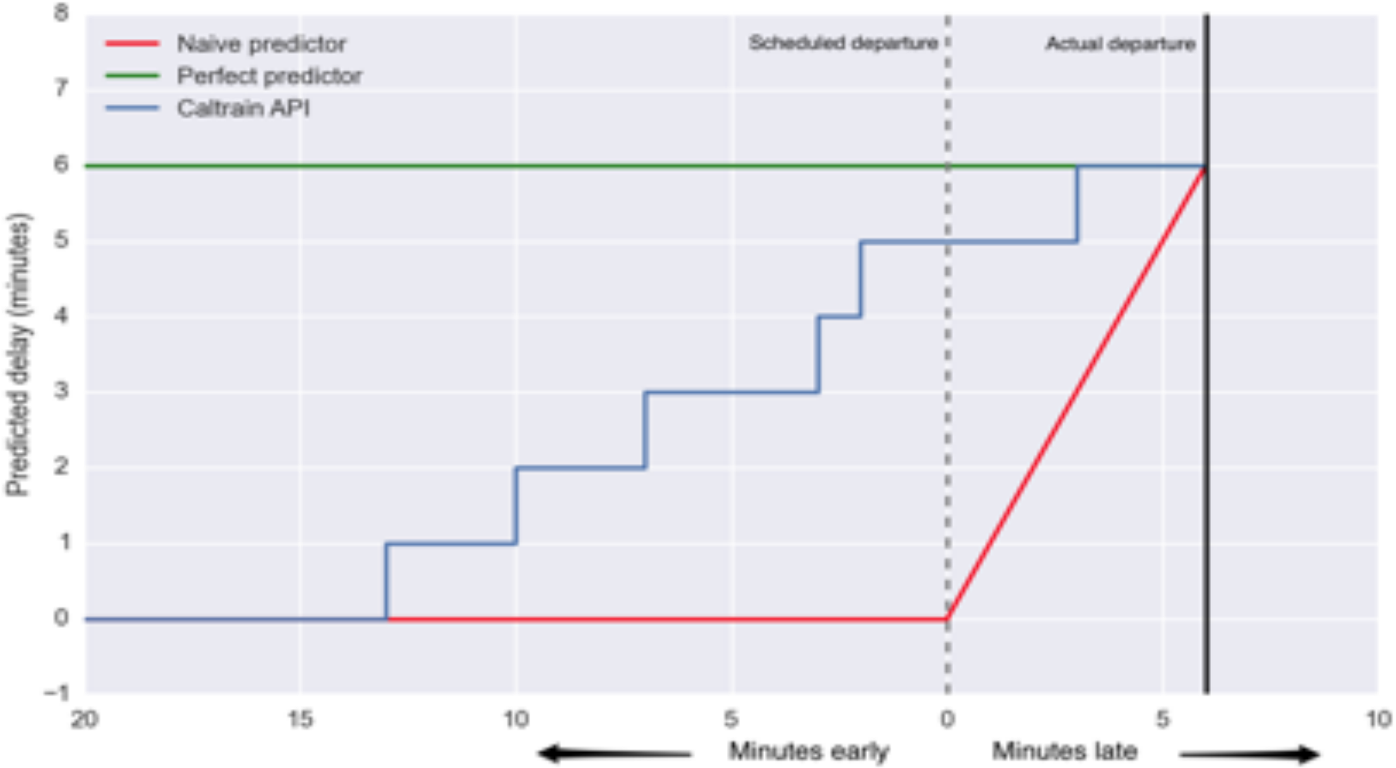
“spread” in observed delays



Observations: Dataset Summary (2)



Prediction: Objective



Delay Prediction Using Various Models

- “on-time (until it’s not)”
- historical average/median

don’t use any “real time” data

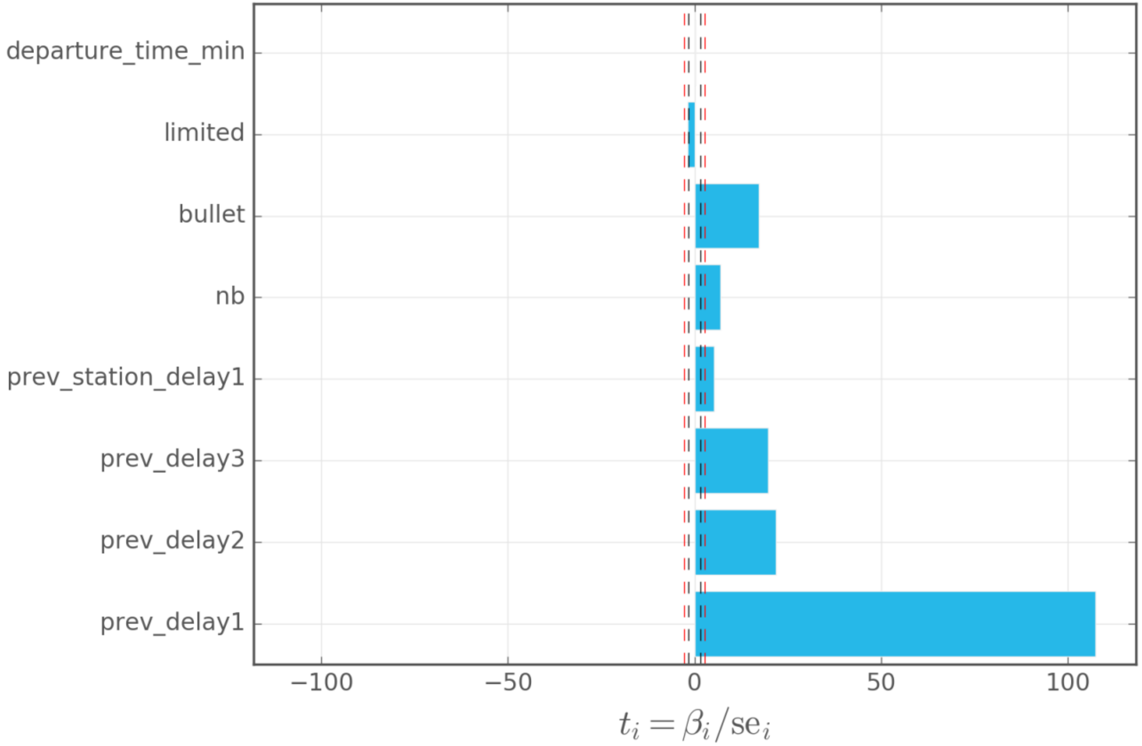
- “previous delay”
- linear regression
- time series regressions

use basic features constructed from real time data

- case-based (nearest neighbor)
- random forest
- bayesian hierarchical models
- neural network

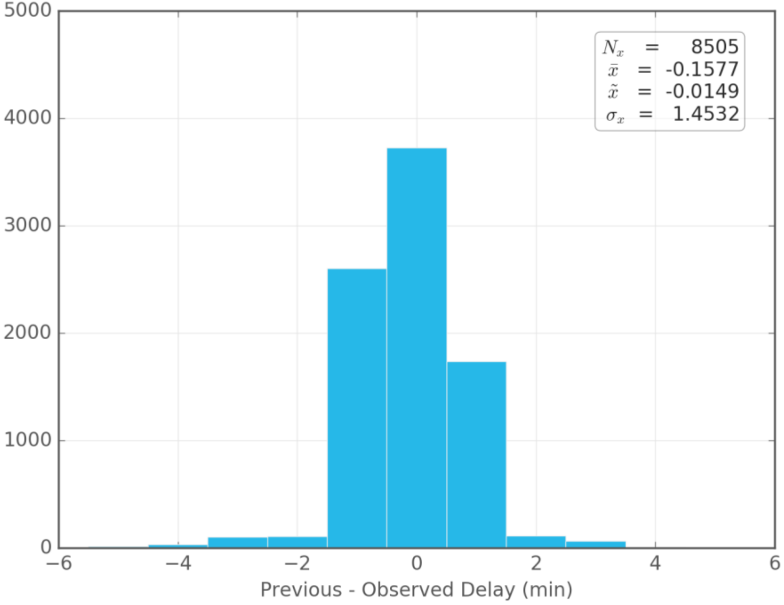
require “arbitrary” model design choices / tuning of hyper-parameters
(also use features constructed from real time data)

Prediction: Features

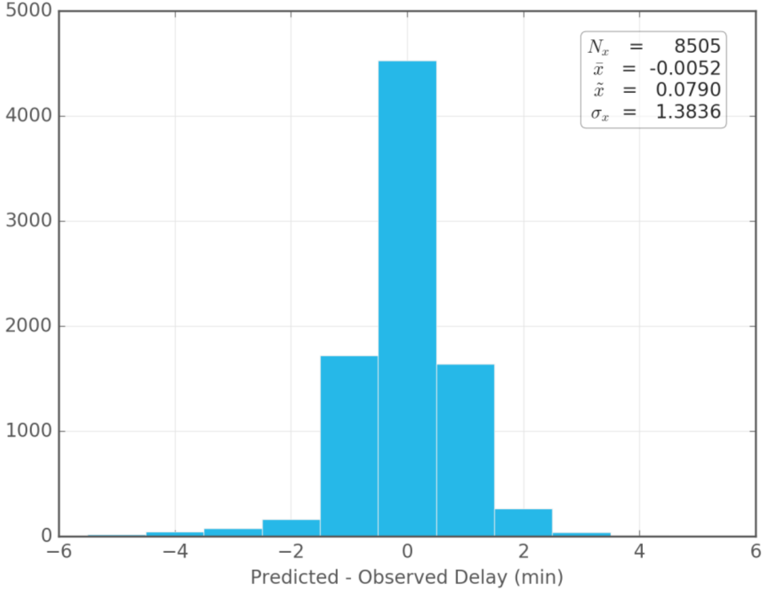


Prediction: Model Evaluation

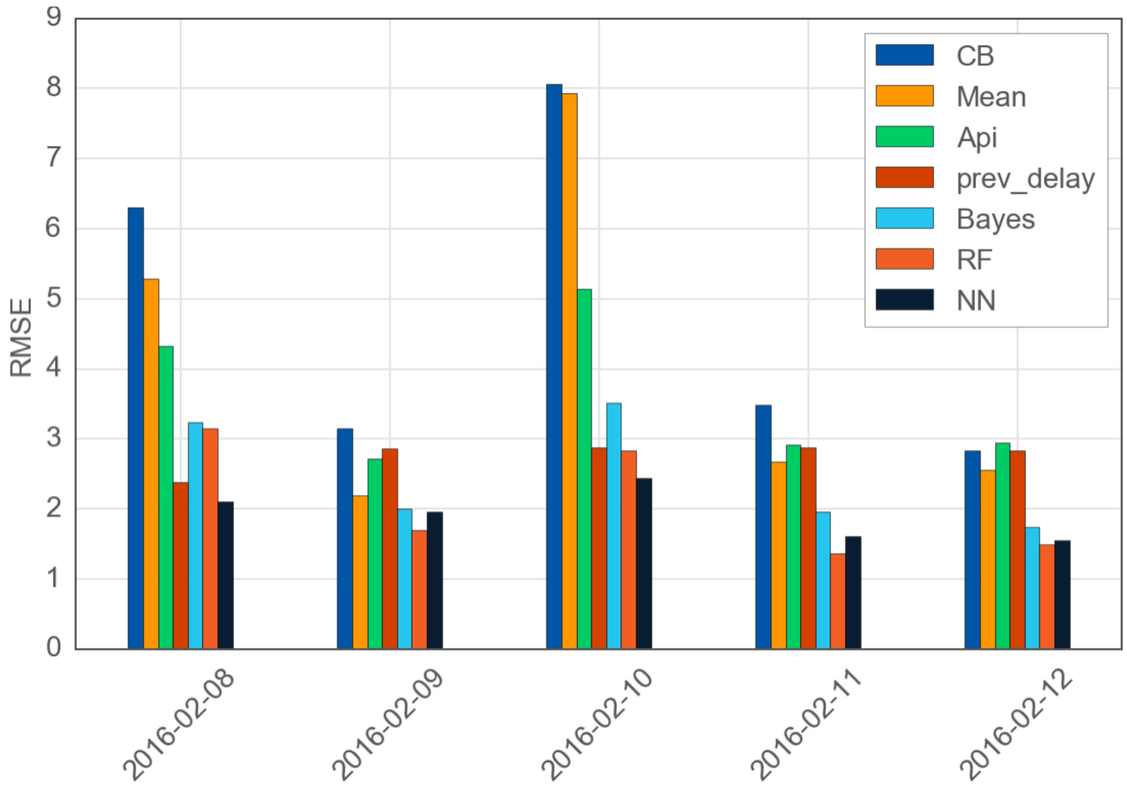
previous delay residuals



neural network residuals



Prediction: Model Comparison



@SVDataScience

SILICON VALLEY
DATA SCIENCE

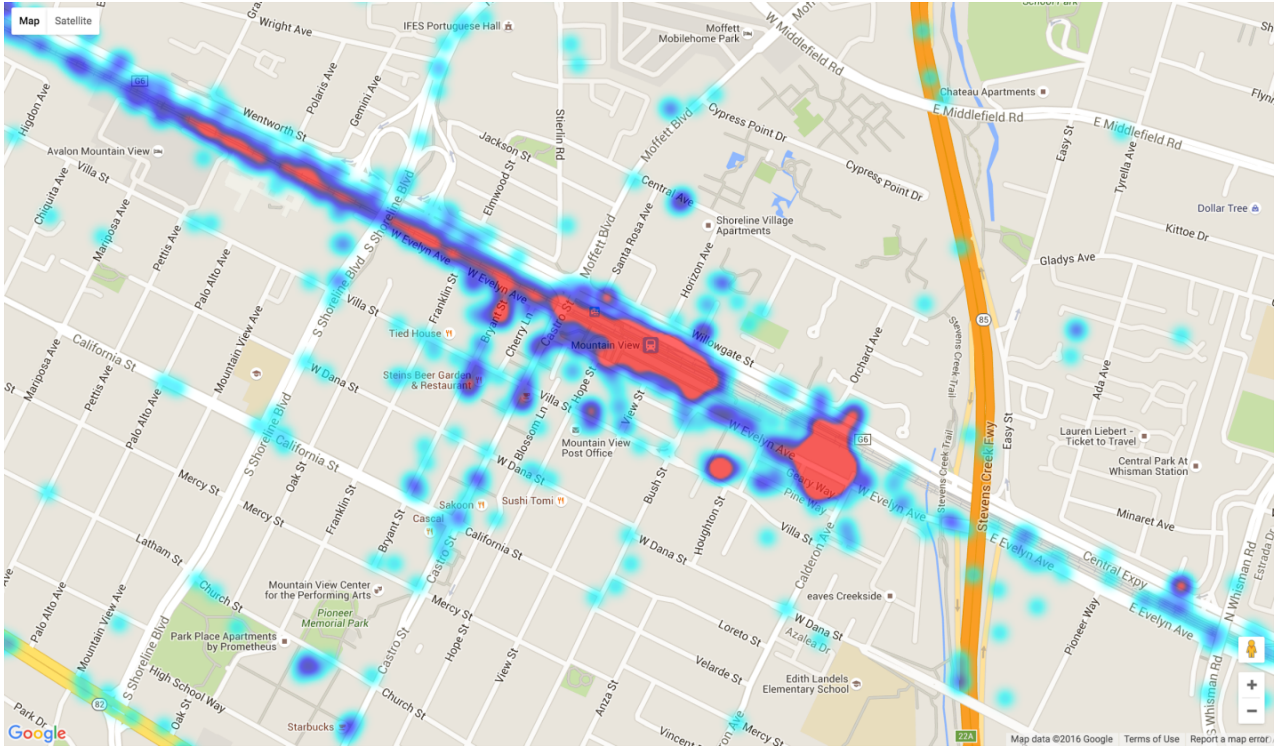
Yes, we're hiring!
info@svds.com

THANK YOU

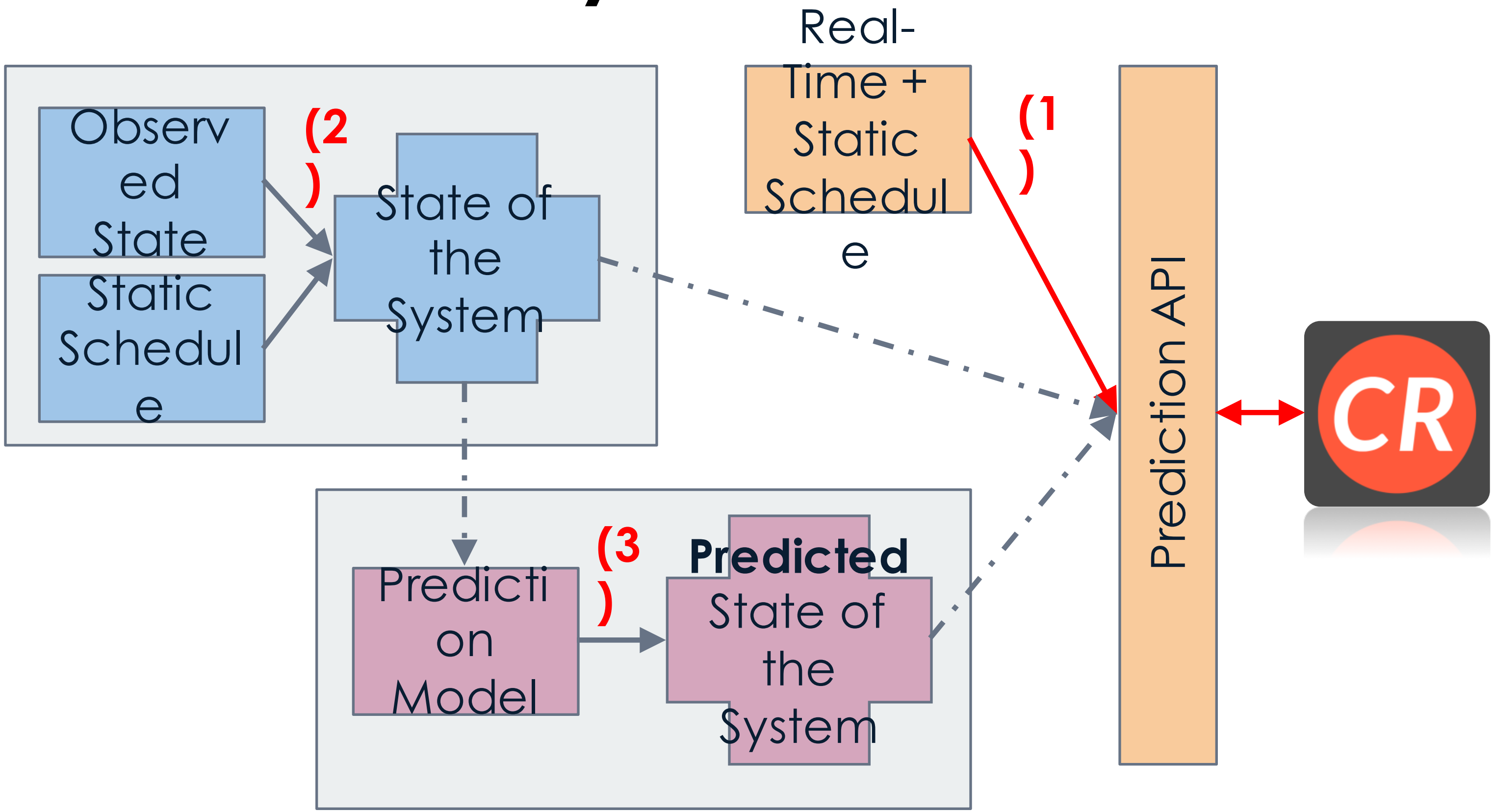
Chloe Mawer | chloe@svds.com
Daniel Margala | daniel@svds.com
Jeffrey Yau | jeffrey@svds.com
@SVDataScience

APPENDIX

Observations: GPS Data

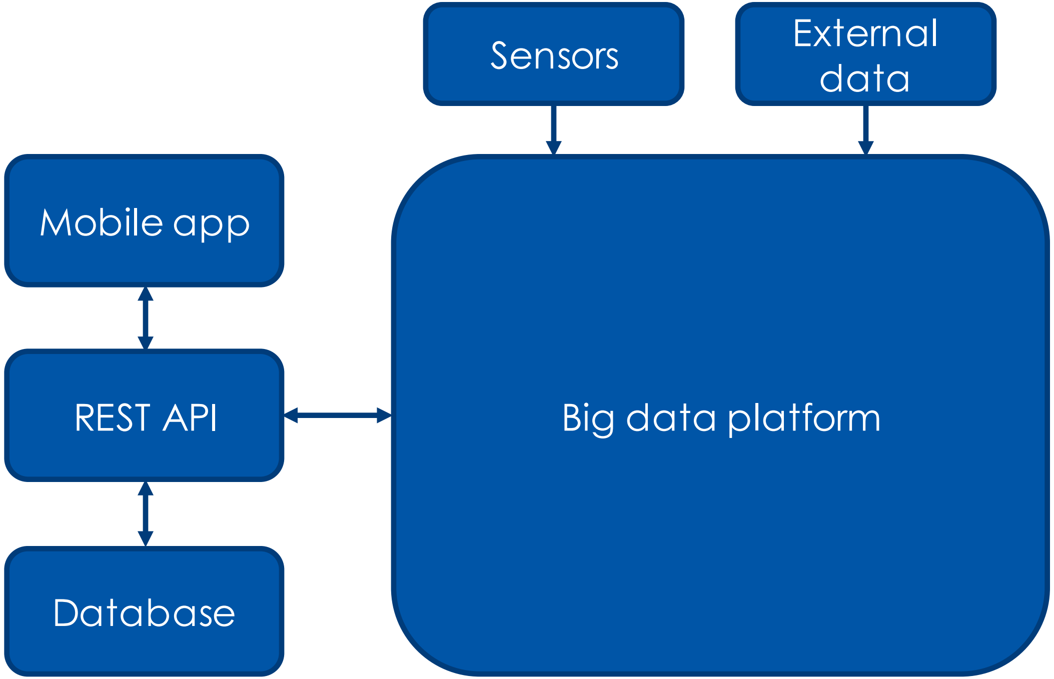


The State of the System

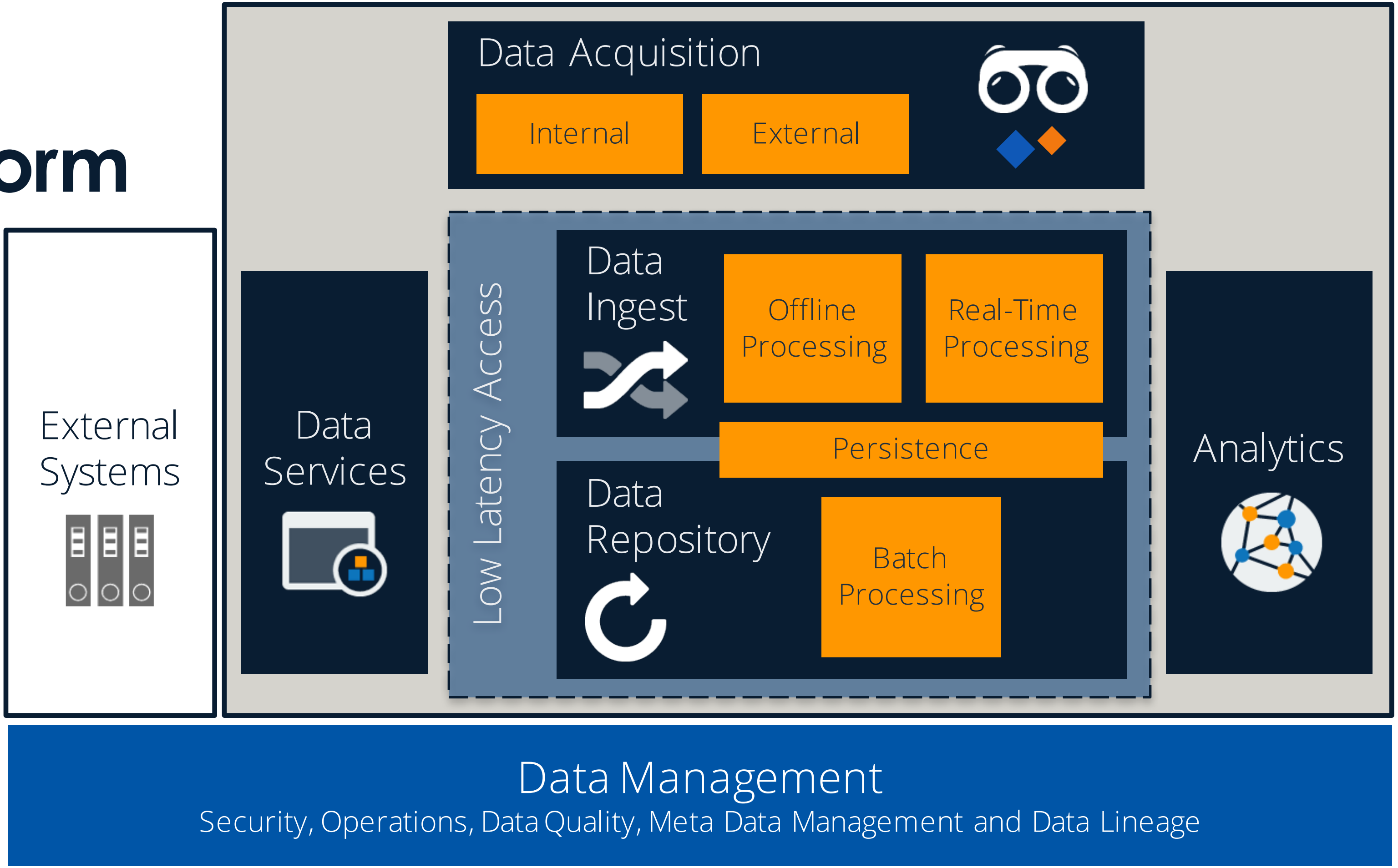



Engineering Aspect of the Caltrain App
Architecture Design
@SVDataScience

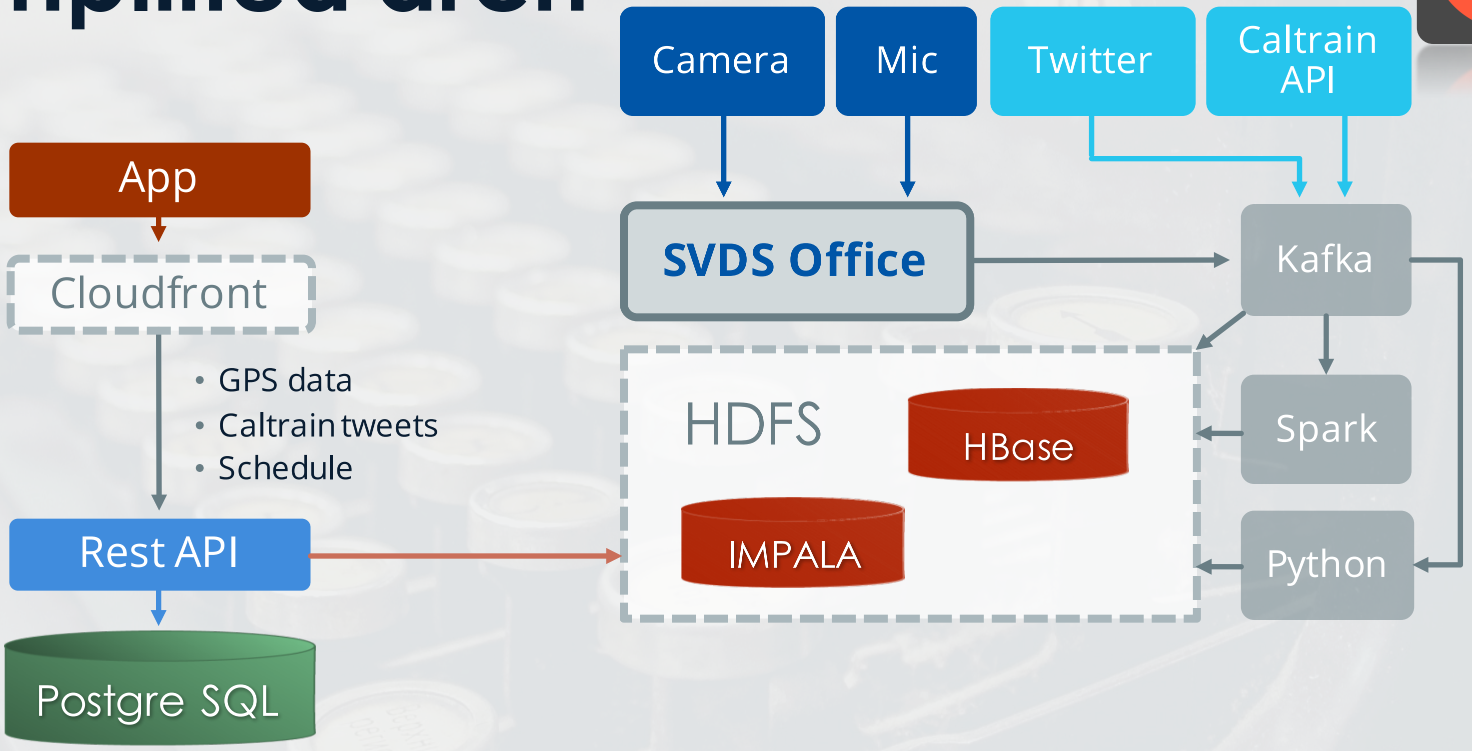
An overview


Sensors
External data
Mobile app
REST API
Database
Big data platform

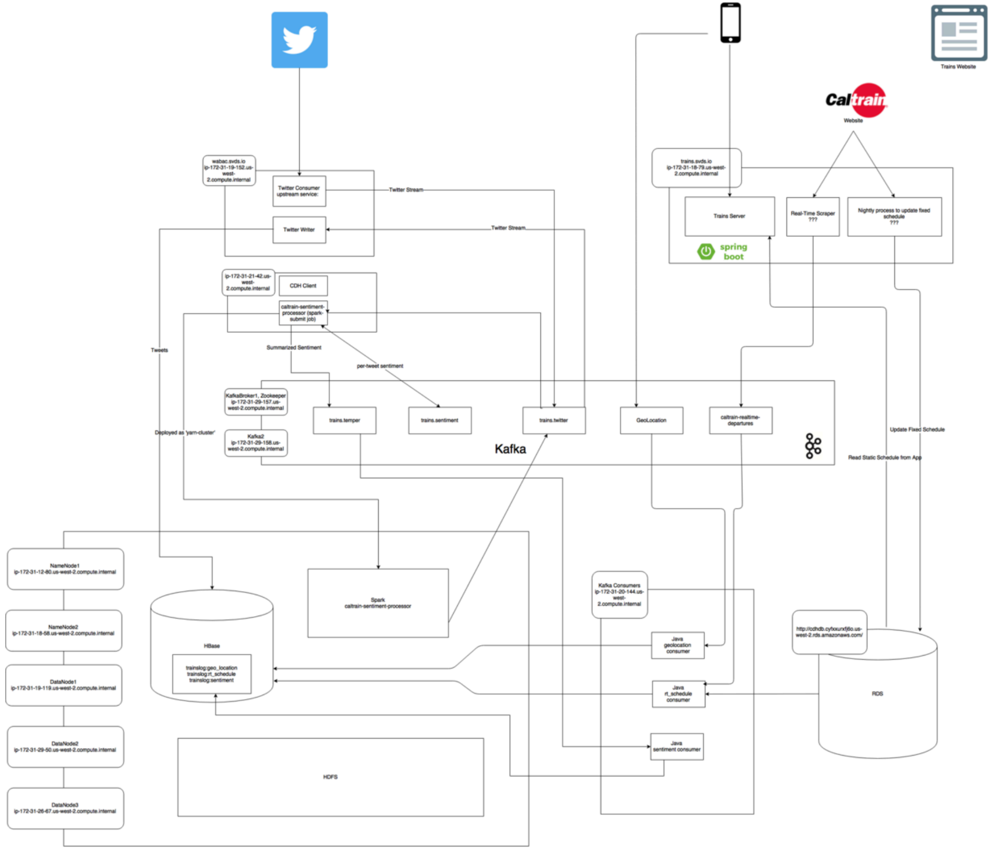
Data platform



Simplified arch



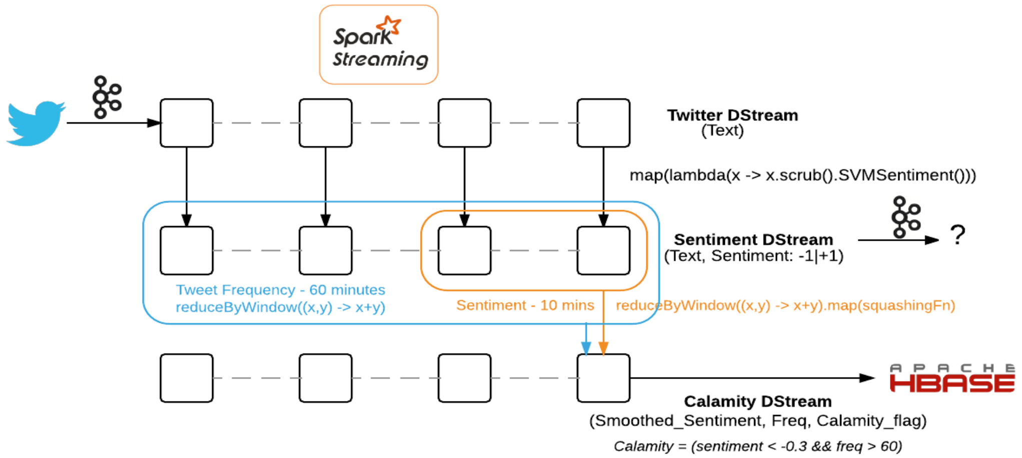
Full Architecture



Platform components

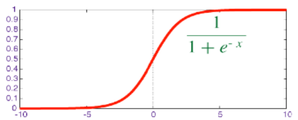
- Kafka – distributed queue, acts as message bus
- Spark / Python – analytics (streaming workloads)
- Hadoop
  - Hbase – persistent storage
  - Impala – fast SQL-like querying
- Postgres – storage of static schedule
- Springboot – REST API for app

Twitter arch

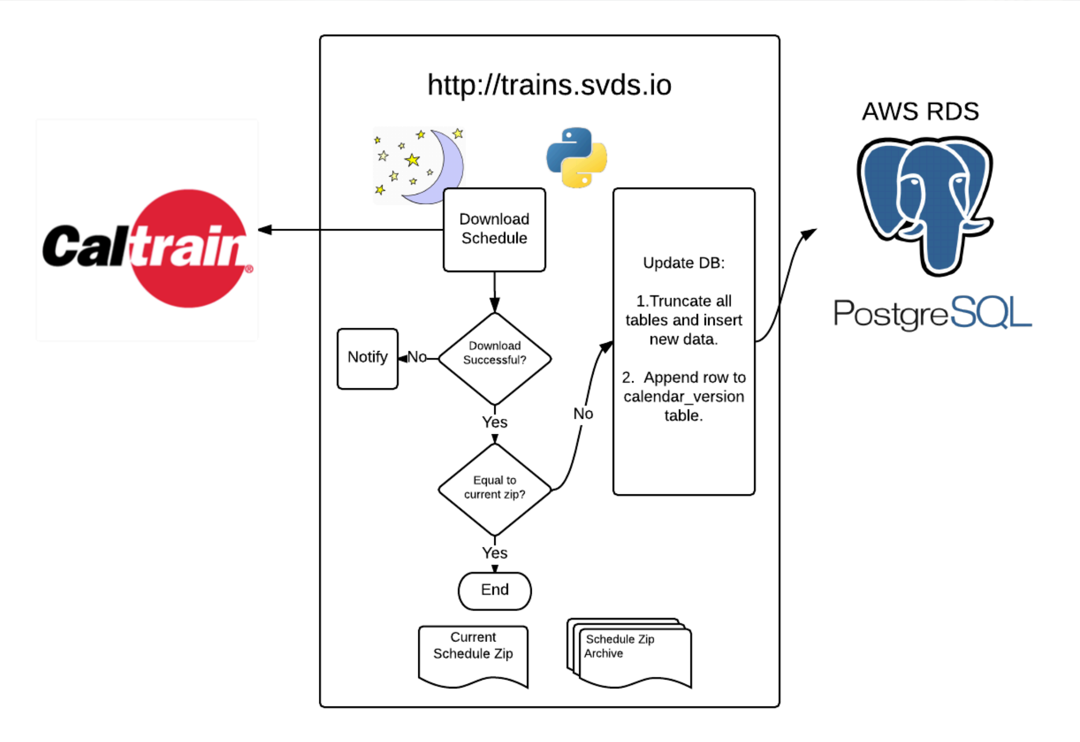


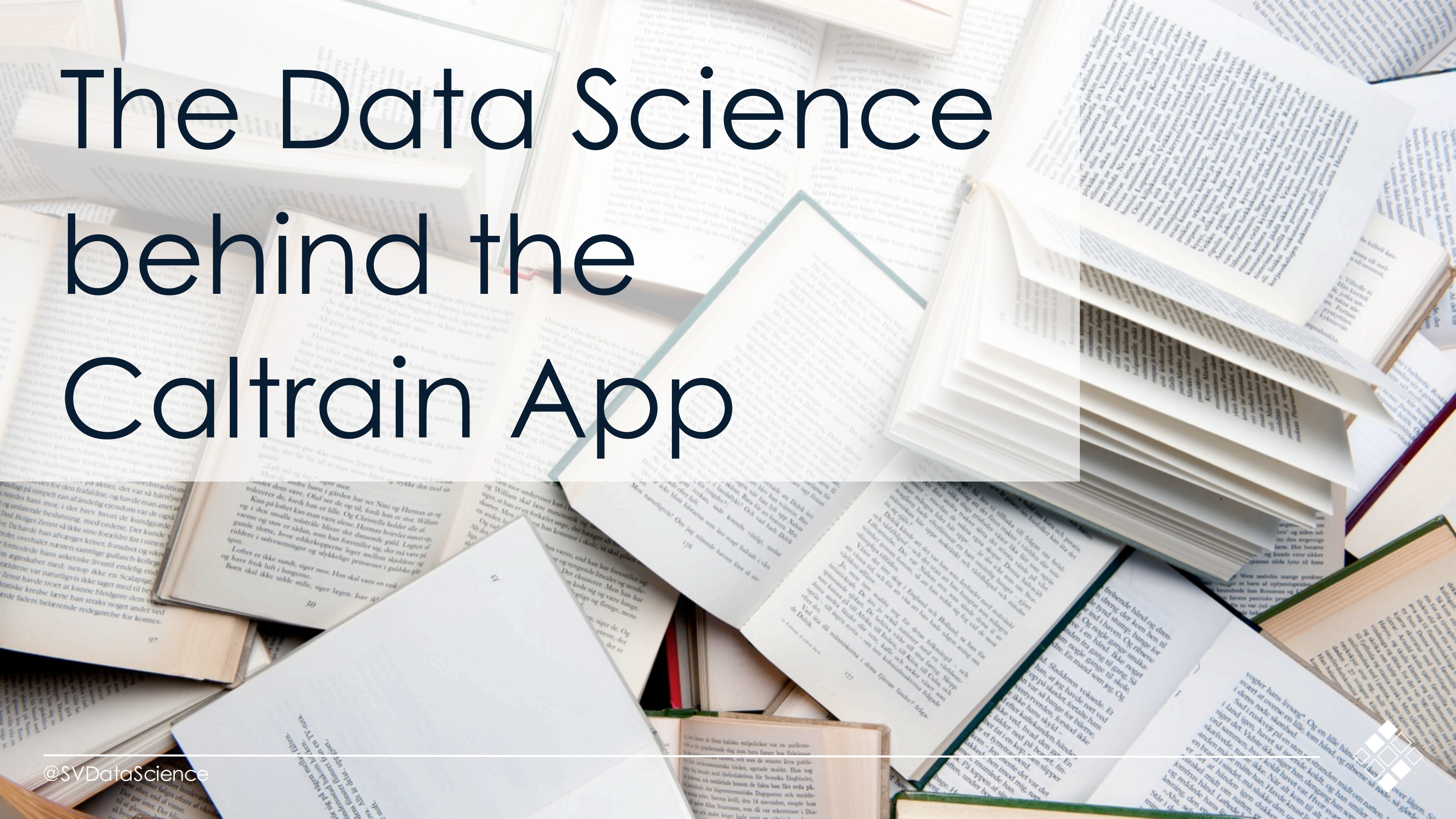
Text scrubbing
- remove hashtags, urls, non-alpha, preserve special tokens :) ! ?
- remove general and context specific stop words e.g. 'into', 'train'
- tokenize, preserving known n-grams e.g. 'San Bruno'
- most common remaining tokens are meaningful e.g. 'accident'

Squashing Function
- allows us to just sum up sentiments in a window
- maps [-inf,+inf] -> [0,1]



Schedule arch





The Data Science behind the Caltrain App

Problem, Challenge, and Our Approach

Problem: Predict arrival time for the next k trains at each station in real-time

Major challenges:

- Caltrain's real-time API information is not a prediction
- Severe system delay is provided manually
- No train location data

Our Approach:

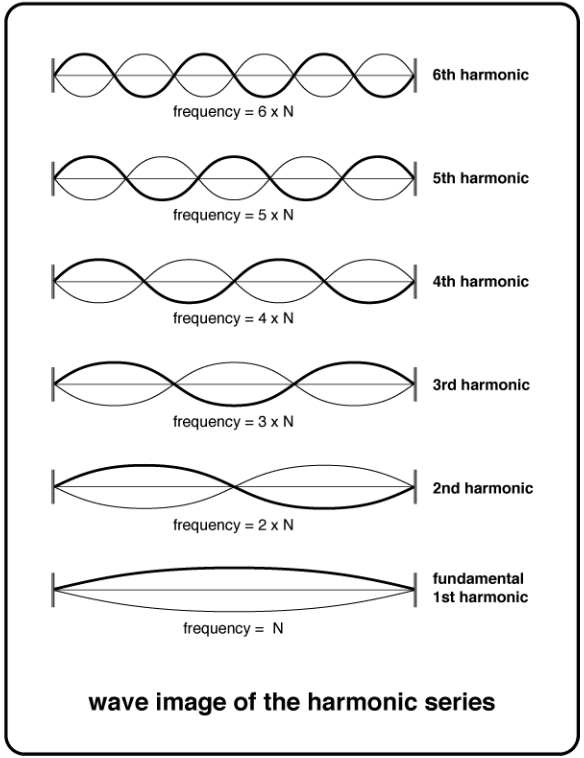
- Build an architecture to collect and process as many signals as we can about the train system
- Design a few solutions with the Caltrain riders in mind

The data science

Train Audio Processing

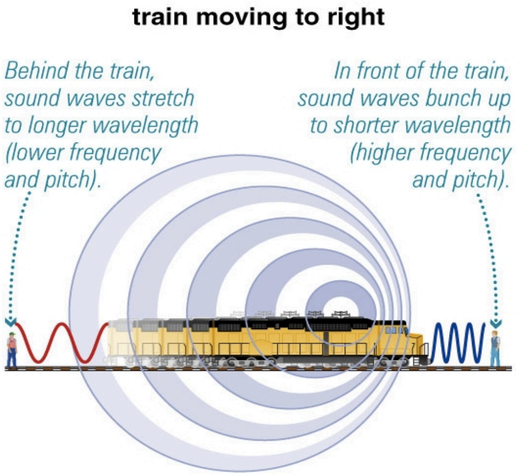
Train Detection

- Need frequencies in addition to volume to distinguish from sirens, car horns, etc.
- Whistles come from standing waves in a cavity, producing harmonics
- Caltrain whistle has fundamental frequency ~340 Hz + integer multiples
- We try to find the fundamental frequency + at least one harmonic



But wait! … there's more

- There are two types of trains with respect to our office – those that stop (locals) and those that pass through (express)
- Knowing whistle frequency allows us to infer train speed, which helps specify which train we are observing



Audio Framework

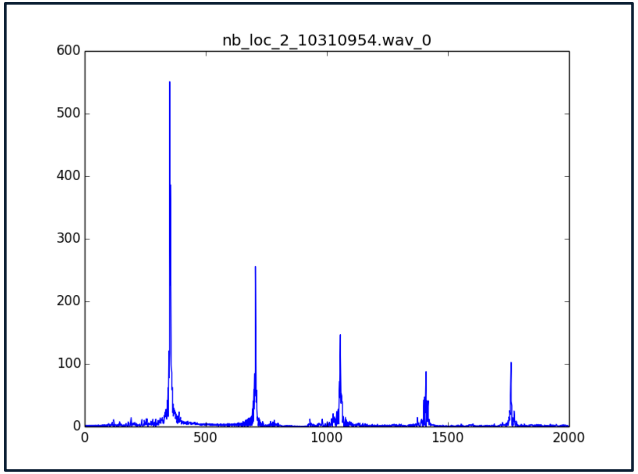
Architecture:

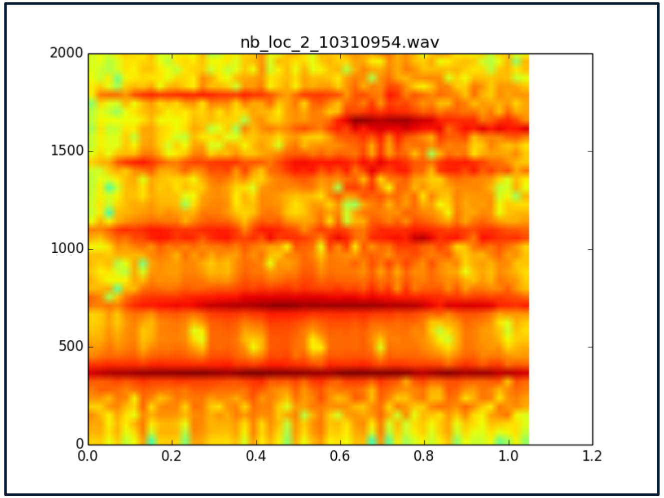
- Microphone → Raspberry Pi → Kafka → Raw Files → Python

Real-Time Analysis in Python:

- Check volume -- if not loud, ignore sample, else:
- Apply FFT to audio signal: $X_k \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi i k n/N}, \quad k \in \mathbb{Z}$
- Find "fundamental frequency" for each time step
- Classify the train into local or express, given list of frequencies

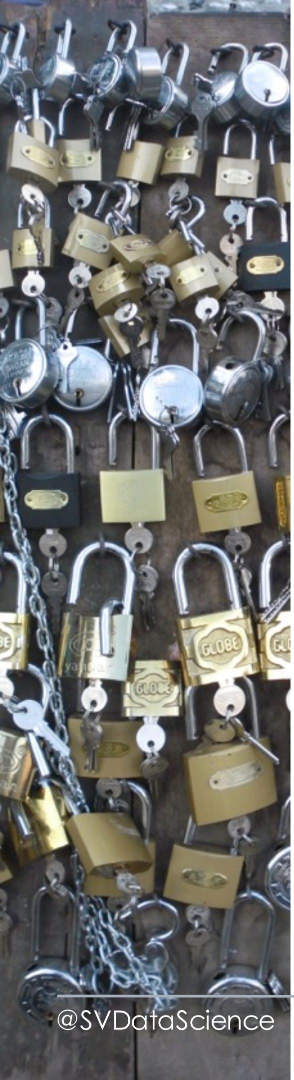
FFTs in Action





Want to learn more?

- Read this blog post! http://www.svds.com/listening-caltrain/



Video detection of trains

Demo

The data science

Tweets Sentiment Analysis and Calamity Index Construction

Twitter sentiment analysis

- A model to classifier positive and negative tweets
  - a dataset of tweets with pre-assigned a sentiment: **Sentiment140 dataset** (http://help.sentiment140.com/for-students/)
  - 1.6m tweets, each labeled with a sentiment, collected between XX and XX
  - sentiment was automatically assigned using emoticons in the tweet text
  - The dataset was balanced in that there were 800,000 positive and 800,000 negative tweets

Twitter sentiment analysis

- We split the dataset into a training and test set using the 80/20 rule.
- Created feature representations of tweets by transforming each tweet into a vector of word count
  - Each dimension in this vector corresponds to a unique word in the vocabulary of the entire corpus
  - CountVectorizer
    - Can accomodate a list (or similar iterable) of strings and will produce the corresponding feature matrix in sparse matrix format

Twitter sentiment analysis

- normalize the results of the above to account for frequency in a particular tweet vs the entire corpus of tweets
    - transform a vector of counts to a vector of tf-idf scores: TfidfTransformer
- Classification Model: Naïve Bayes with parameter tuning and evaluated on F1 score
    - Also tested on Support Vector Machine

Model Results on the Test Set and a Hand-Labeled Caltrain Related Tweets Dataset

Model Results on the Test

```
# predict sentiment labels using the best estimator from the grid search protocol
predicted_labels = gridsearch_classifier_SVM.best_estimator_.predict(tweets_test)
# output the classification report
print(classification_report(labels_test, predicted_labels, target_names=['Negative', 'Positive']))
```

```
             precision    recall  f1-score   support

   Negative       0.81      0.84      0.83    159808
   Positive       0.84      0.81      0.82    160192

avg / total       0.82      0.82      0.82    320000
```

Model Results on Hand-Labeled Set

```
# predict classes using the best NB classifier
nb_pred = best_nb.predict(benchmark_data['text'])
print(classification_report(target_labels, nb_pred, target_names=['Negative', 'Positive']))
```

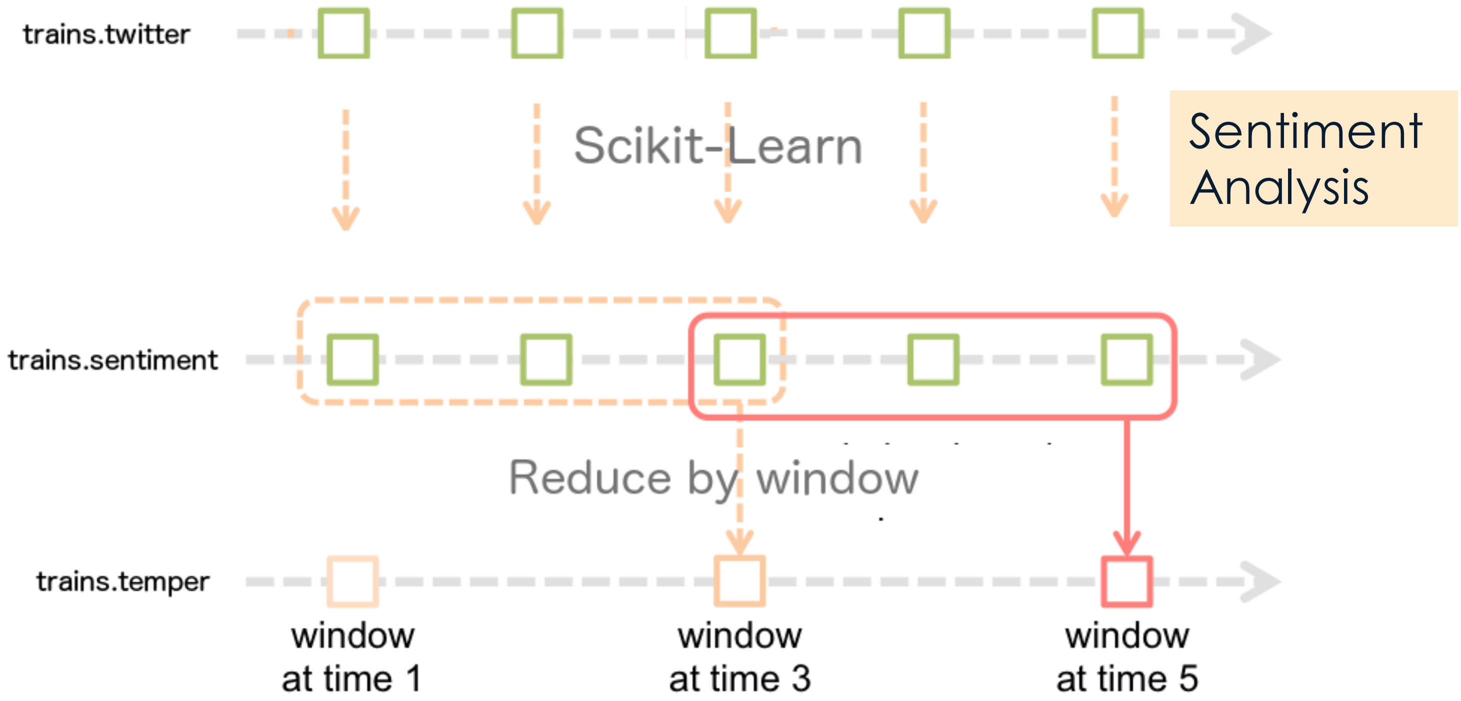
```
             precision    recall  f1-score   support

   Negative       0.63      0.78      0.70       376
   Positive       0.71      0.55      0.62       376

avg / total       0.67      0.66      0.66       752
```

Vocabulary of Positive / Negative Caltrain Tweets

Linking Twitter Sentiment to Delay Classification

- Can sentiment predict delay? Catastrophic delay?
- Can we simply use the **volume of tweets** to identify normal and severe/catastrophic delays
- To what extent **negative sentiment** (or some functions of it) can be use to distinguish normal vs. severe/catastrophic delays?
- Do we need **specific severe-delay keywords** (or some functions of it) to distinguish normal vs. severe/catastrophic delays?

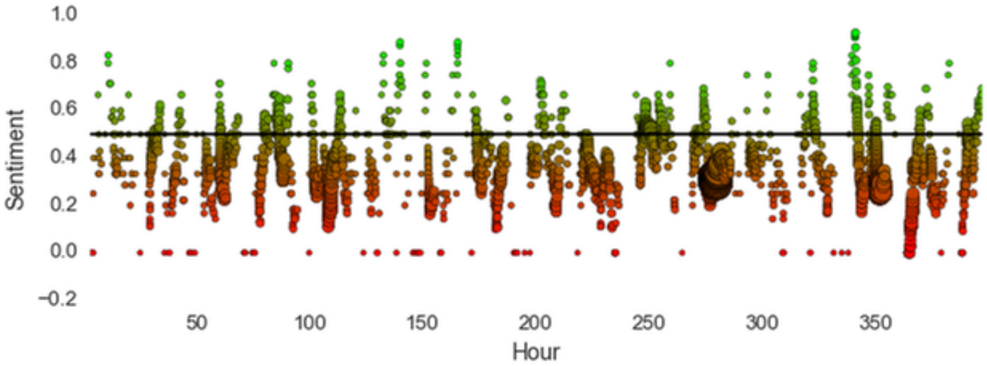
From (Raw) Tweets to Useful Signals



Decide on the volume of tweets within each time interval

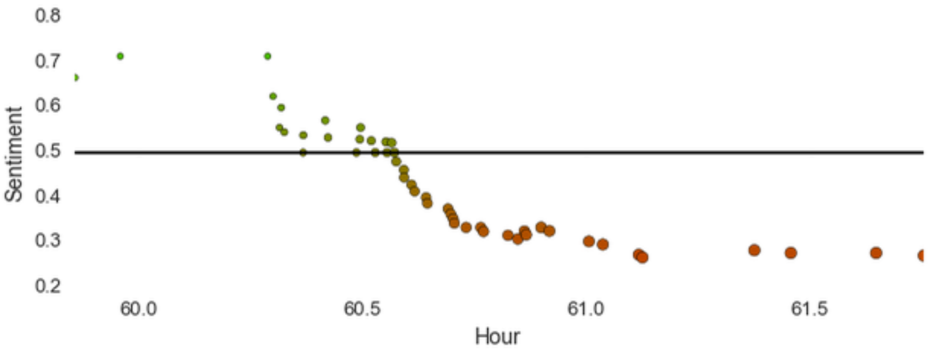
Sentiment Pattern

Positive tweets: sparse, noisy, unspecific
Negative tweets: dense, clustered, specific

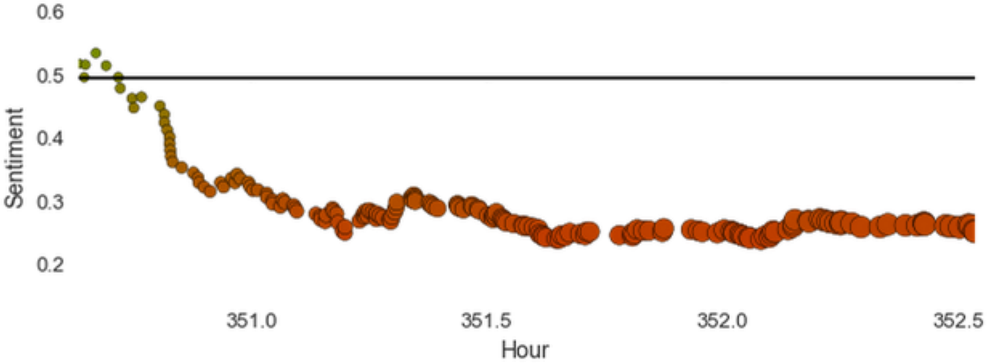


Detected Event: Trains Delayed by 30 mins [May 27]

Frequency – increases to 30 tweets / hour
Sentiment – moves to 0.3 with decreasing variance

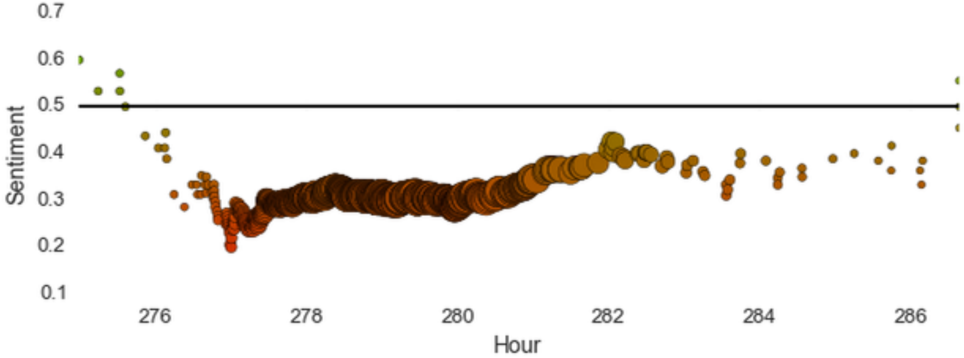


Detected Event: Engine Overheated in Menlo Park [June 9]

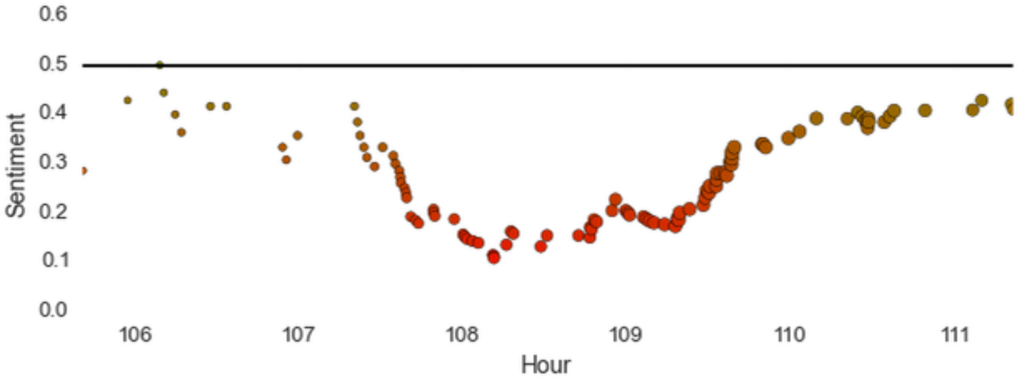


Detected Event: Person Killed by Train in Atherton [June 5]

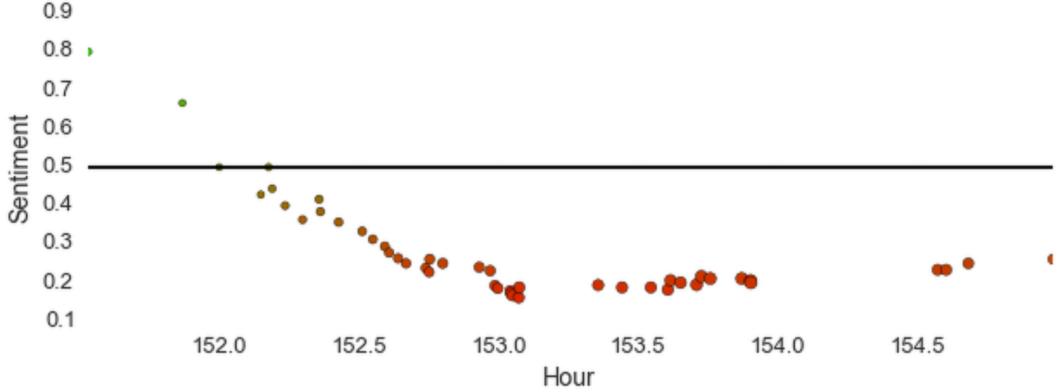
Highest frequency in sample: > 120 tweets / hour
Longest duration event in sample: > 9hours



Detected Event: Caltrain Conductor Assaulted [May 29]

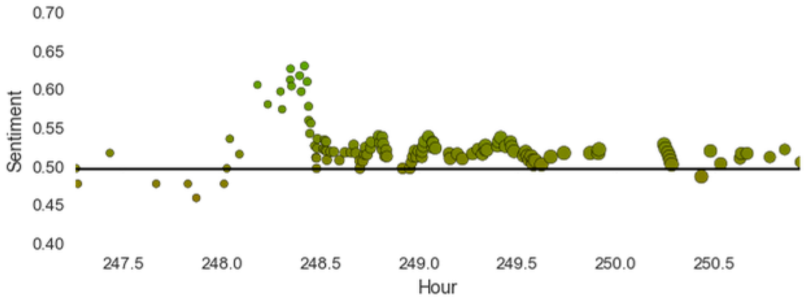


Detected Event: Giants Game [May 31]



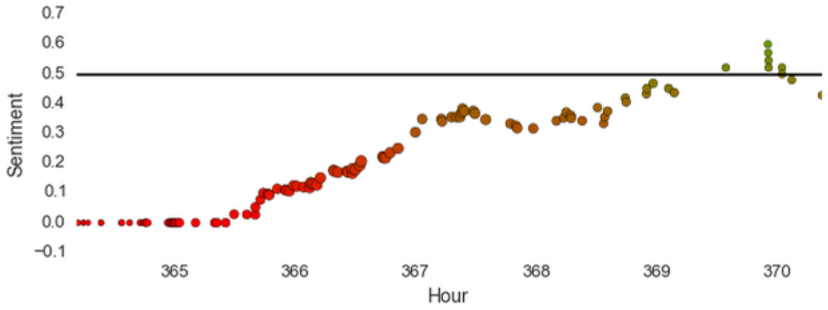
However, High Frequency Alone Is Insufficient! [June 4]

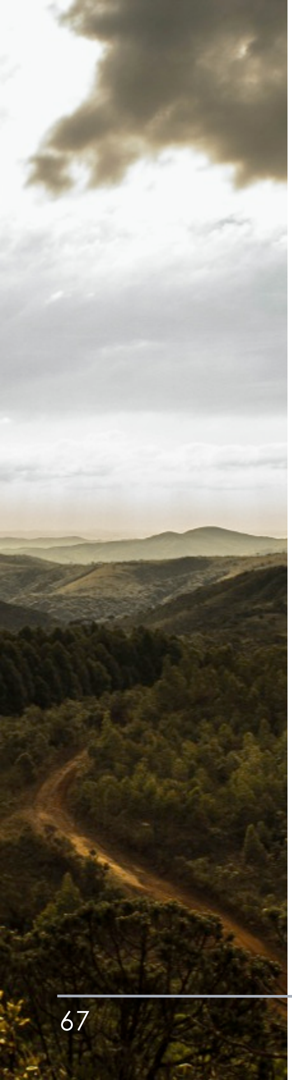
Lively discussion between Caltrain Tweets on Caltrain's proposal to increase bike capacity by reducing the number of available bathrooms



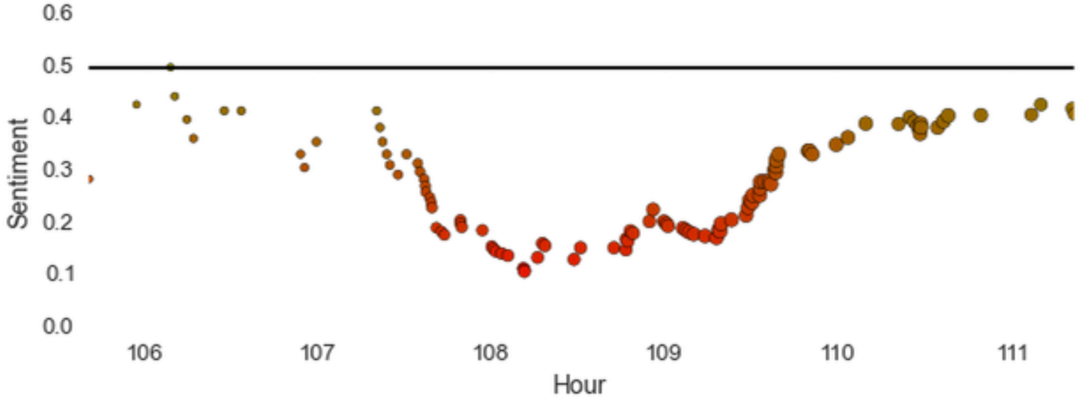
False Positive: News From Multiple Sources [June 9]

Multiple occurrences of a news tweet unrelated to sentiment, but the text happens to be classified as negative ("Developer to build parking lot near Caltrain") Moving average floored at 0.0 and takes 4 hours to fade away.





Sentiment Allows us to Distinguish Calamity from Chatter



Flagging keywords to Indicate Failure/ Excessive Delay

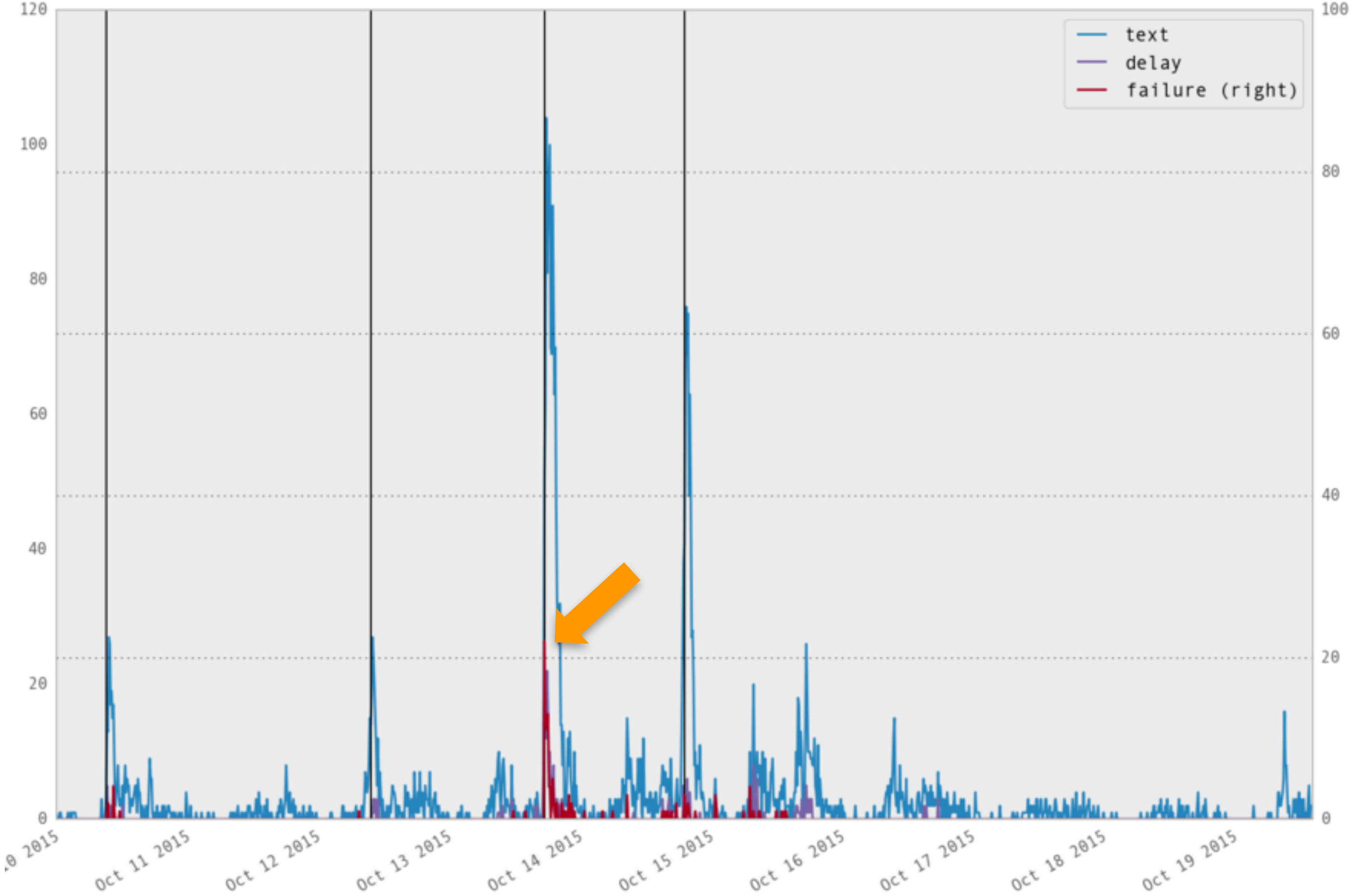
Events

- **Oct 07 10:45 pm**
  - 197 struck a vehicle at bayswater ave
- **Oct 10 8:19 am**
  - 423 stopped due to 'emergency incident'
  - no other affected trains according to twitter
- **Oct 12 8:55am**
  - SB324 stalled (didn't say where in twitter) and coupled to 324
  - 20 to 60 min delay (longest for stalled train + 324)
- **Oct 13 4:43 pm**
  - pedestrian incident at San Mateo, 261
  - Transit PD stopped single tracking so both ways were jammed
  - incident train was cancelled, 30-60+ minute delays
  - 282, 284, and 386 departed late from SF
- **Oct 14 6:18 pm:**
  - stalled train 273 near Millbrae
  - delays ~20-40 minutes (NB trains behind stall were on later side)
  - 190 had 50 min turnaround delay from SF
  - followup trains 279 and 287 turned into local trains at MIL

- Flag certain words as potentially indicating failure or excessive delays.
- The advantage over NLP approaches: a much smaller training set is needed

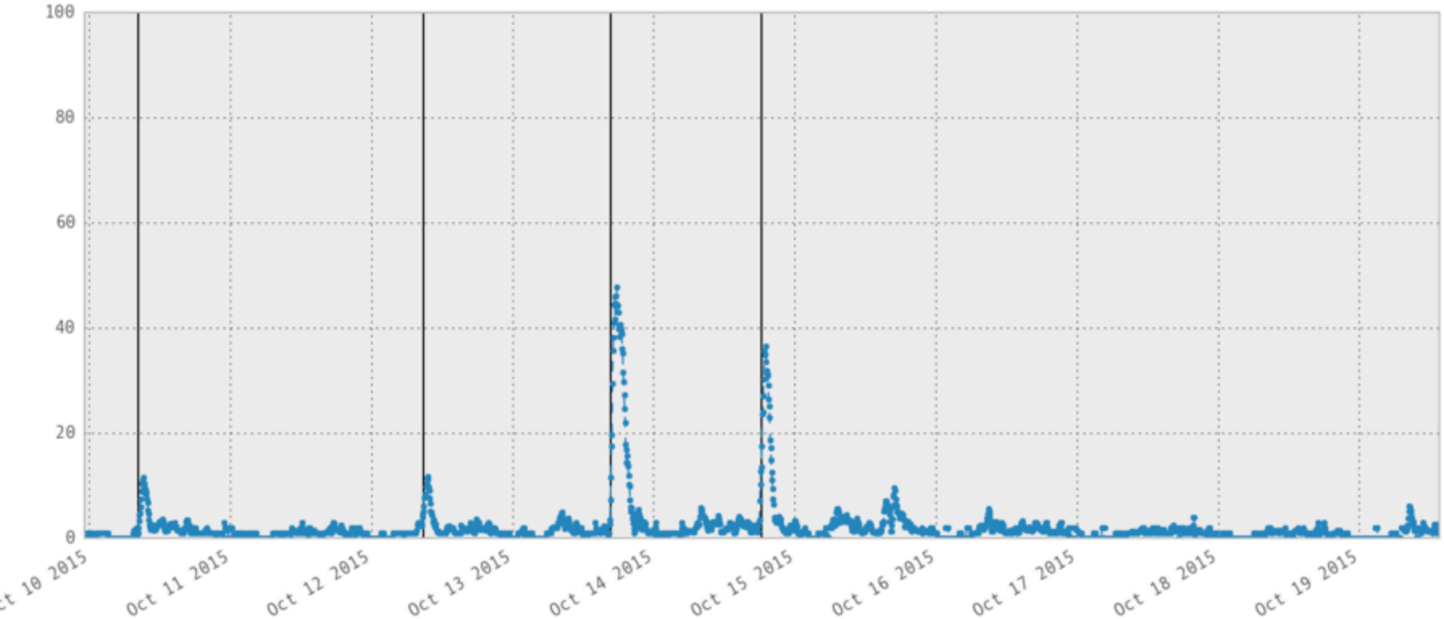
```
failure_words = ['pedestrian',
                 'vehicle',
                 'stalled train']

delay_words = ['heavy ridership',
               'emergency incident',
               'reduced speed', 'restricted speed',
               'expect delays',
               'min late', 'mins late',
               'switch issue', 'switching issue'
              ] + failure_words
```

Volume of Tweets and Volume of Keywords



The Volume of tweets and keywords spike at the time when there was a severe delay

Volume of keywords indicating Failure/ Excessive Delay

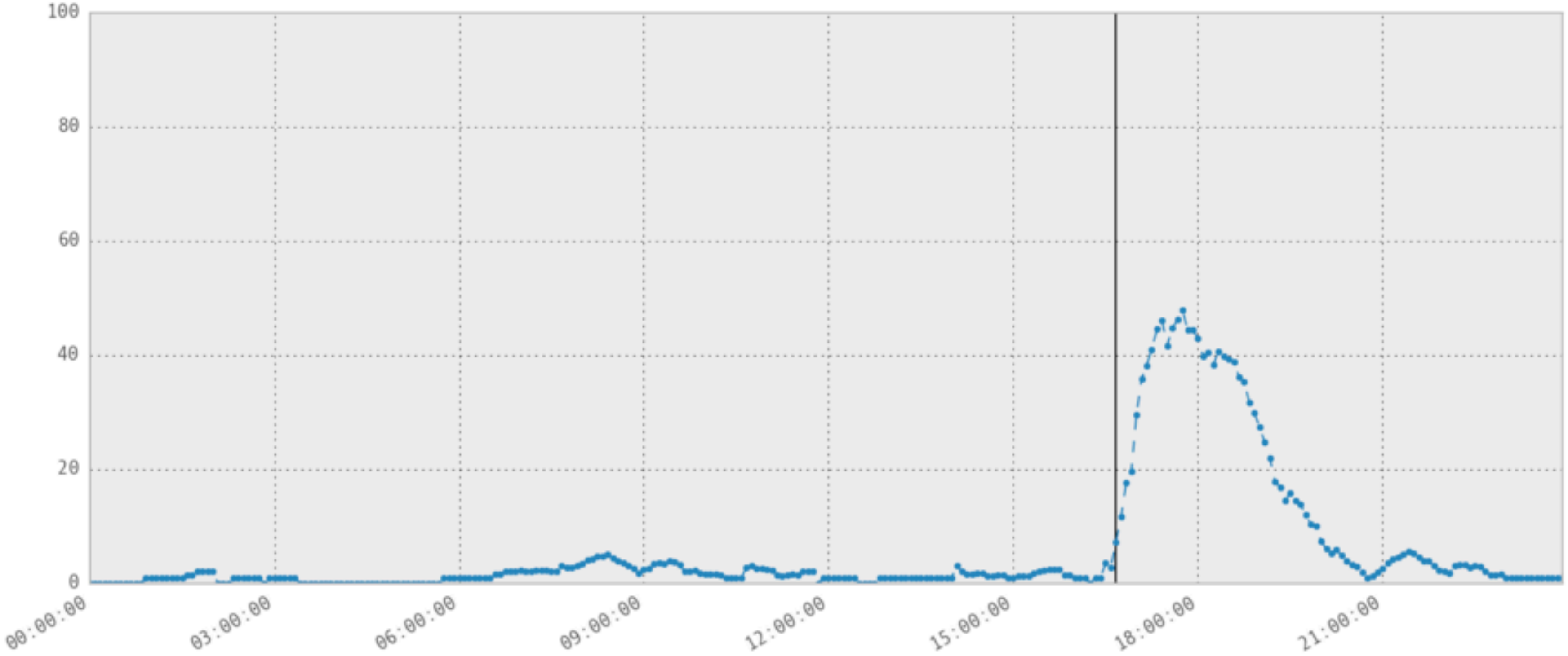


- The Volume of tweets and keywords spike at the time when there was a severe delay

- Volume is measured in an one-hour interval, updated every 10 minutes.

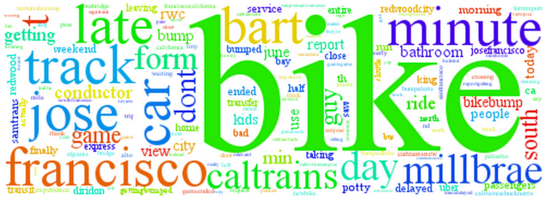
Volume of keywords indicating Failure/ Excessive Delay

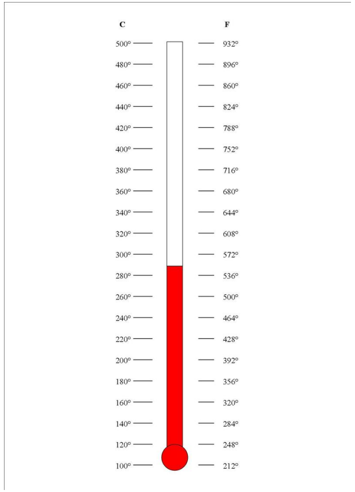
… further zoom in on the event on October 13, 2015

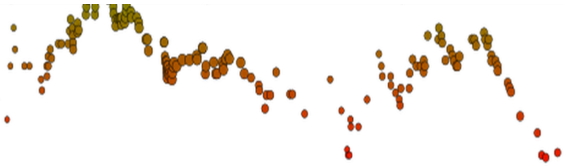


Sentiment itself has values to the Caltrain riders

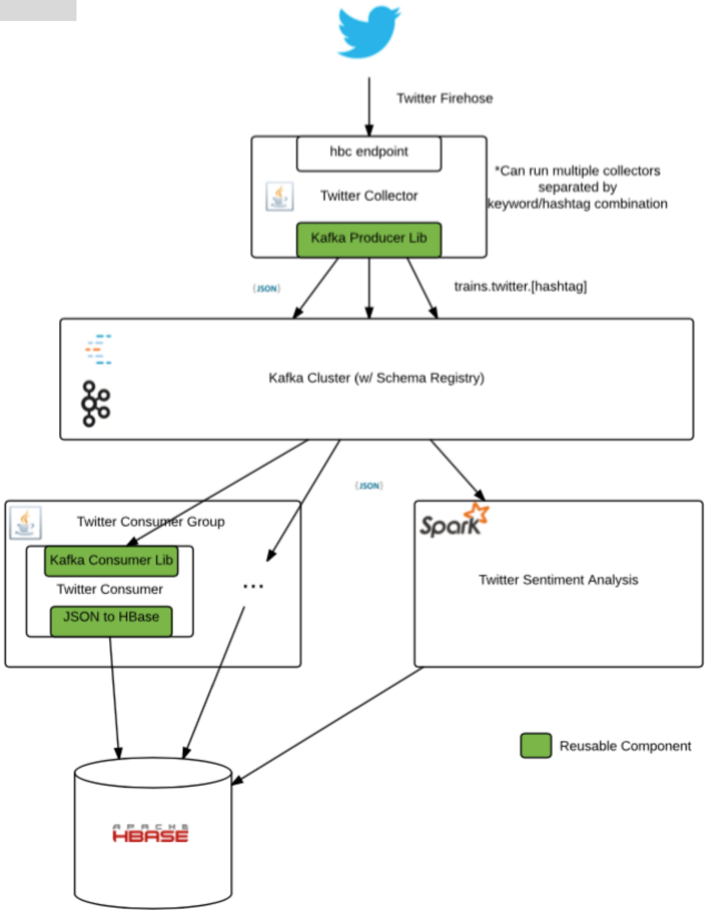
Different ways to display sentiment to the app user







TWITTER PIPELINE



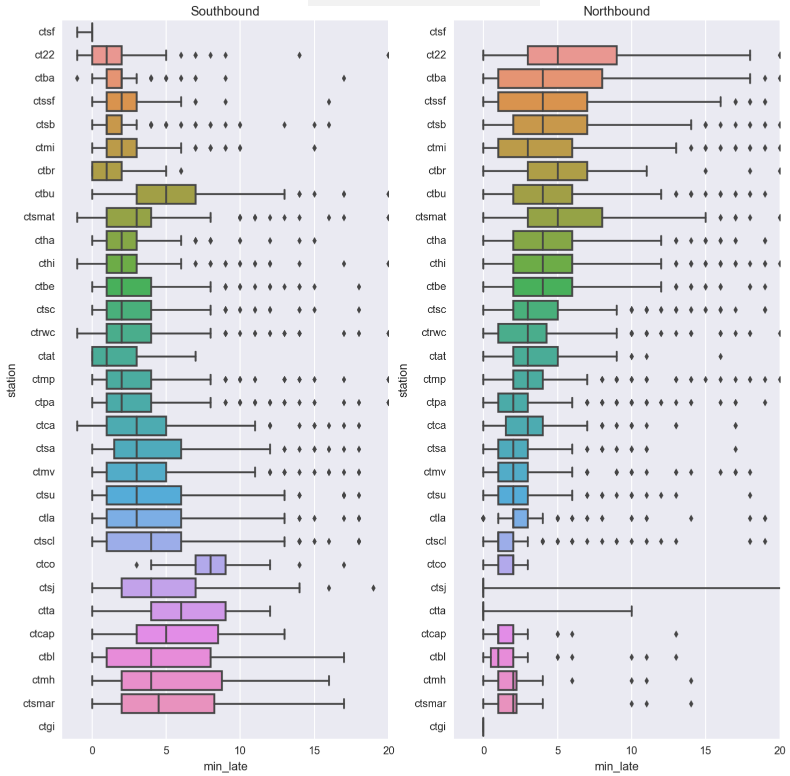
In-Progress and Future Work

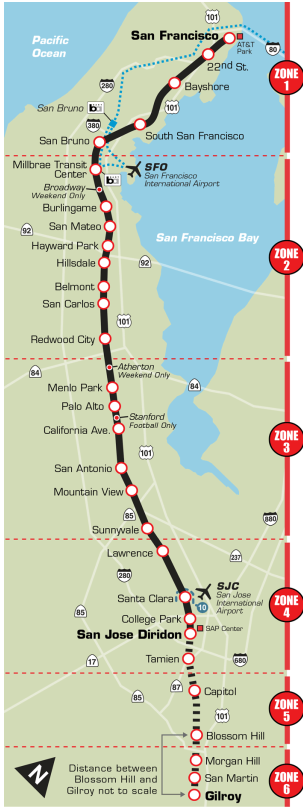
The data science

Empirical Patterns of Train Delays

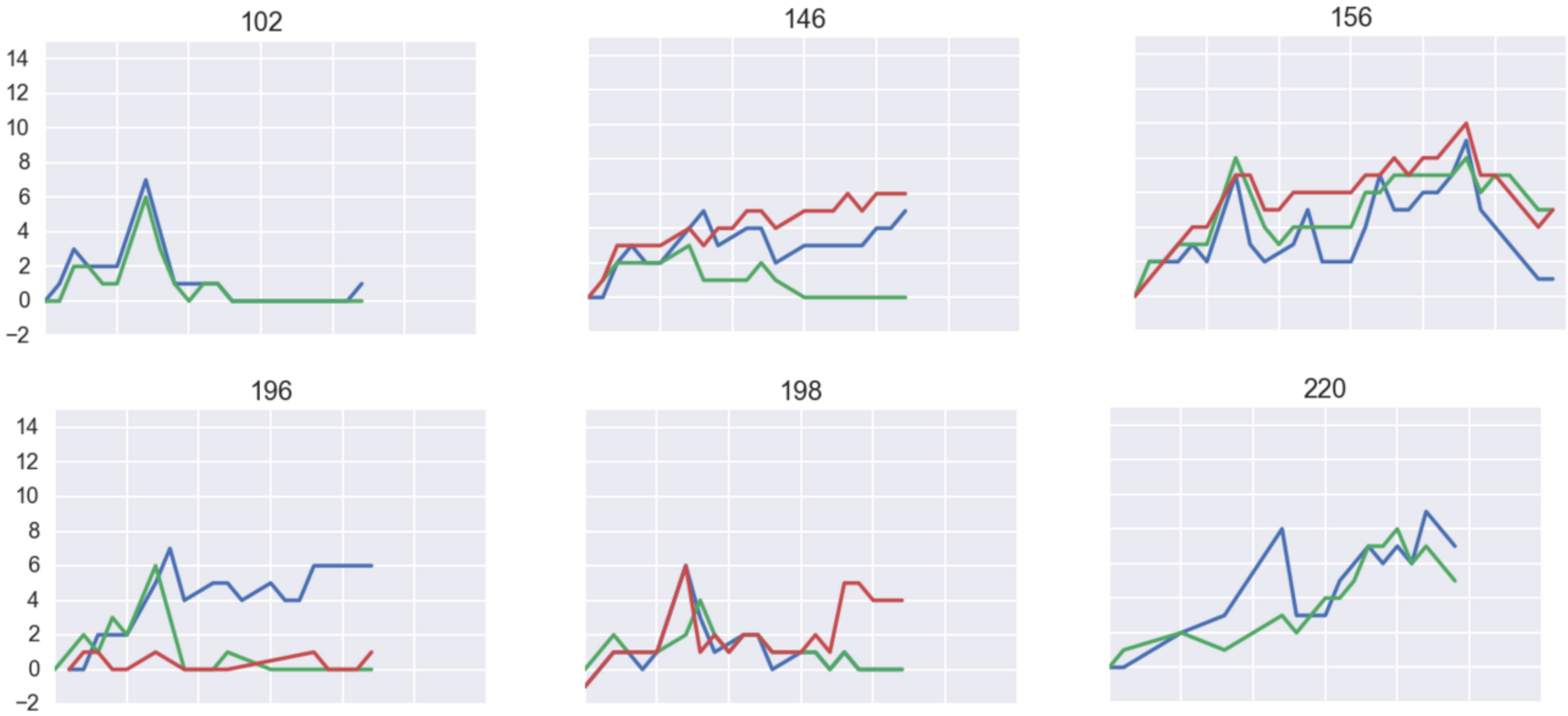
Delays vary greatly by stations

All trains



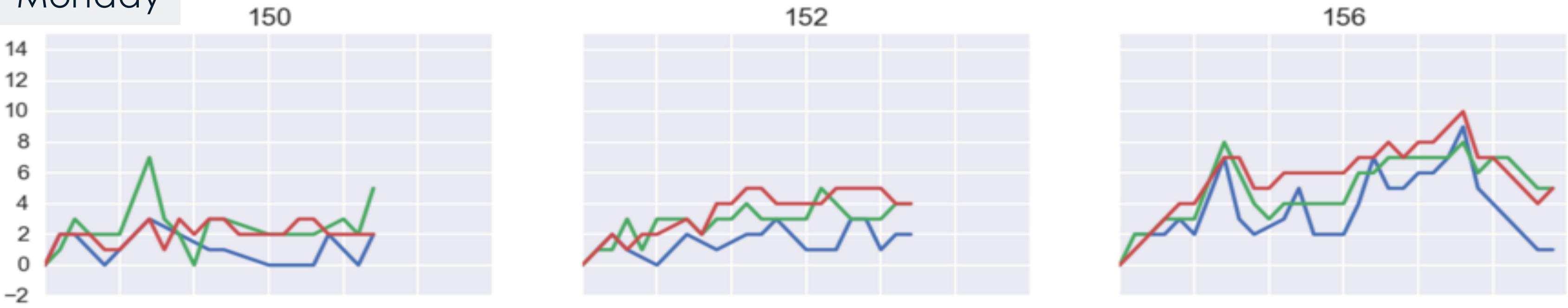


Delays vary greatly by trains (i.e. time of day)

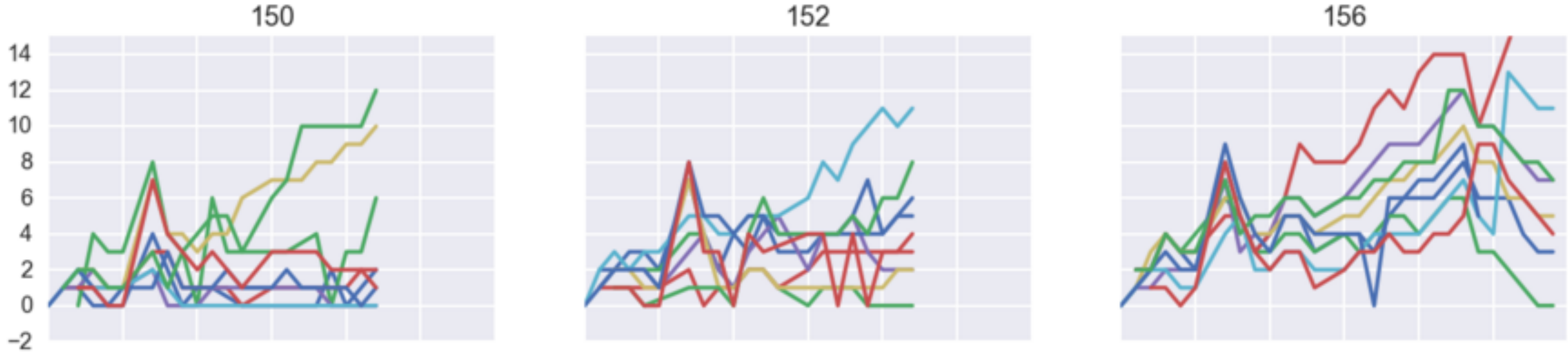


Delays vary greatly by day (for the same train)

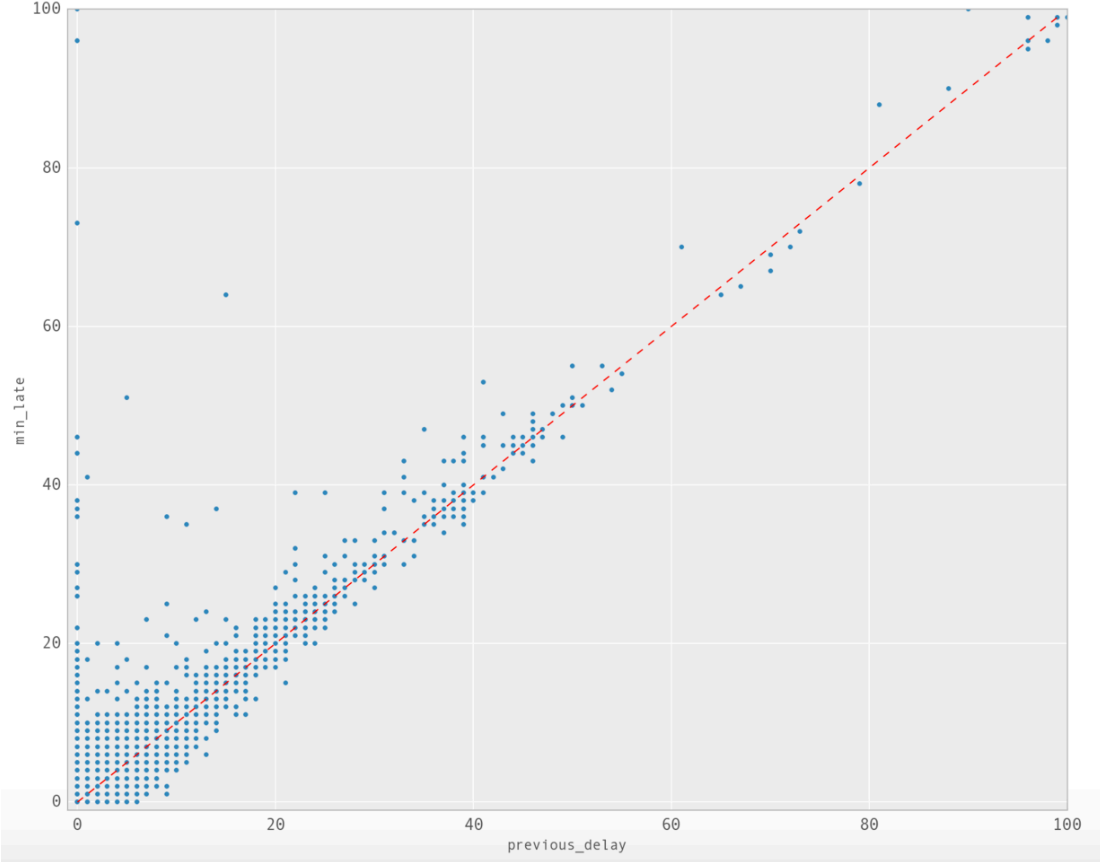
Monday



Tue - Thu

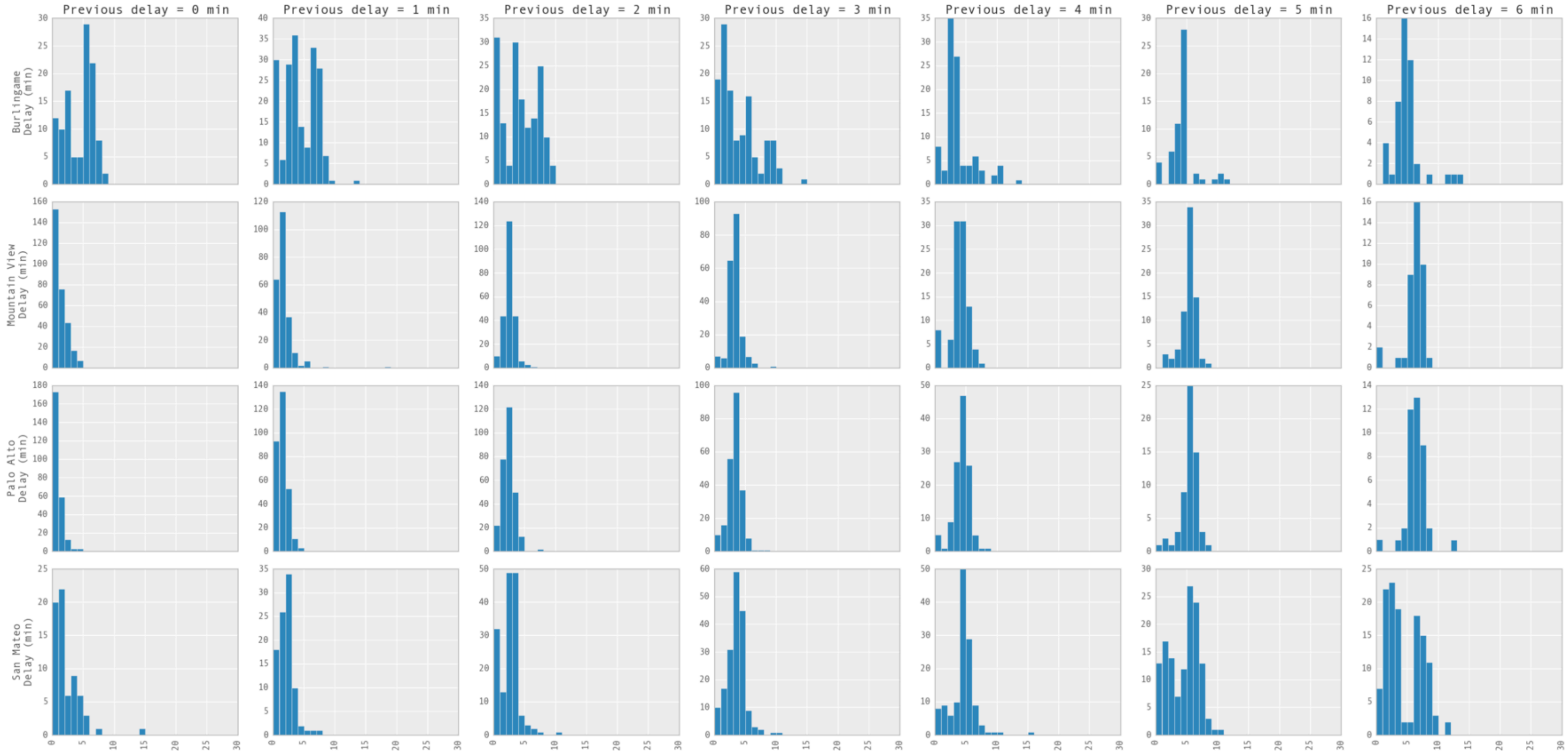


Previous Delay is a Strong Predictor

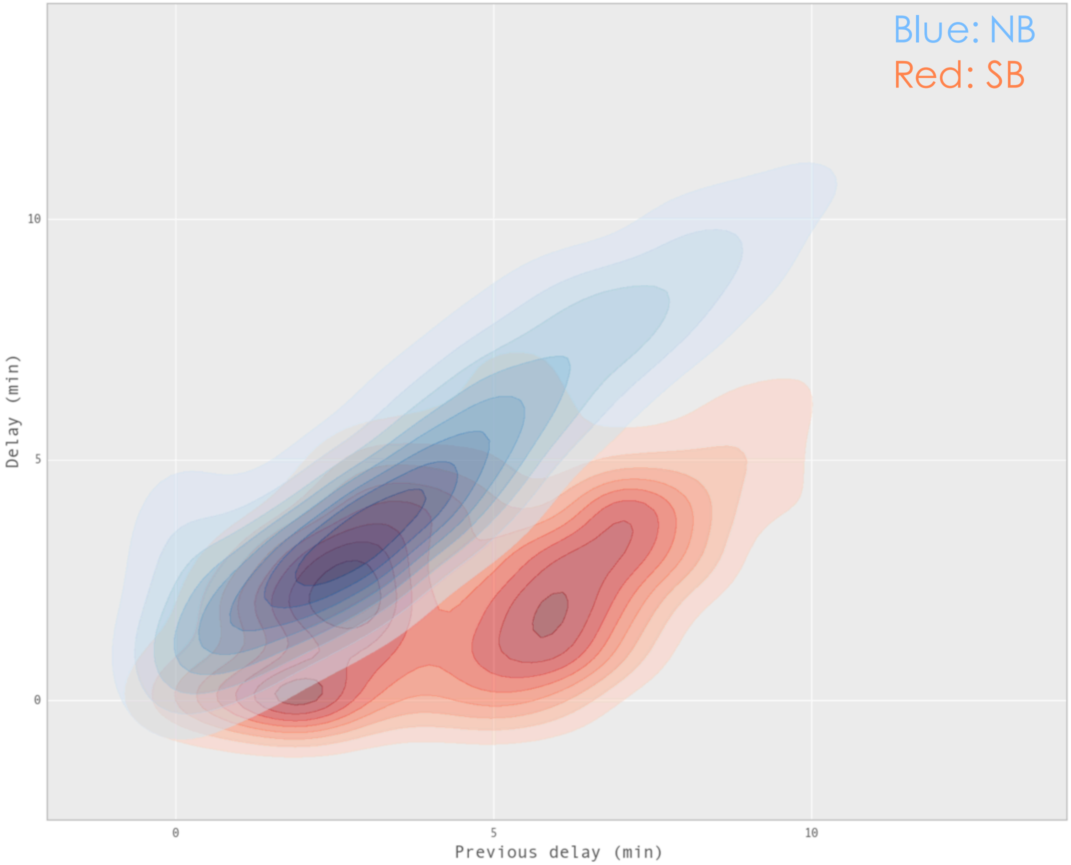


... But, Knowing Previous Delay alone is Not Enough

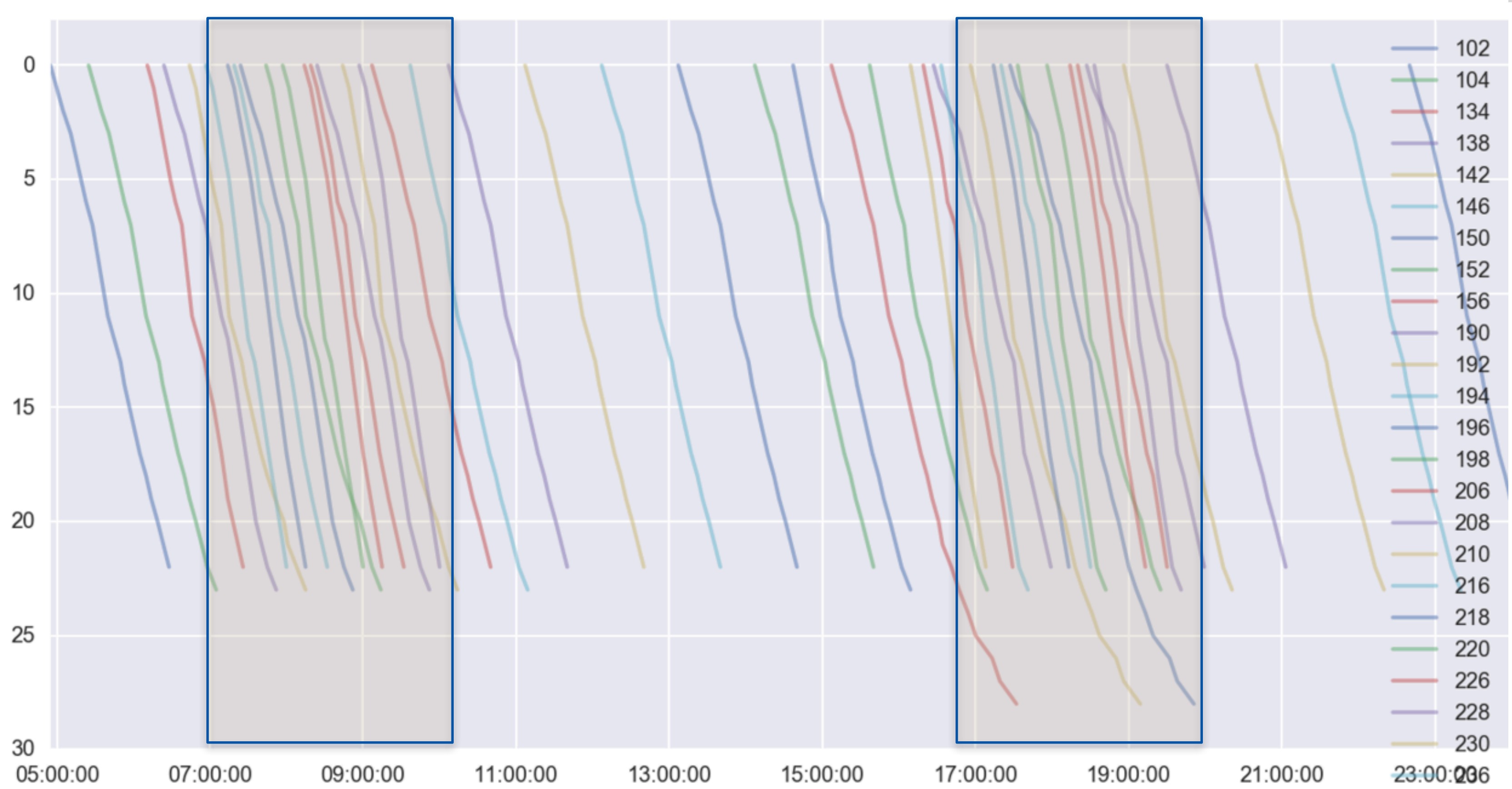
Station matters



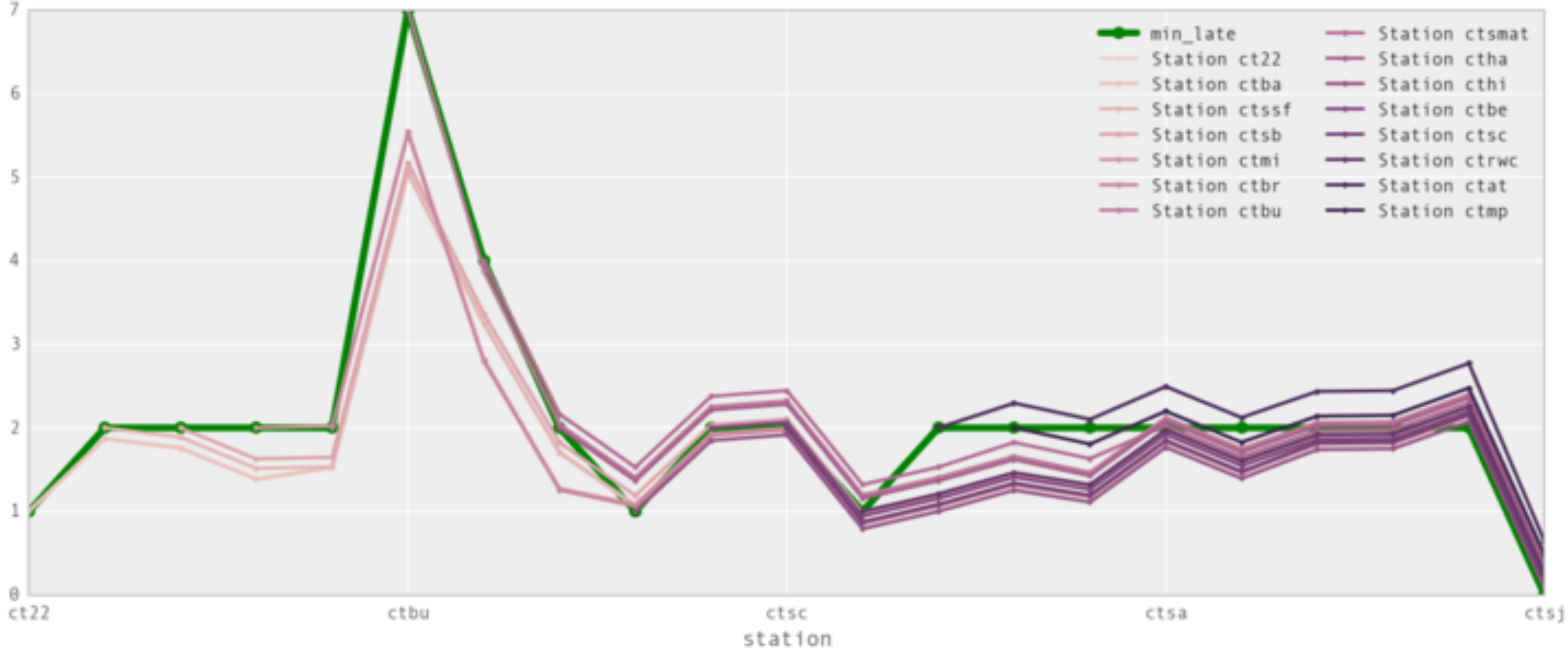
Train Direction also Matters



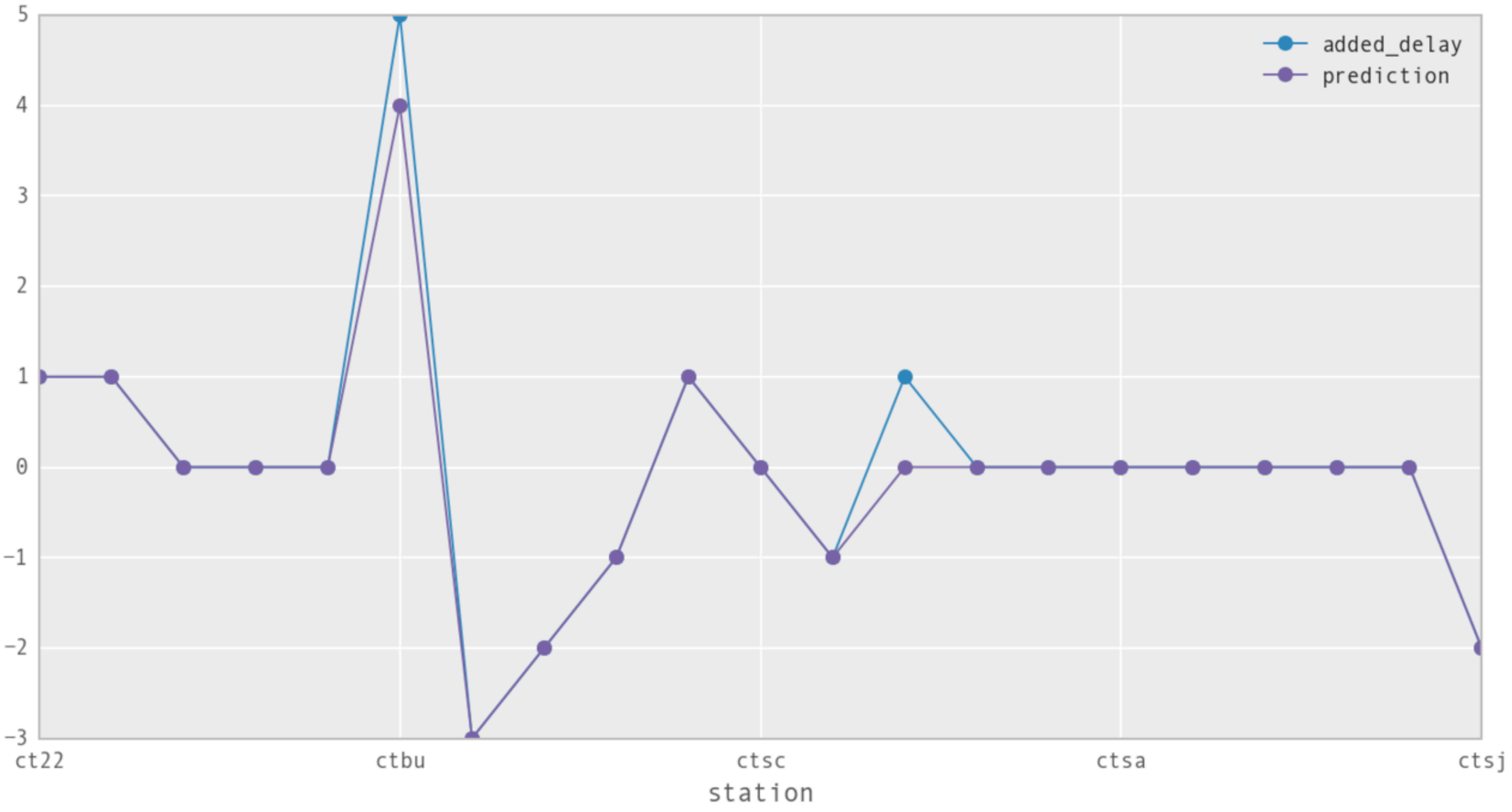
Rush-hours Matters and Other Trains Matter



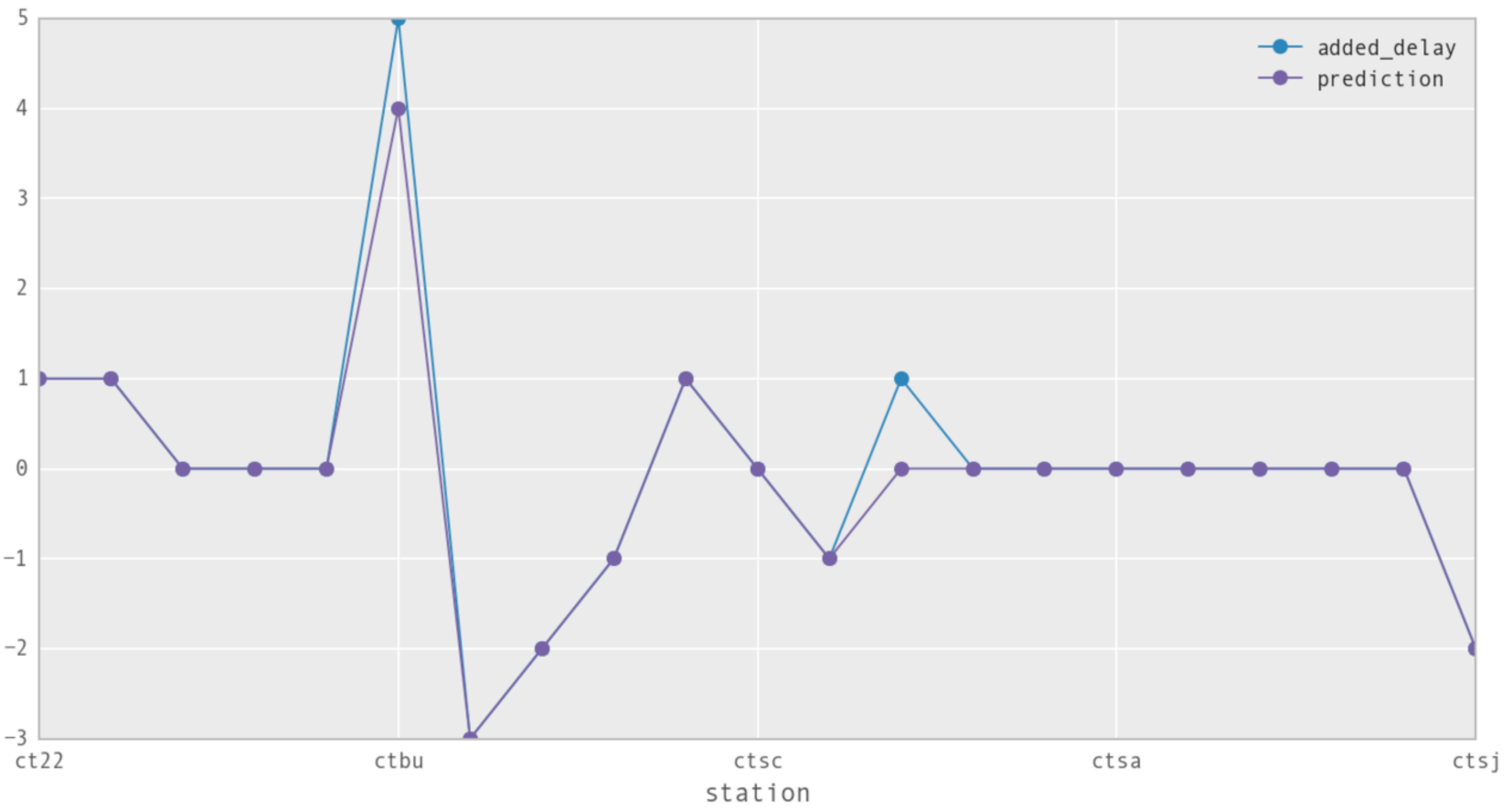
Preliminary Results from RF Regression



Preliminary Results from RF Regression



Preliminary Results from RF Regression

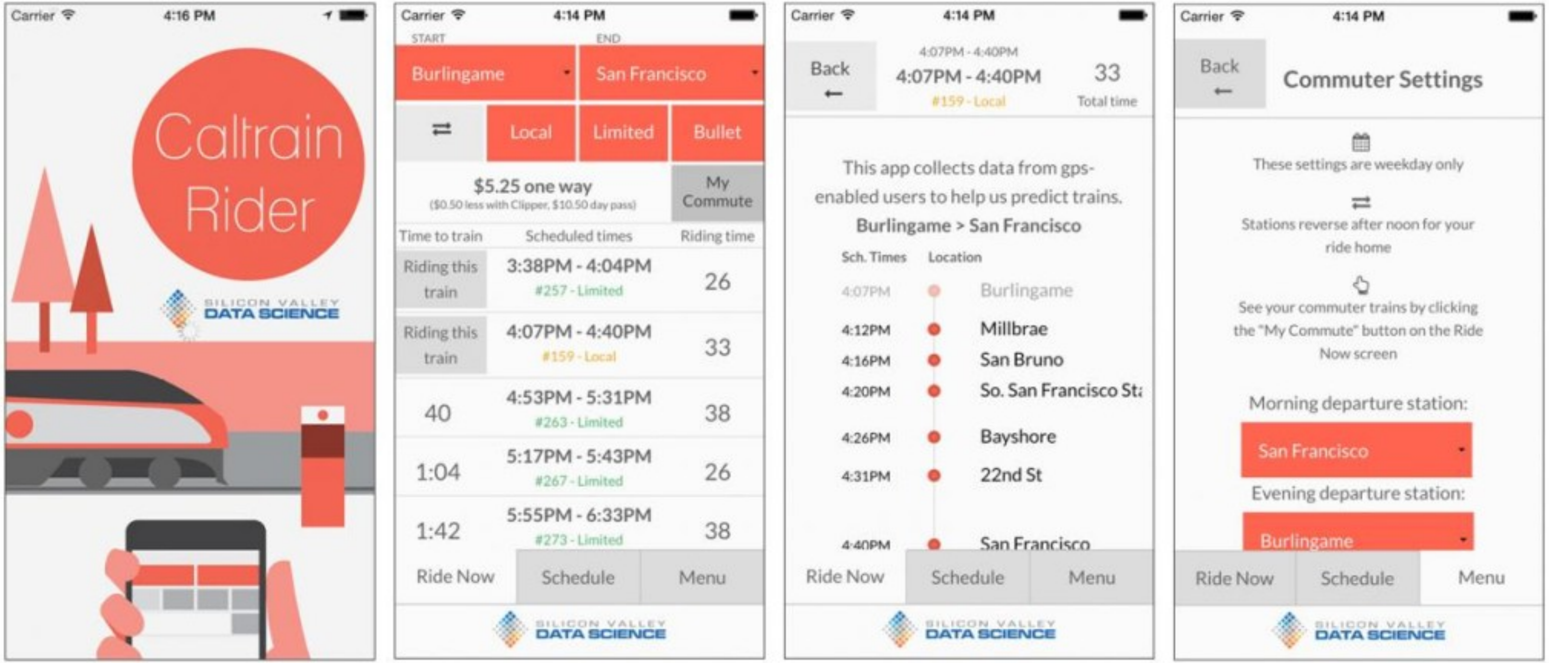


The data science

Train Arrival Time Prediction (Preliminary)

Caltrain Rider

- SVDS has made a Caltrain mobile application: Caltrain Rider
- It provides static schedules for the riders
- It has not incorporated the real-time arrival prediction but collected anonymized riders' GPS data



1. Continue to develop the train delay prediction model using Caltrain API data
2. Continue to collect more API data and combine with data such as weather and events (Giants games, Shark games, etc)
3. Exploit information from GPS data obtained from riders
4. Exploit information from real-time train arrival prediction provided by Caltrain API
5. Update the app to incorporate sentiment and prediction information