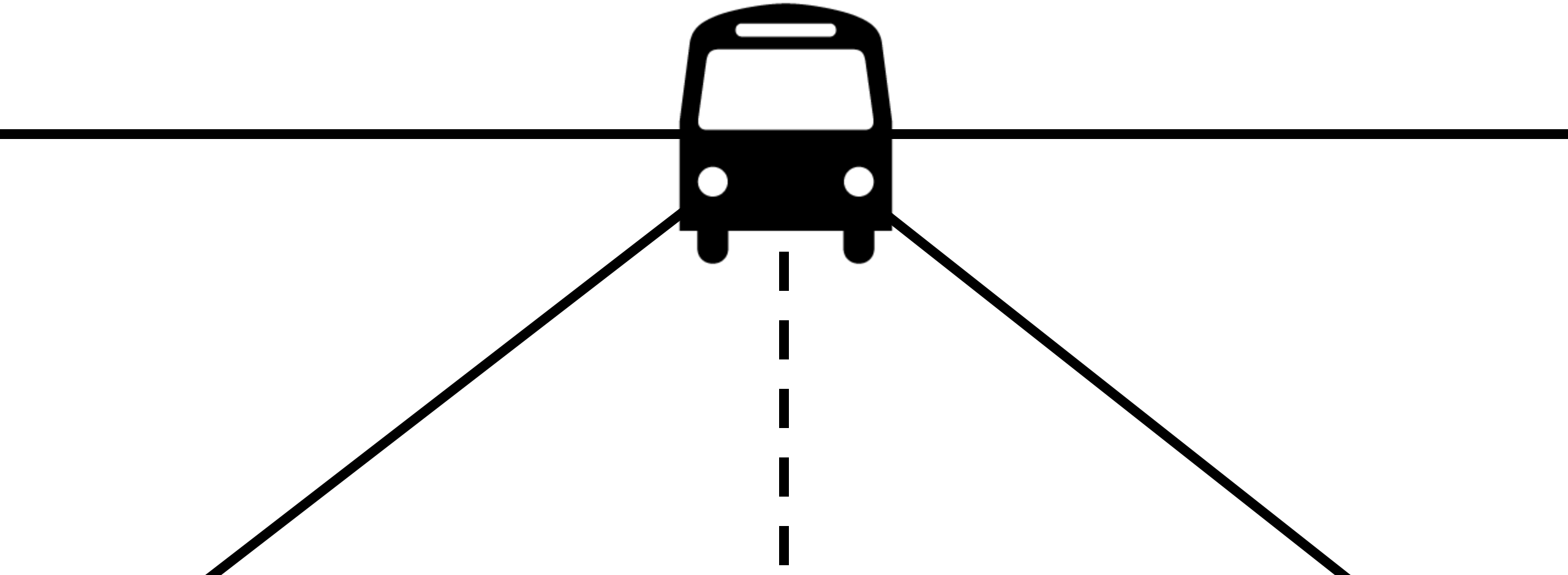


Using Machine Learning to Optimize Public Transit Scheduling and Efficiency

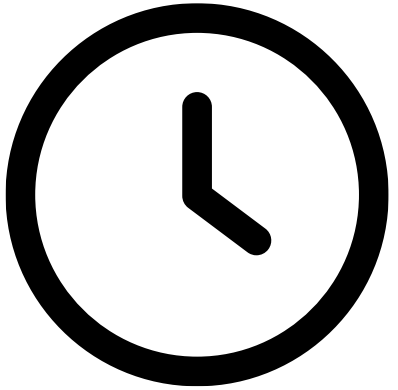
Ningsong Shen
Supervised by Professor Mike Katchabaw
Department of Computer Science, Western University

A Final Presentation

Acknowledgement

- Professor Mike Katchabaw
  - For giving me directions when I got lost
- Course Coordinators
  - So that I could take this great course
- Western Admissions
  - For accepting me to this great school

Background

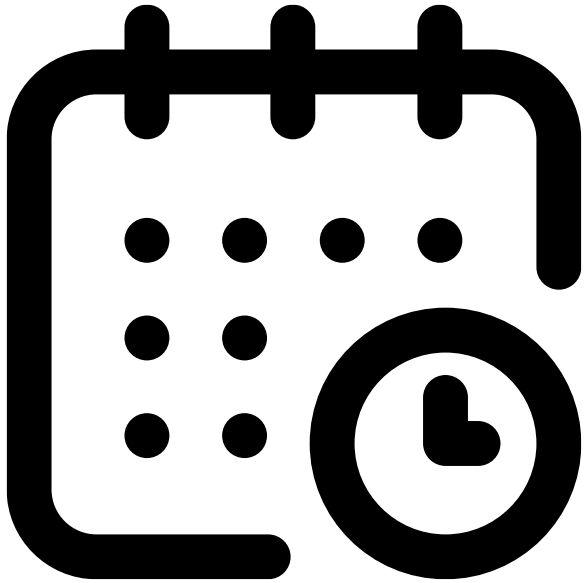


Wasting time

Anxious and unhappy

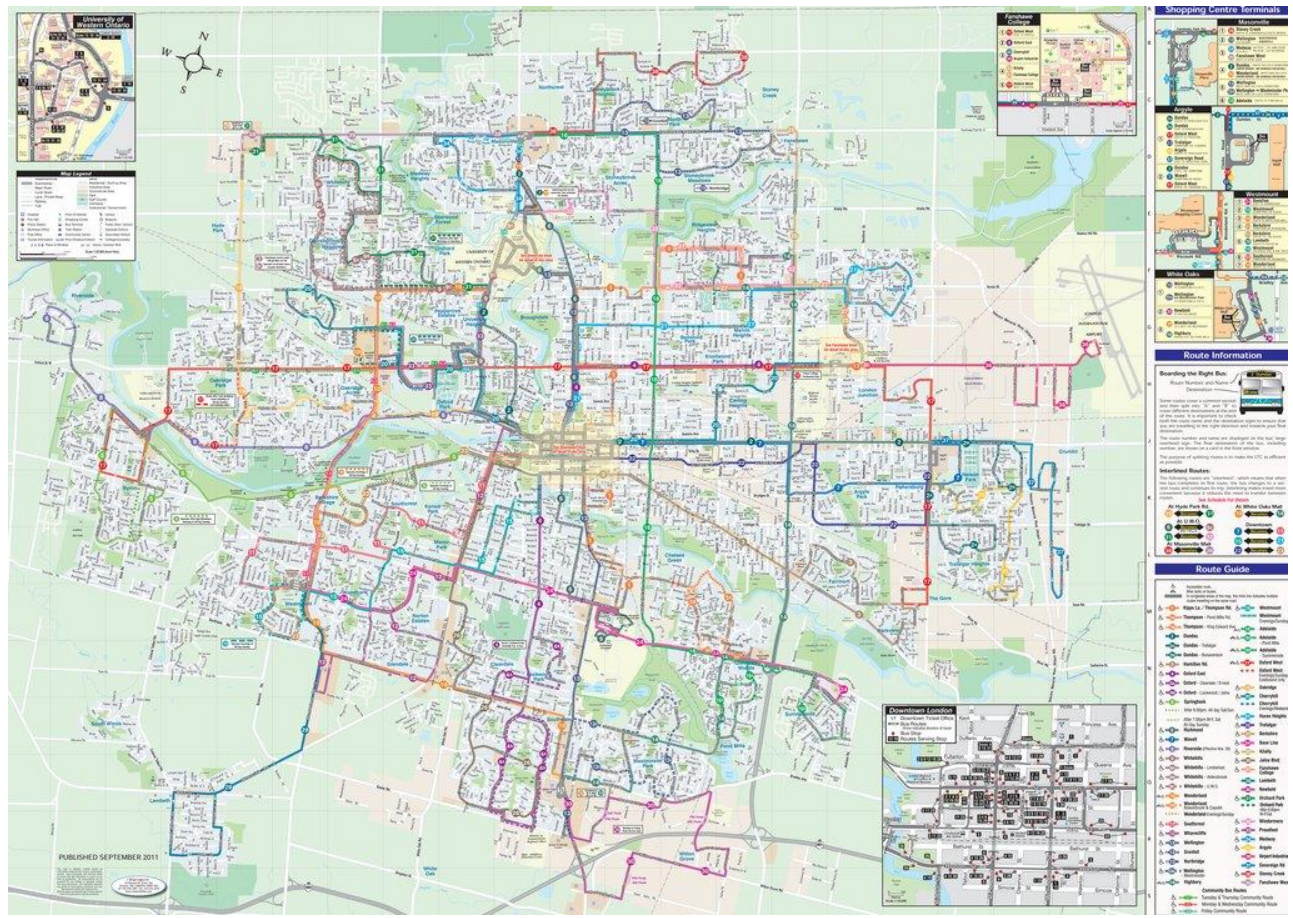
Lose riders

Current Fixes



Predicting

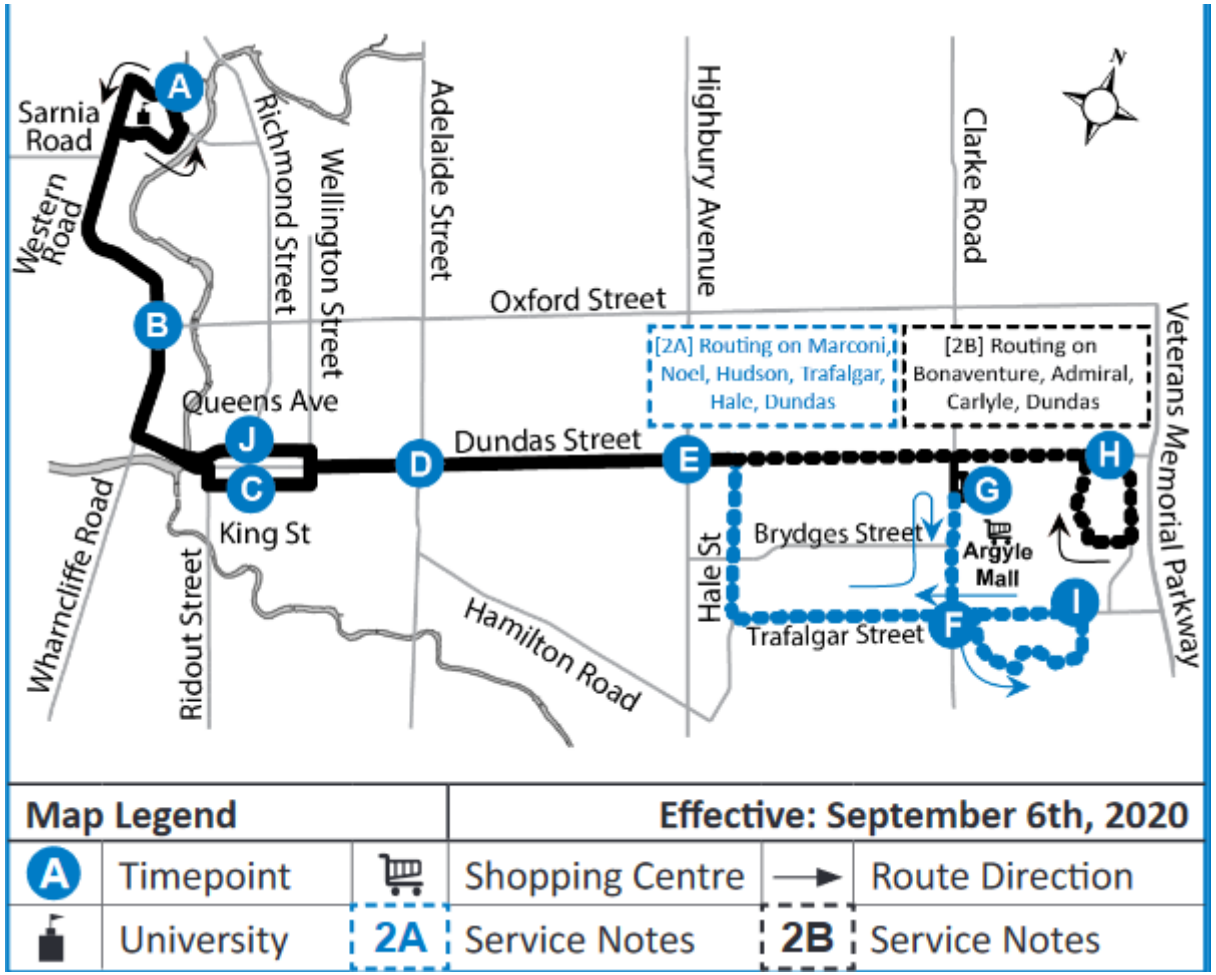
DELAYS


University of Western Ontario
Map Legend
Fanshawe College
Downtown London
PUBLISHED SEPTEMBER 2011
Shopping Centre Terminals
Masonville
Argyle
Westmount
White Oaks
Route Information
Boarding the Right Bus:
Route Numbers and Name Destination
Interlined Routes:
Route Guide

Operational reasons

- Ensure transit personnel maximum working hours
- Coordinate bus layovers and maintenance schedules
- Reduce terminal crowding and keep connections
- Reduce bunching on high frequency routes
- Monitor fuel requirements
- Posted trip lengths
- Other city planning based on travel times

Current Methods: Timepoints



Current Methods: Timepoints

ROUTE 2 - MONDAY TO FRIDAY

EASTBOUND								WESTBOUND							
Natural Science	Wharncliffe at Oxford	King at Richmond	Dundas at Adelaide	Dundas at Highbury	Trafalgar at Clarke	Argyle Mall	Argyle Mall	Bonaventure at Dundas	Trafalgar at Hudson	Argyle Mall	Dundas at Highbury	Dundas at Adelaide	Queens at Richmond	Warncliffe at Oxford	Natural Science
A	B	C	D	E	F	G	G	H	I	G	E	D	J	B	A
LVS						ARR	LVS			LVS					ARR
					2A			2B	2A	2B					
-	-	-	-	-	-	-	-	-	5:50	-	6:03	6:10	6:15	6:21	6:29
-	-	-	-	-	-	5:46	5:46	5:50	-	6:10	6:18	6:25	6:30	6:36	6:44
-	-	-	-	-	6:20	6:24	6:24	-	6:33	-	6:46	6:53	7:00	7:08	7:19
-	-	6:07	6:13	6:19	-	6:27	6:27	6:31	-	6:53	7:01	7:08	7:15	7:23	7:34
-	-	-	-	-	6:50	6:54	6:54	-	7:03	-	7:16	7:23	7:30	7:38	7:49
6:12	6:22	6:30	6:37	6:44	-	6:52	6:56	7:01	-	7:23	7:31	7:38	7:45	7:53	8:04
6:31	6:41	6:48	6:55	7:02	7:10	7:14	7:23	-	7:33	-	7:46	7:53	8:00	8:08	8:19
6:47	6:57	7:04	7:11	7:18	-	7:26	7:26	7:31	-	7:53	8:01	8:08	8:15	8:23	8:34

Current Methods: Timepoints

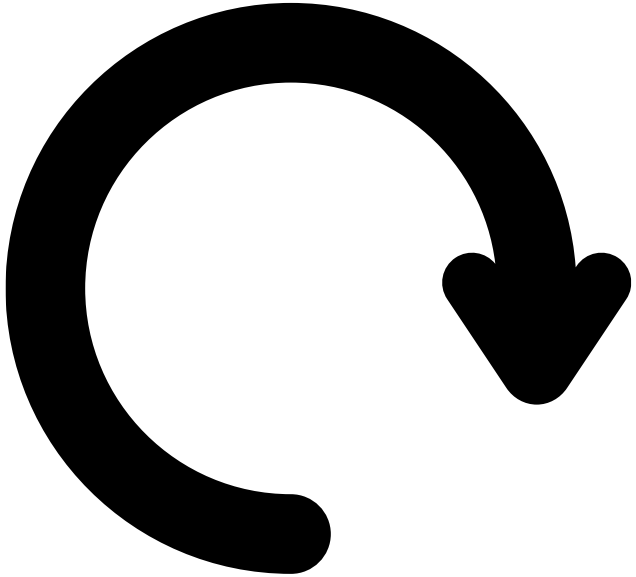
Hello Ningsong,

Thank you for your email inquiring about historical GTFS data.

I was wondering if you could provide some more information with regard to what you are looking for. We do not have records of historical GTFS data, but do keep information on schedule adherence, i.e. if the bus departed a Timepoit early or late and by how much from the schedule. This would be in a CSV format. Additionally, how far back for historical data are you looking?

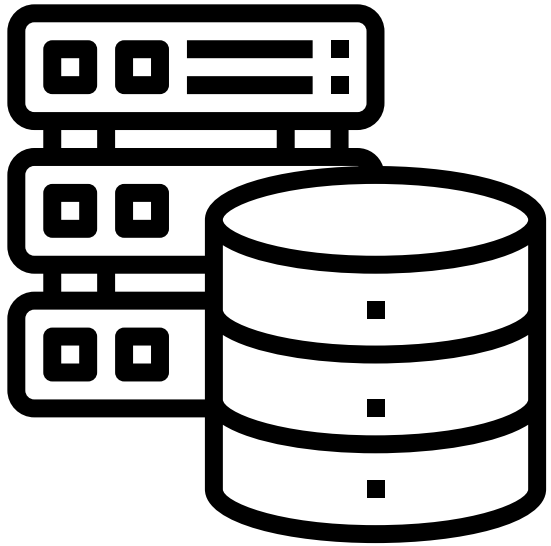
Katie Burns
Director of Planning
London Transit Commission

Current Methods: Manual Adjustment

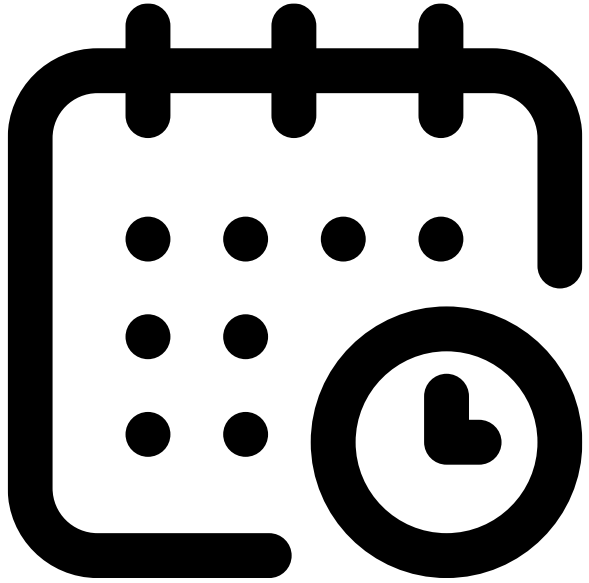


Project Rationale (the WHY)

Widely used techniques



Data and storage is cheap



Dynamic Schedules

- Do these things
- Improve transit schedules
- Increase system efficiency

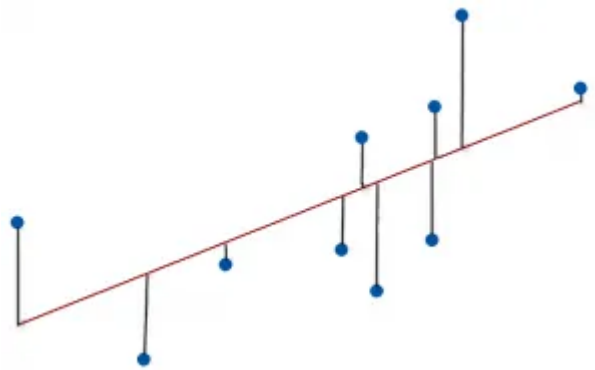
Using Machine Learning to Optimize Public Transit Scheduling and Efficiency

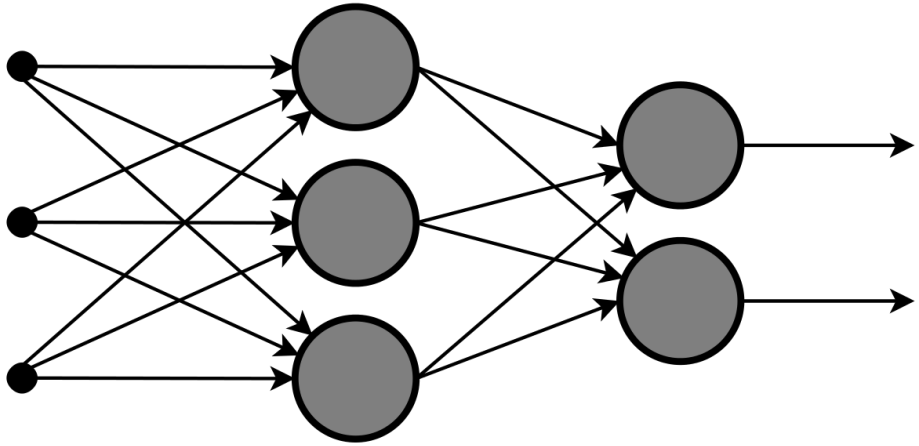
Ningsong Shen
Supervised by Professor Mike Katchabaw
Department of Computer Science, Western University

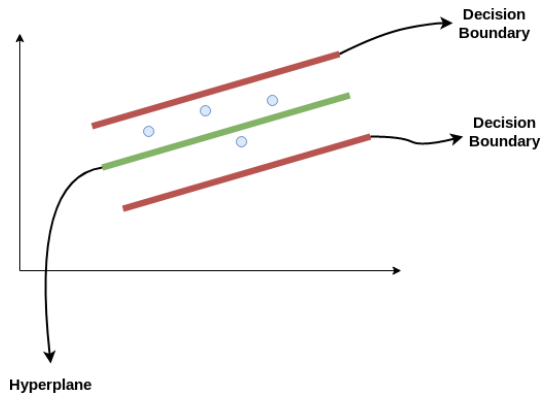
A Final Presentation

Project Goals

Try a bunch of techniques to see which ones work well

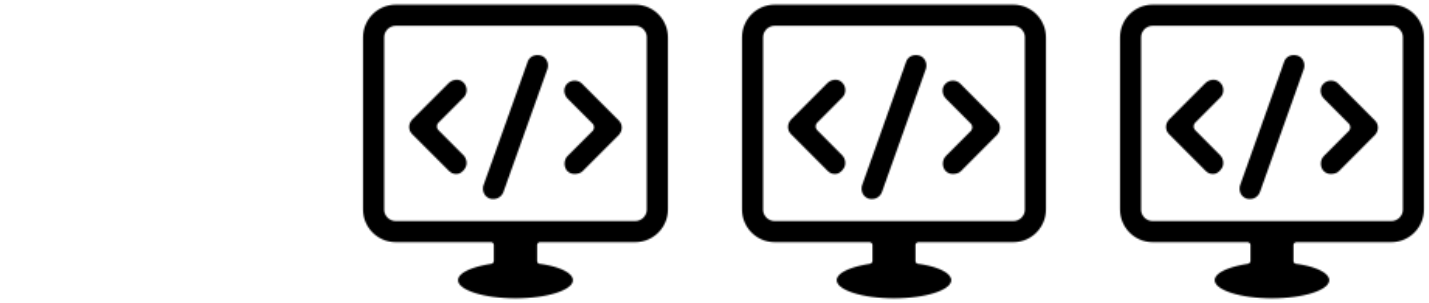


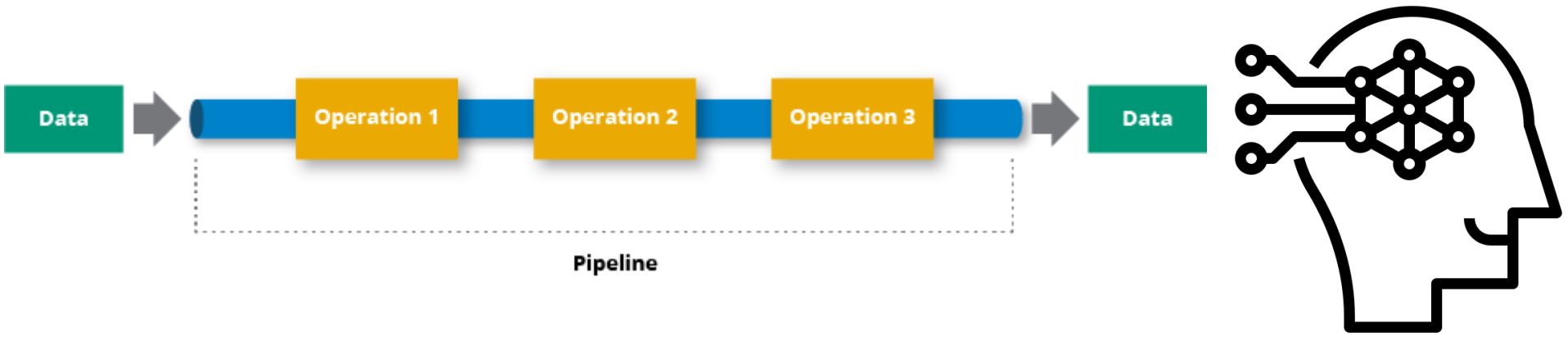




See if we can build a system to improve schedules with a continuous stream of information

Learn about:
system design and machine learning





Literature Review

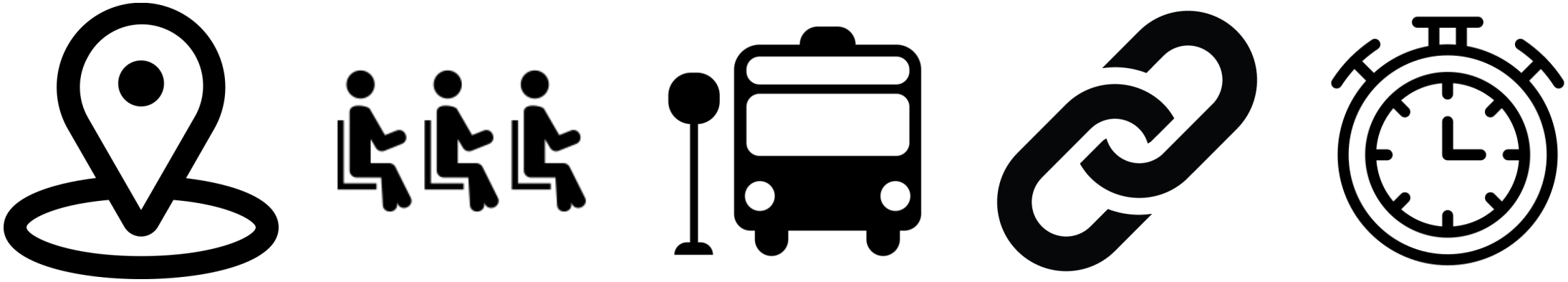
What can I build off of?

List of Consulted Sources

A Dynamic Bus-Arrival Time Prediction Model Based on APC Data
A multiobjective single bus corridor scheduling using machine learning-based predictive models
A prescription for transit arrival/departure prediction using automatic vehicle location data
All about General Transit Feed Specification
Automated Setting of Bus Schedule Coverage Using Unsupervised Machine Learning
Bus arrival time prediction at bus stop with multiple routes
Bus Arrival Time Prediction Using Support Vector Machines
Bus arrival time prediction with real-time and historic data
DelayRadar: A Multivariate Predictive Model for Transit Systems
Dynamic Bus Arrival Time Prediction with Artificial Neural Networks
Dynamic Prediction of traffic volume through Kalman filtering theory
Estimation of Bus Arrival Times using APC Data
Experimental Study of Real-Time Bus Arrival Time Prediction with GPS Data
Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS Data
Multi-Output Deep Learning for Bus Arrival Time Predictions
Prediction Model of Bus Arrival and Departure Times Using AVL and APC Data
Prediction of Bus Arrival Times at Bus Stop
Prediction of Bus Travel Time Using Random Forests Based on Near neighbors
Real-time road traffic prediction with spatio-temporal correlations
Traffic prediction using multivariate nonparametric regression
Transit network design and scheduling: A global review
Travel time prediction model for regional bus transit
Validating the coverage of bus schedules: A Machine Learning approach
Vehicle Scheduling of an Urban Bus Line via an Improved Multiobjective Genetic Algorithm
Urban Bus Arrival Time Prediction: A Review of Computational Models
How long to wait? Predicting bus arrival time with mobile phone based participatory sensing

Major Findings Overall

- Multiple Data Sources Used



- Multiple Techniques Used
  - CNN, Regression, SVM, Kalman filters, etc.

Major things I did not find

- Use of large amounts of data
  - Many used simulated data
  - Many did not look at entire system data
  - Significant usage of open data
- Focus on the operator and system side
  - Main focus is on predicting delays

How this ties into our project

- Focus on a system
- Focus on using lots of data
- Try it out in a real-world setting

Hypothesis

- “evaluate a variety of predictive models and algorithms to optimize transit schedules on a more frequent basis”

- “if it is possible to accurately predict bus arrival times with continuous streams of real-time information, this will show that existing techniques are sufficient to provide enhanced scheduling methods”

Methods

Lettuce now talk about our process

Data Collection

The Setting

All routes, note that many studies only looked at one bus line

The Data Source

GTFS

City of London Open Data

API

GTFS (Google Transit) schedule data feed

http://www.londontransit.ca/gtfsfeed/google_transit.zip

GTFS-RT (realtime) links:

Trip Updates	gtfs.ltconline.ca/TripUpdate/TripUpdates.pb
	gtfs.ltconline.ca/TripUpdate/TripUpdates.json
Alerts	gtfs.ltconline.ca/Alert/Alerts.pb
	gtfs.ltconline.ca/Alert/Alerts.json
Vehicle Positions	gtfs.ltconline.ca/Vehicle/VehiclePositions.pb
	gtfs.ltconline.ca/Vehicle/VehiclePositions.json

GIS Spatial Data

September 2019	Bus Routes	
September 2018	Bus Routes	Bus Stops

Tabular (MS Excel) Data

September 2018	Bus Stop Inventory – LTC stops info including Long/Lat, Associated Routes, shelters, benches. etc.

stop_times.txt

trip_id	arrival_time	departure_time	stop_id	stop_sequence	pickup_type	drop_off_type	timepoint
1342560	6:13:00	6:13:00	KIPPADEL	1	0	1	1
1342560	6:13:51	6:13:51	KIPPBELF	2	0	0	0
1342560	6:14:41	6:14:41	KIPPBARK	3	0	0	0
1342560	6:15:13	6:15:13	KIPPARBO	4	0	0	0
1342560	6:16:04	6:16:04	KIPPBRIA	5	0	0	0
1342560	6:16:43	6:16:43	BRIAMEL2	6	0	0	0
1342560	6:17:17	6:17:17	BRIAMEL3	7	0	0	0
1342560	6:17:35	6:17:35	BRIAHURO	8	0	0	0
1342560	6:17:56	6:17:56	HUROBRI2	9	0	0	0
1342560	6:18:46	6:18:46	HUROBARK	10	0	0	0

stop_times.txt
routes.txt
stops.txt
trips.txt
calendar.txt
calendar_dates.txt
agency.txt
shapes.txt
transfers.txt

GTFS Protobuf Update


```
trip {
  trip_id: "1404519"
  start_time: "13:57:00"
  start_date: "20210322"
  route_id: "20"
  direction_id: 1
}
stop_time_update {
  stop_sequence: 2
  departure {
    time: 1616436252
  }
  stop_id: "BEAVCAP1"
}
stop_time_update {
  stop_sequence: 3
  departure {
    time: 1616436329
  }
  stop_id: "BEAVOAKC"
}
```

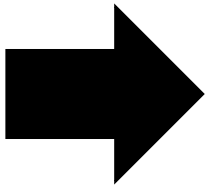
The Data Collector

```
def get_realtime_feed_data():
    """Pull data from GTFS realtime feed and write update to text file

    """
    feed = gtfs_realtime_pb2.FeedMessage()
    response = urllib.request.urlopen('http://gtfs.ltconline.ca/TripUpdate/TripUpdates.pb')
    feed.ParseFromString(response.read())

    x = datetime.datetime.now()
    dt_string = x.strftime("%Y-%m-%d-%H-%M-%S")
    f = open(f'data/{dt_string}.txt', 'a+')
    for entity in feed.entity:
        if entity.HasField('trip_update'):
          f.write(repr(entity.trip_update))

    f.close()
```

The Data Collector

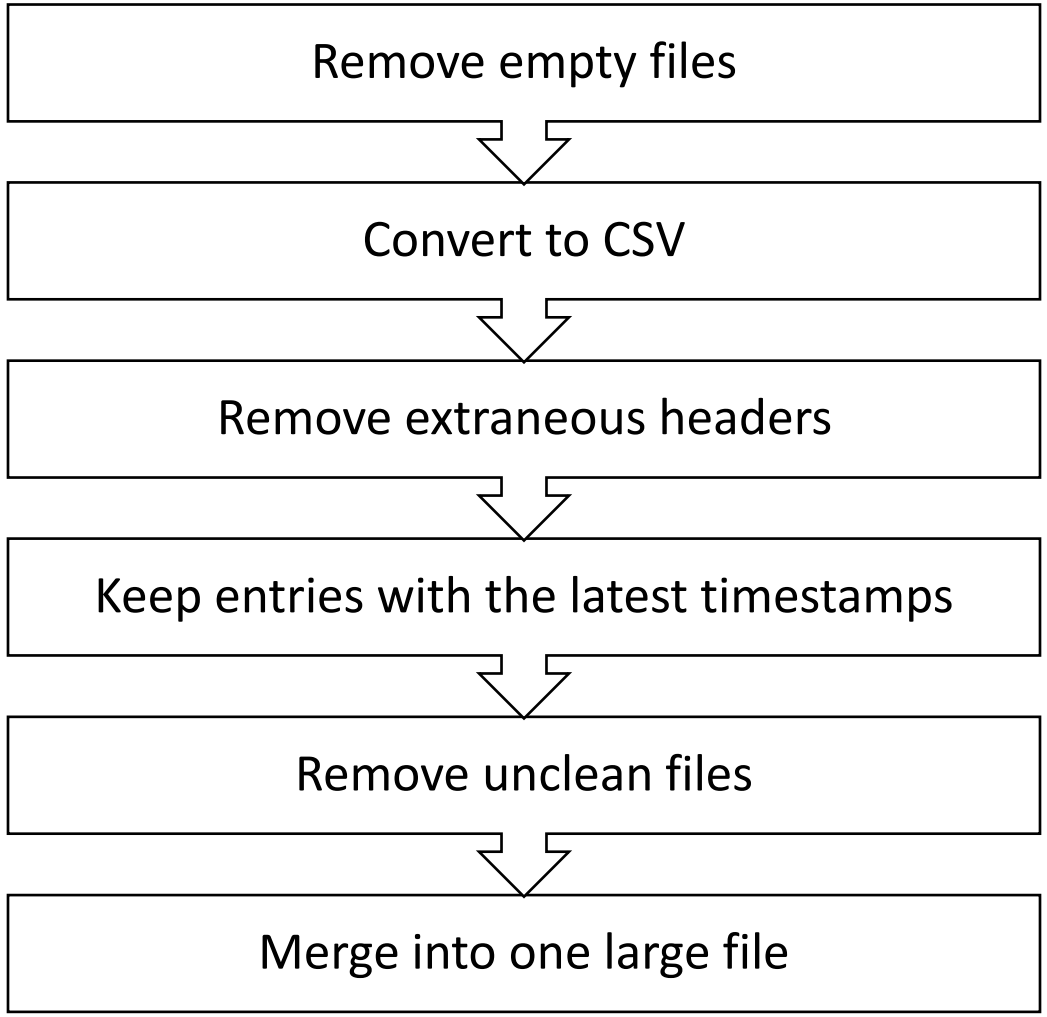
```
# ┌──────────── min (0 - 59)
# │ ┌────────── hour (0 - 23)
# │ │ ┌──────── day of month (1 - 31)
# │ │ │ ┌────── month (1 - 12)
# │ │ │ │ ┌──── day of week (0 - 6) (0 to 6 are Sunday to
# │ │ │ │ │                  Saturday, or use names; 7 is also Sunday)
# │ │ │ │ │
# │ │ │ │ │
# * * * * *  command to execute
```

The Data Collector



- Multiple files
- Repetitious rows
- Redundant updates
- Time zone issues
- Timestamp issues
- Mixed header rows
- Empty rows

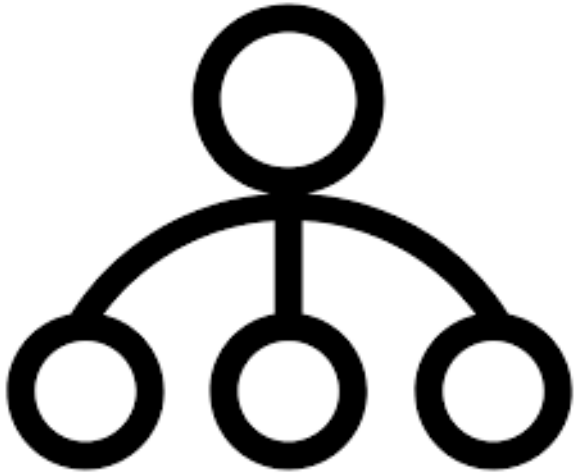
The Data Collector




```
    start = time.time()
    print(f'Start: {time.strftime("%H:%M:%S", time.localtime())}')

    # Remove empty files from download
    print(f'Removing empty files from {LOCAL_DIRECTORY}...')
    total_removed = remove_empty_files(LOCAL_DIRECTORY)
    print(f'Done. {total_removed} files removed. {time.time() -
start} seconds elapsed.')

    # Convert all files to CSV
    print('Converting files to CSV...')
    total_converted = convert_folder(LOCAL_DIRECTORY, OUTPUT_DIRECTORY)
    print(f'Done. {total_converted} files converted. {time.time() -
start} seconds elapsed.')

    # NOT NEEDED RIGHT NOW: Remove header from the files
    # remove_header(LOCAL_DIRECTORY, OUTPUT_DIRECTORY)

    # Keep only the latest timestamps
    print('Cleaning up data...')
    total_processed = keep_latest(OUTPUT_DIRECTORY)
    print(f'Done. {total_processed} files processed. {time.time() -
start} seconds elapsed.')
    # DON'T DELETE THE REMAINING FILE, It's needed for the next pipeline down
load

    # Remove empty files from cleaned
    print(f'Removing empty files from {CLEAN_DIRECTORY}...')
    total_removed = remove_empty_files(CLEAN_DIRECTORY)
    print(f'Done. {total_removed} files removed. {time.time() -
start} seconds elapsed.')

    # Combine all files into one big file
    print(f'Combining files...')
    combine(CLEAN_DIRECTORY, RESULT_DIRECTORY)
    print(f'All processing done. Total {time.time() -
start} seconds elapsed.')
```

The Data Collector

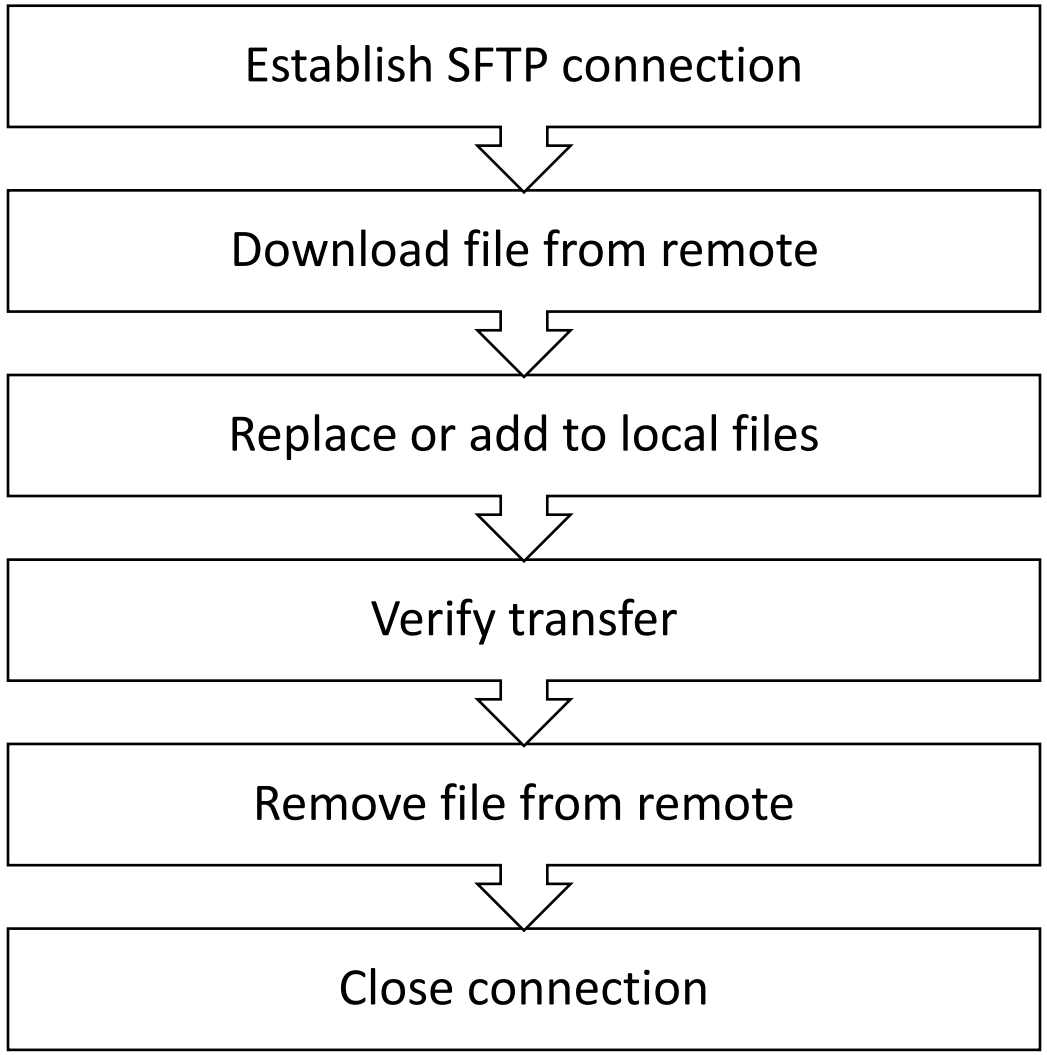
File transfer



Data duplication and version updates

Pipeline automation

The Data Collector




```
start = time.time()
print(f'Start: {time.strftime("%H:%M:%S", time.localtime())}')

ssh_client = paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
key = paramiko.RSAKey.from_private_key_file(PRIVATE_KEY_FILE_LOCATION)
ssh_client.connect(
    hostname=HOSTNAME,
    username=USERNAME,
    pkey=key
)

sftp_client = ssh_client.open_sftp()
file_list = sftp_client.listdir(REMOTE_DIRECTORY)
print(f'Getting files from {REMOTE_DIRECTORY}...')

for file in file_list:
    remote_file_path = REMOTE_DIRECTORY + "/" + file
    local_file_path = LOCAL_DIRECTORY + "/" + file
    sftp_client.get(remote_file_path, local_file_path)
    sftp_client.remove(remote_file_path)

print(f'Done. Total {time.time() - start} seconds elapsed.')
sftp_client.close()
```

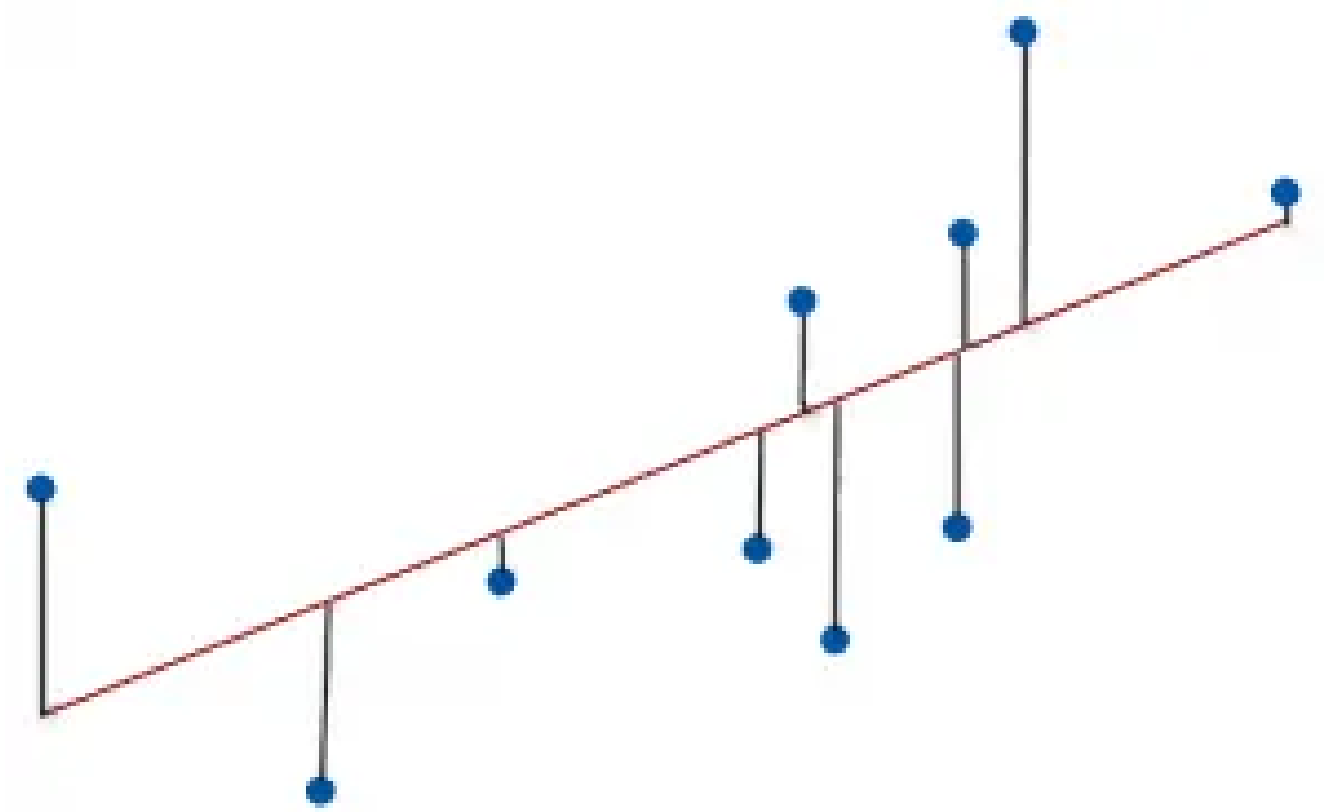
The Data Collector

	trip_id	start_date	route_id	stop_seq	realtime_dep_time	vehicle_id	scheduled_dep_time
2735543	1331895	2020-11-07	6	16	16:48:00	3172	16:48:00
613213	1343233	2020-11-11	10	12	15:45:01	3136	15:45:01
607801	1343225	2020-11-17	10	69	15:40:24	3144	15:40:59
1924122	1339888	2020-11-18	31	4	08:00:52	3350	07:59:52
1841457	1338069	2020-11-09	20	3	07:15:18	3378	07:15:00
2636984	1343356	2020-11-21	10	26	13:00:37	3312	13:02:46
1171153	1345792	2020-11-13	15	33	19:20:41	3172	19:19:41
779022	1340516	2020-11-06	34	9	16:44:00	3335	16:43:00
639682	1345017	2020-11-09	07	3	15:49:20	3147	15:49:35
1593704	1345753	2020-11-24	15	20	23:23:26	3177	23:23:41

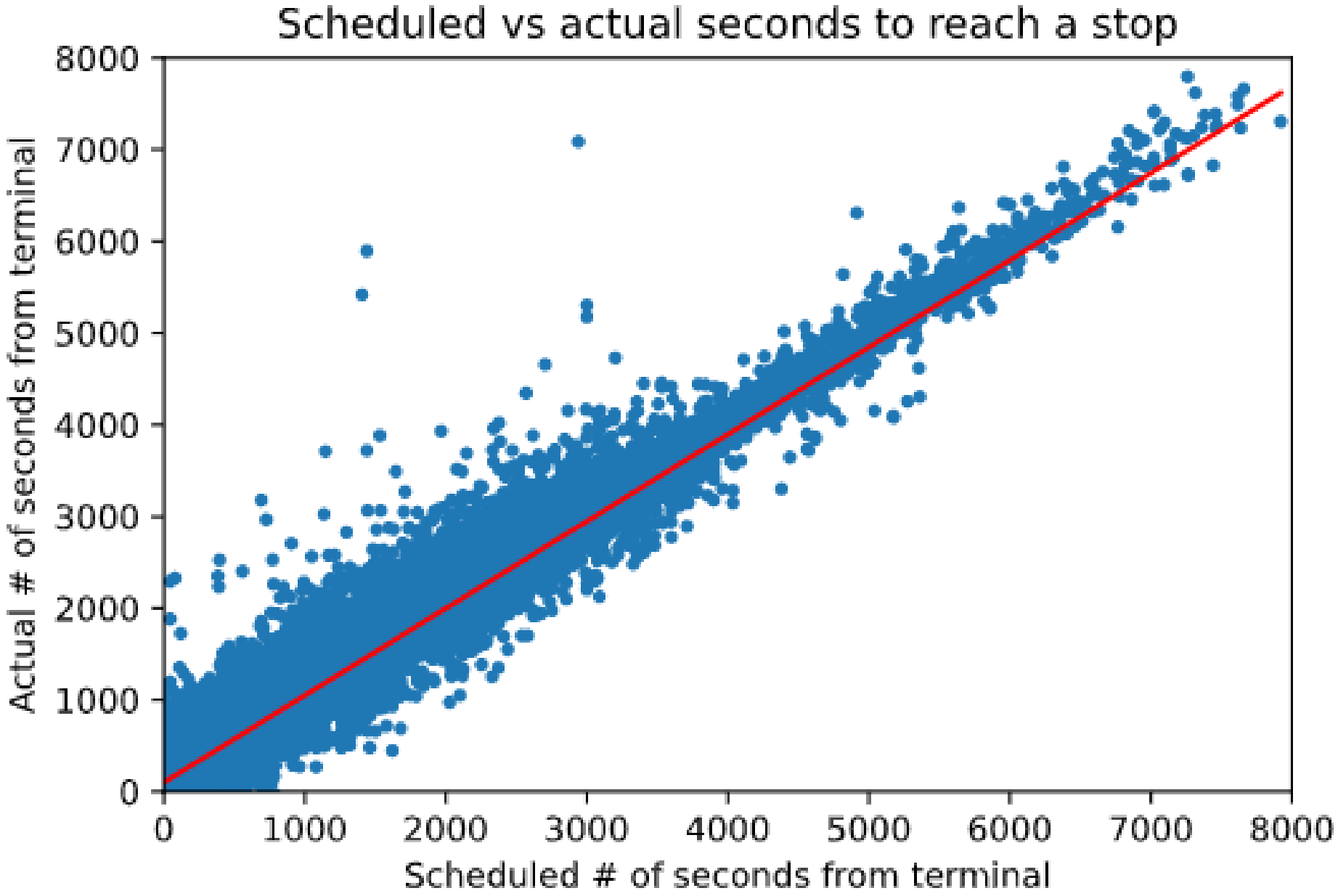
Data Analysis

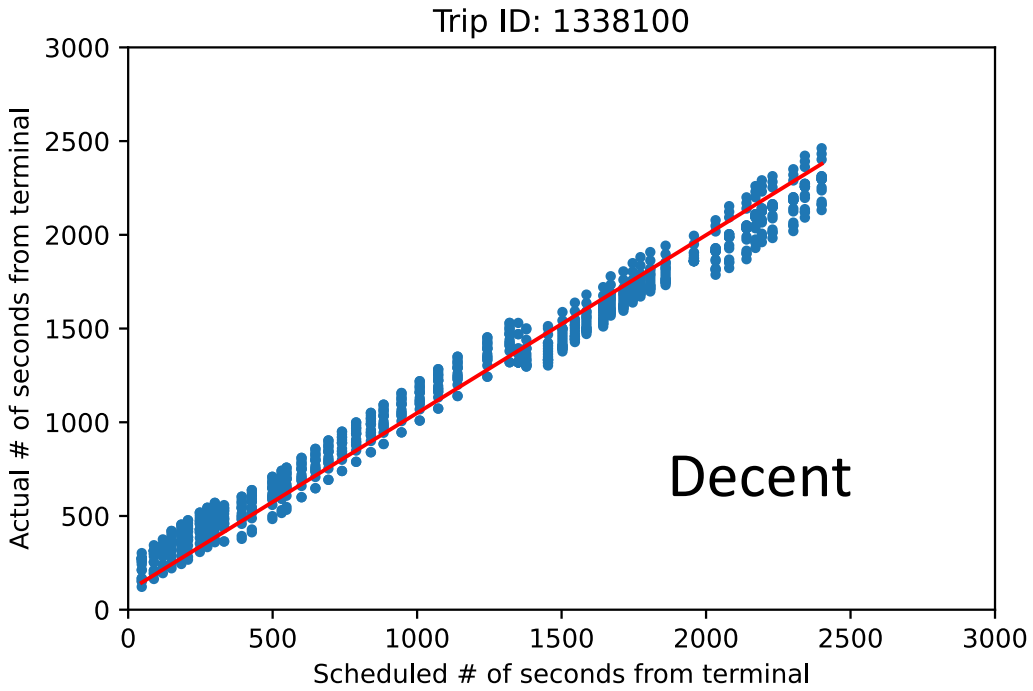
Tools

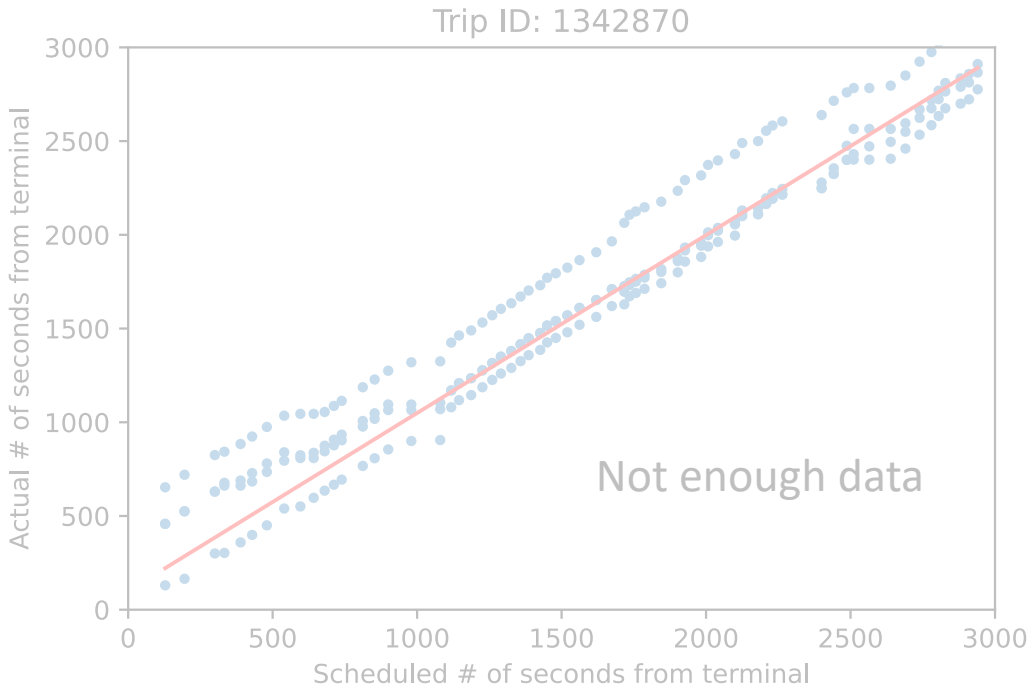
OLS Regression

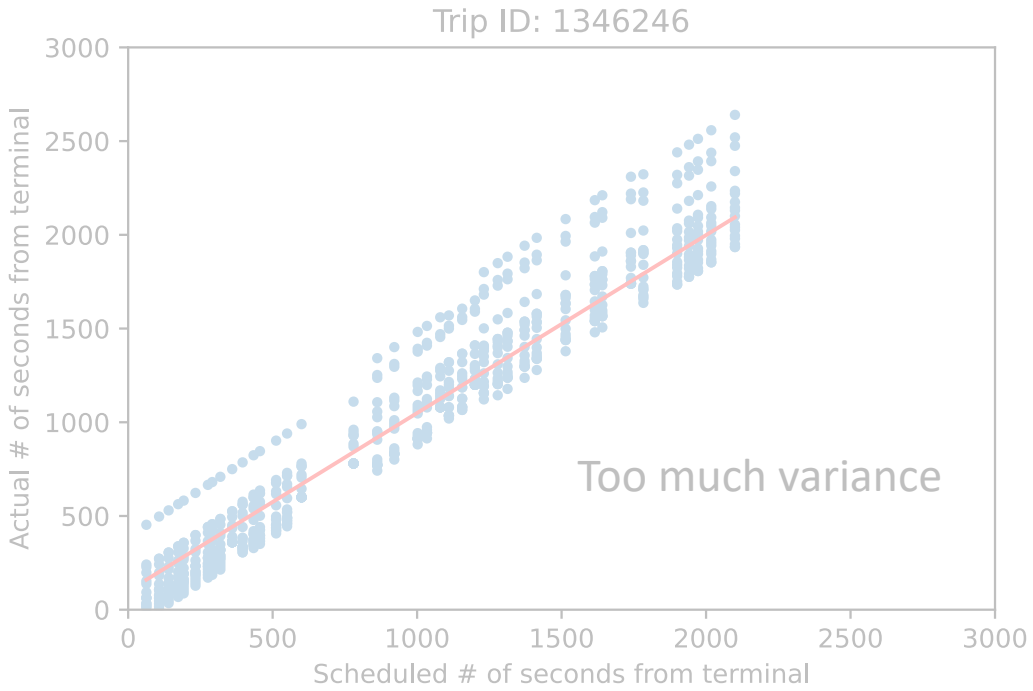


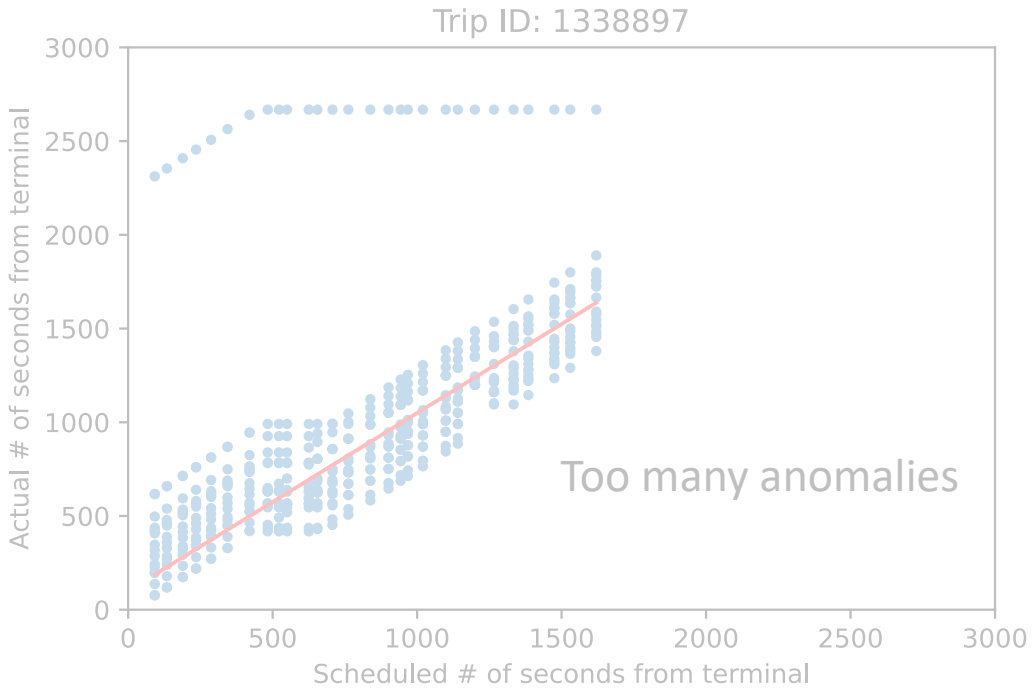
All Lines


Scheduled vs actual seconds to reach a stop
Actual # of seconds from terminal
Scheduled # of seconds from terminal
0
1000
2000
3000
4000
5000
6000
7000
8000


Trip ID: 1338100
Actual # of seconds from terminal
Scheduled # of seconds from terminal
0
500
1000
1500
2000
2500
3000
Decent


Trip ID: 1342870
Actual # of seconds from terminal
Scheduled # of seconds from terminal
0
500
1000
1500
2000
2500
3000
Not enough data


Trip ID: 1346246
Actual # of seconds from terminal
Scheduled # of seconds from terminal
0
500
1000
1500
2000
2500
3000
Too much variance

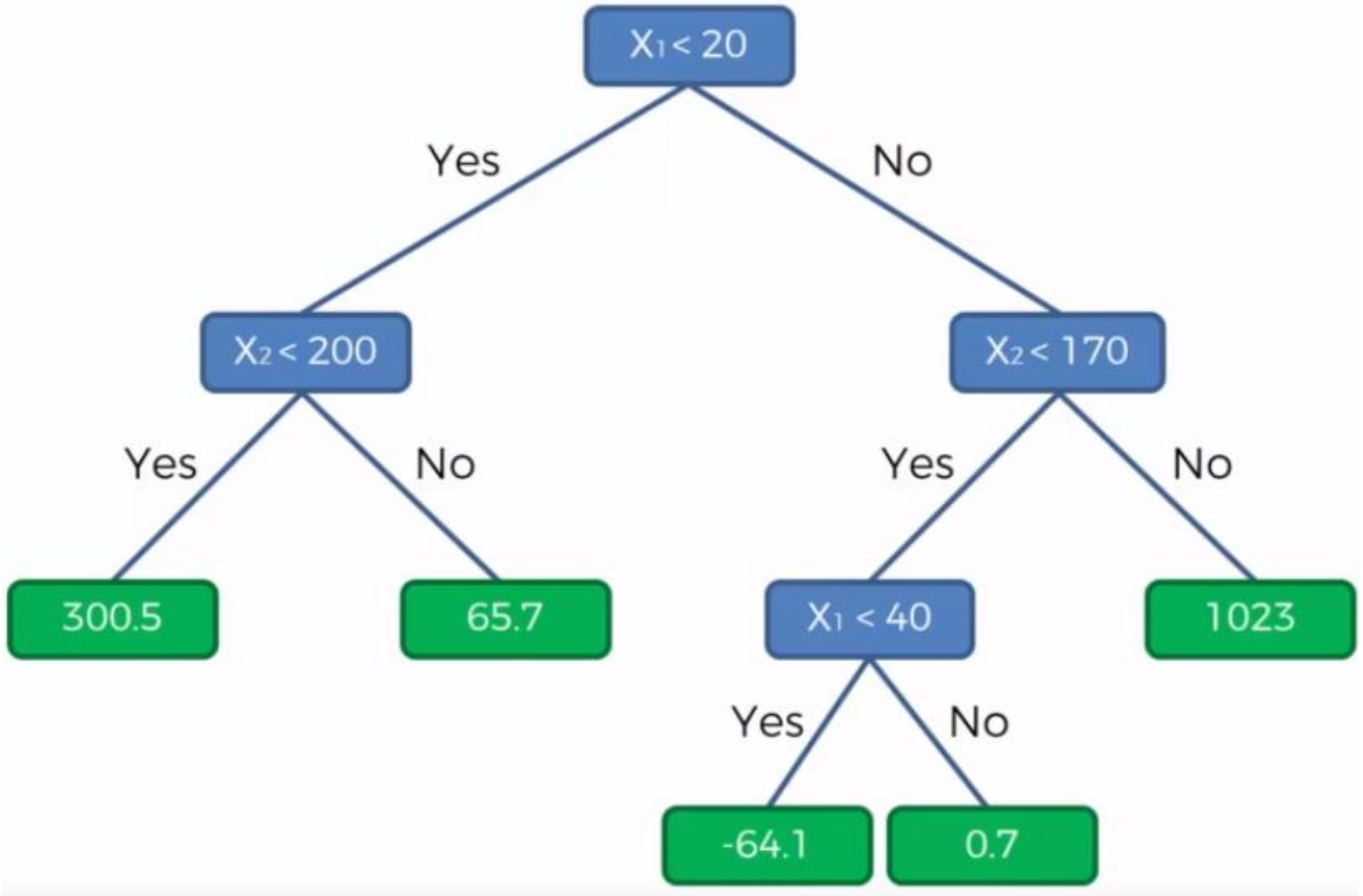

Trip ID: 1338897
Actual # of seconds from terminal
Scheduled # of seconds from terminal
0
500
1000
1500
2000
2500
3000
Too many anomalies

Simple Cases

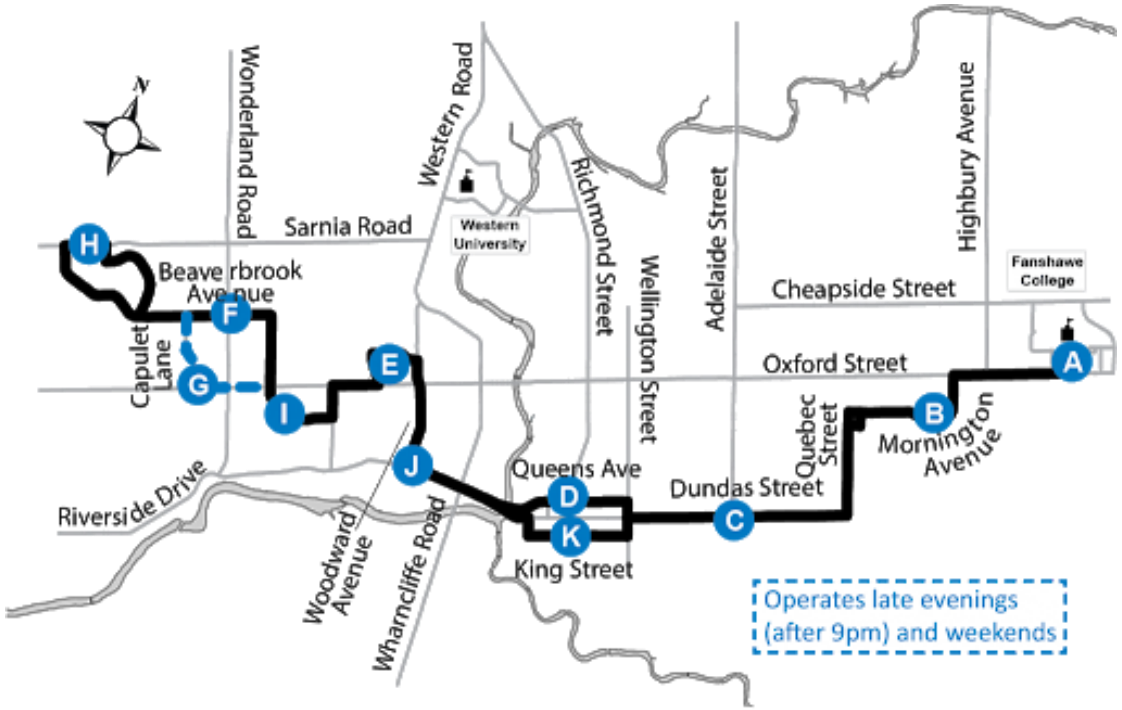
Does not capture variance well

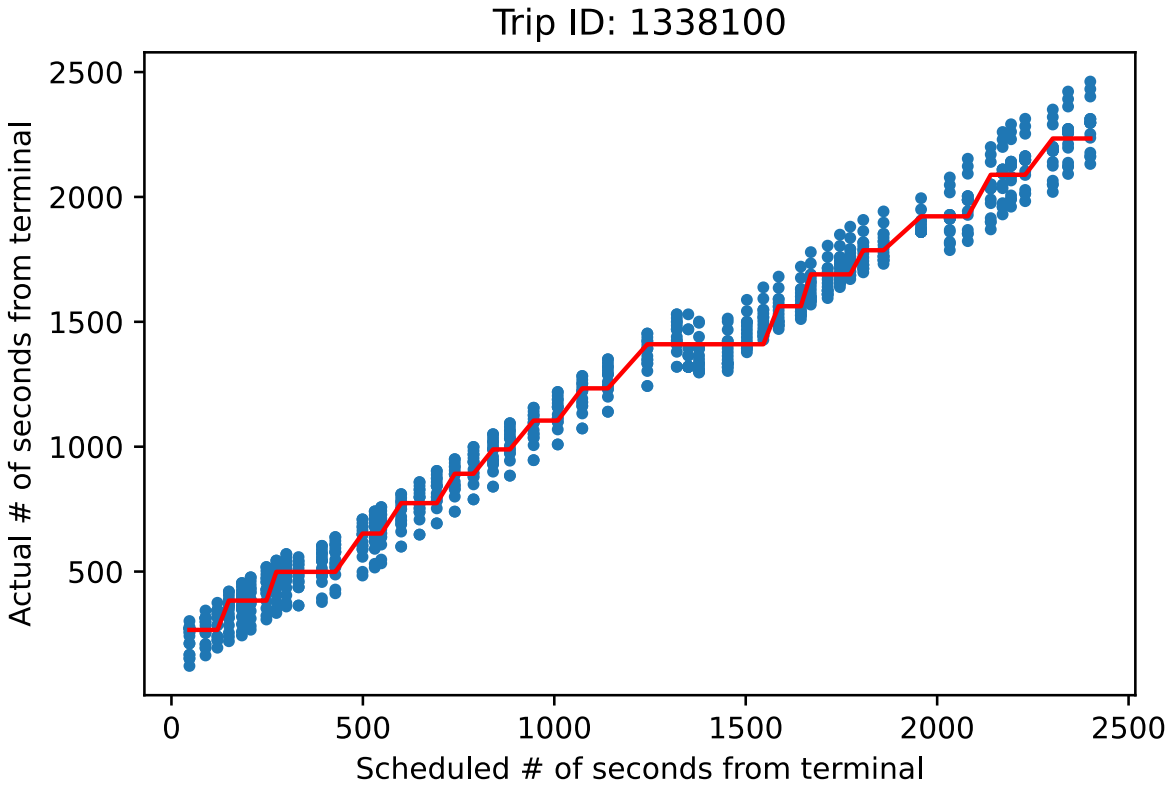
Better than averaging timepoints

Decision Trees



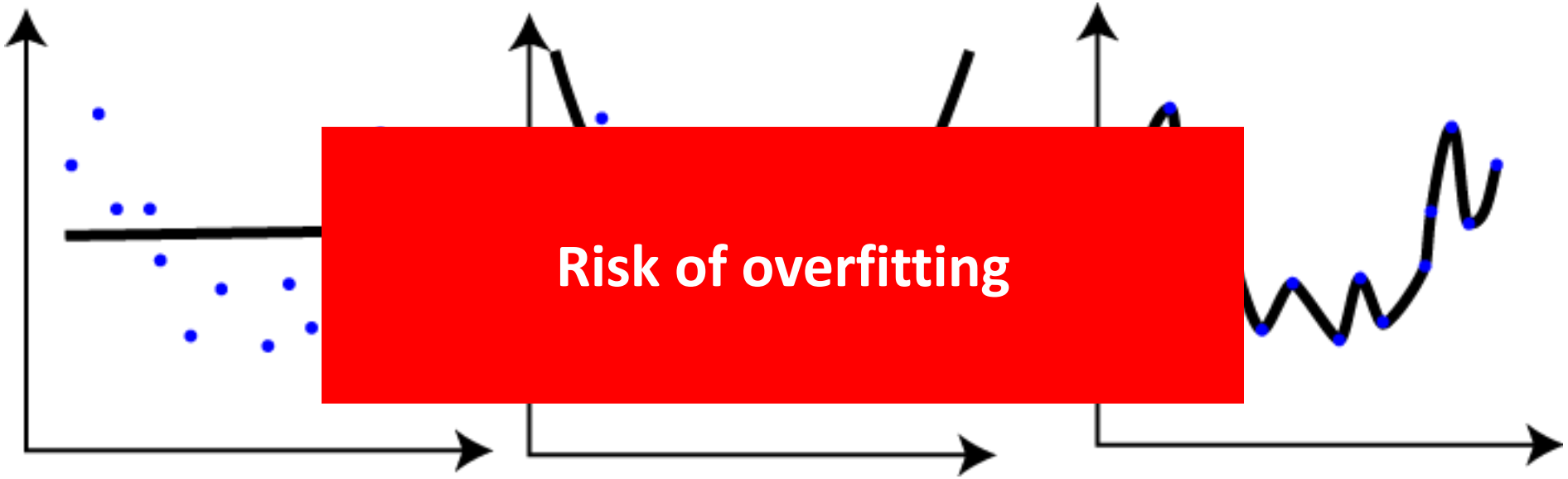
Route 20


N
Wonderland Road
Western Road
Richmond Street
Wellington Street
Adelaide Street
Highbury Avenue
Sarnia Road
Western University
Fanshawe College
Cheapside Street
Beaverbrook Avenue
Capulet Lane
Oxford Street
Quebec Street
Mornington Avenue
Queens Ave
Dundas Street
Riverside Drive
Woodward Avenue
Wharncliffe Road
King Street
H
F
G
I
E
J
D
K
C
B
A
Operates late evenings (after 9pm) and weekends


Trip ID: 1338100
Actual # of seconds from terminal
Scheduled # of seconds from terminal
0
500
1000
1500
2000
2500

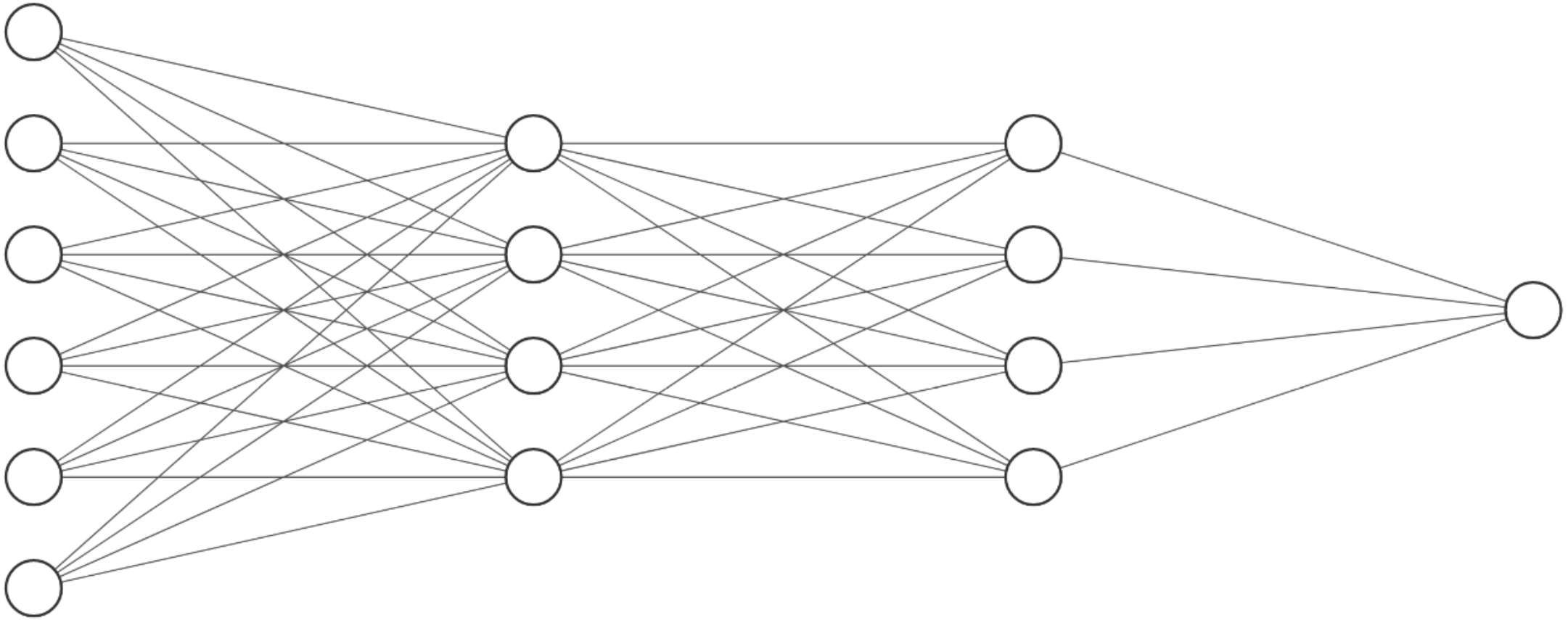
Discussion of Results

Better at adjusting



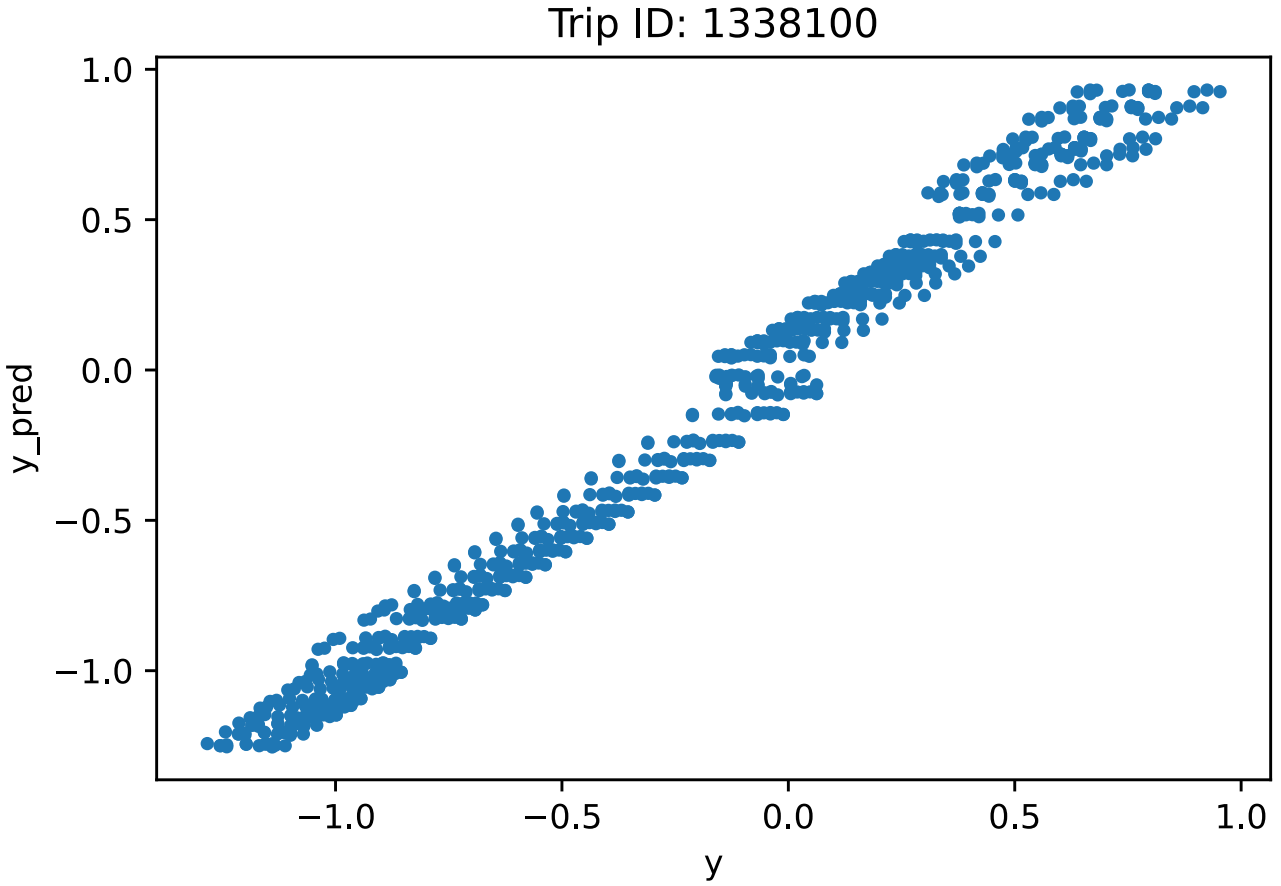
Risk of overfitting

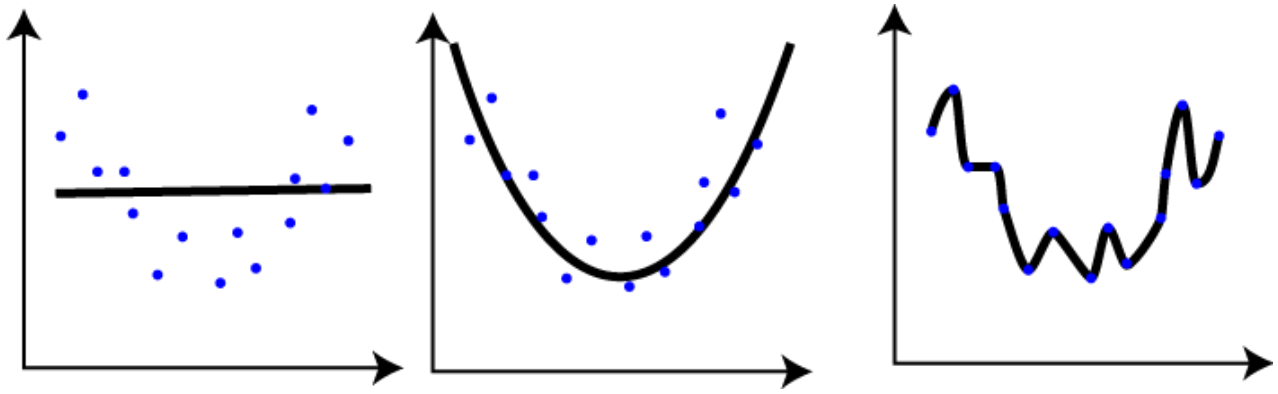
Black Box Algorithms

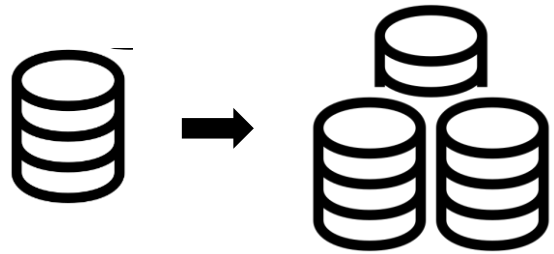


Input Layer $\in \mathbb{R}^6$ Hidden Layer $\in \mathbb{R}^4$ Hidden Layer $\in \mathbb{R}^4$ Output Layer $\in \mathbb{R}^1$

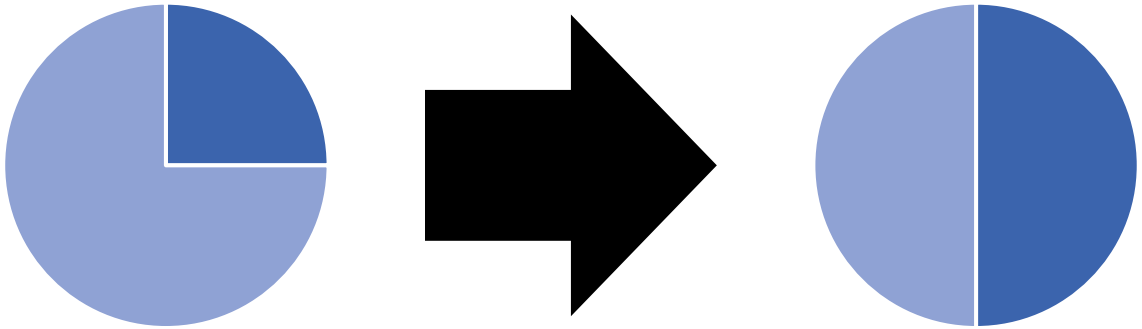
Train set, then test set

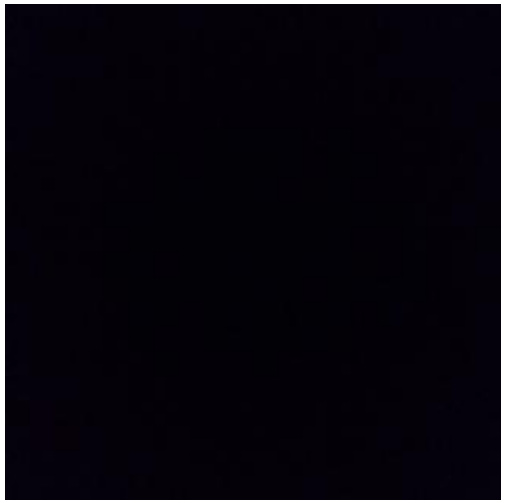






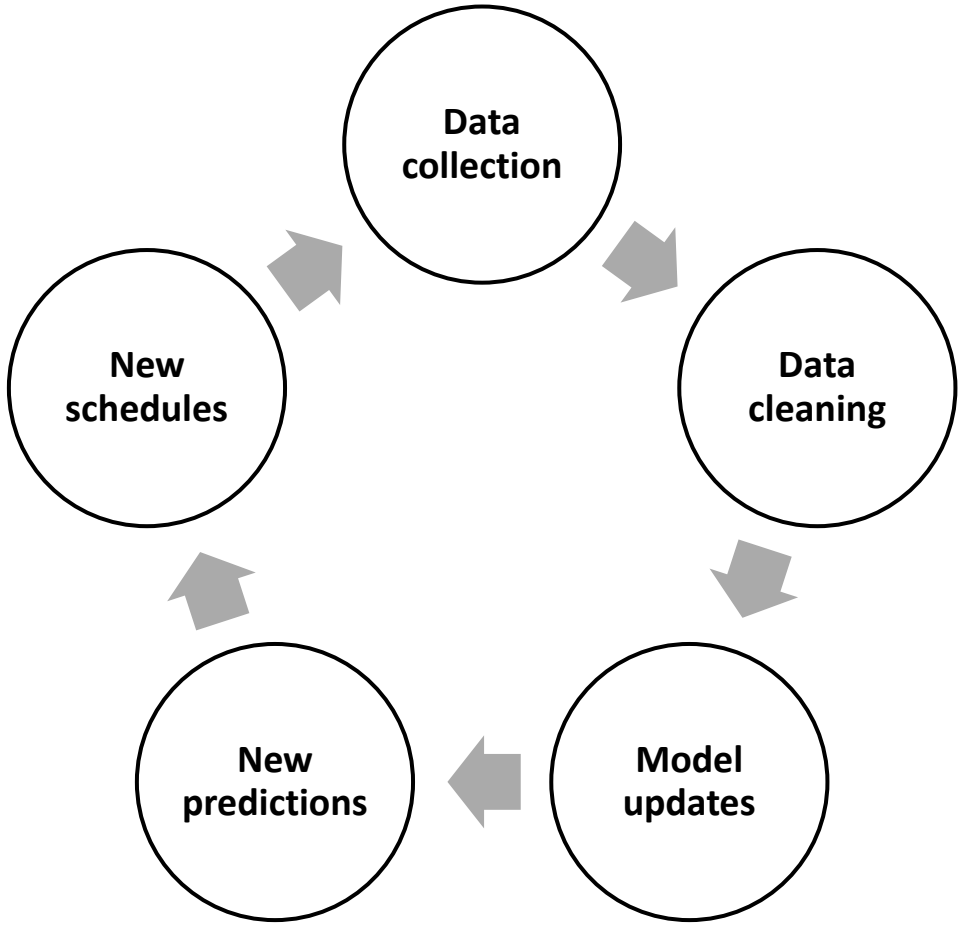
Neural Network Effectiveness





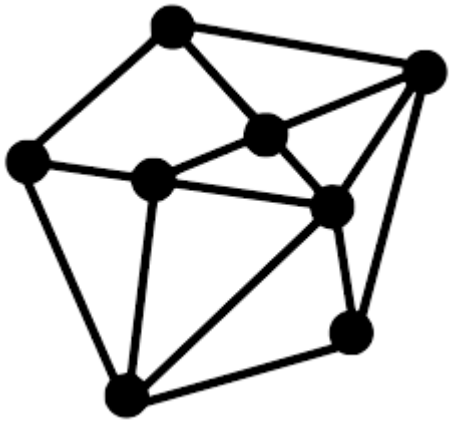
Overall Discussion of Results

Continuous Information Incorporation



Potential Future Research

Other Data



More Sophisticated Algorithms

Partnerships and real-world implementation

Conclusion

Better schedules, better transit, better city