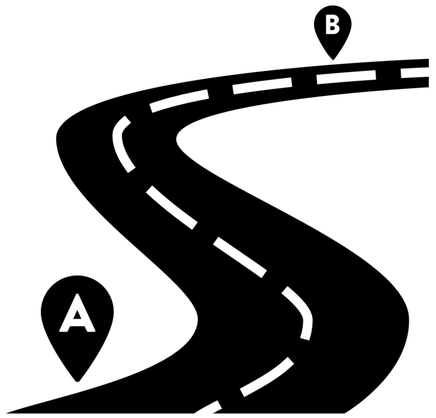


Honors Thesis/Creative Project

A Graph-Based Machine Learning Approach to Realistic Traffic Volume Generation

Kyle Otstot, Spring 2022

Brief Introduction

Hello, I'm Kyle Otstot!

- Third year major in Computer Science & Mathematics
- Incoming Master's in Computer Science
- Research Assistant at ASU
  - Robustifying ML models under real-world dataset corruptions

Committee members:

- Dr. Gennaro De Luca – Director
- Dr. Yinong Chen – Second reader

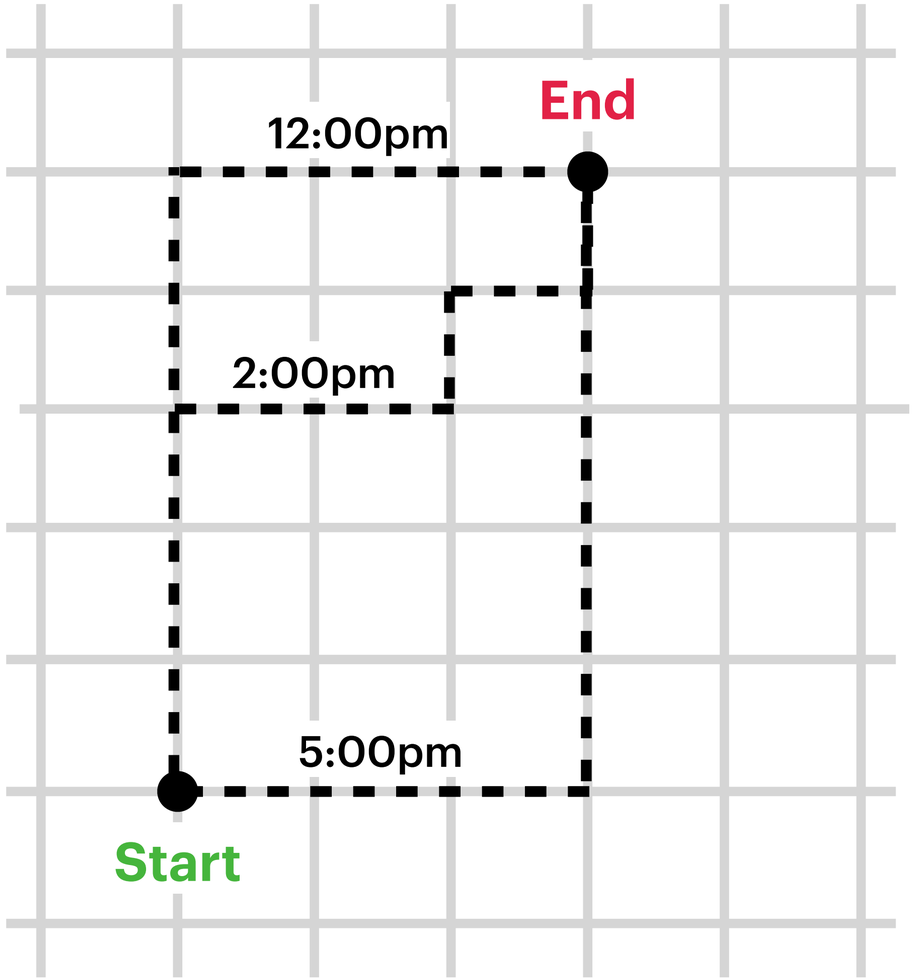
CSE Capstone Project

Dynamic Shortest-Path Algorithm:

- Every hour, the algorithm predicts the shortest path between any start / end points in a fixed road network
- The algorithm needs the following:
  - Structure (nodes / edges) of road network
  - Hourly edge weights – expected travel time
- Road network & hourly data should be realistic

Honors Thesis/ Creative Project:

- An extension of the capstone project



Searching For Traffic Data

How is real-world traffic measured?

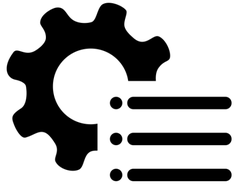
- **Traffic volume:** the number of passing cars per unit of time (cars/hour)
- **Traffic density:** the number of cars in a specified length of roadway (cars/mile)
- $$\text{travel time (hour)} = \frac{\text{density (cars/mile)} \times \text{road length (miles)}}{\text{volume (cars/hour)}}$$
- **New York State Department of Transportation** (NYSDOT) reports hourly traffic volume [1]
- Traffic density is not recorded
  - Set constant, focus on obtaining traffic volume

Searching For a Road Network

Where can we find a real-world road network?

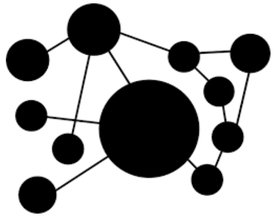
- **OpenStreetMap (OSM)** is a free geographic database of the world [2]
- A GraphML file of **Manhattan, New York City** was created from the OSM database [3]
- The Manhattan graph consists of the following:
  - Nodes (Intersections)
  - Node data (GPS coordinates, roads, *etc.*)
  - Directed edges (Road segments)
  - Edge data (road name, length, speed limit, *etc.*)

OpenStreetMap

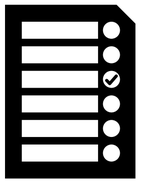


Problem Setup

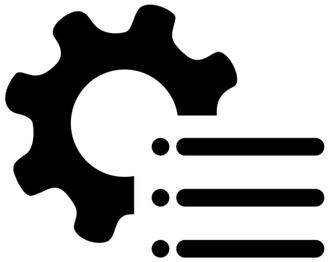
Dataset Generation & Preprocessing



Predictive Modeling



Model Evaluation & Discussion

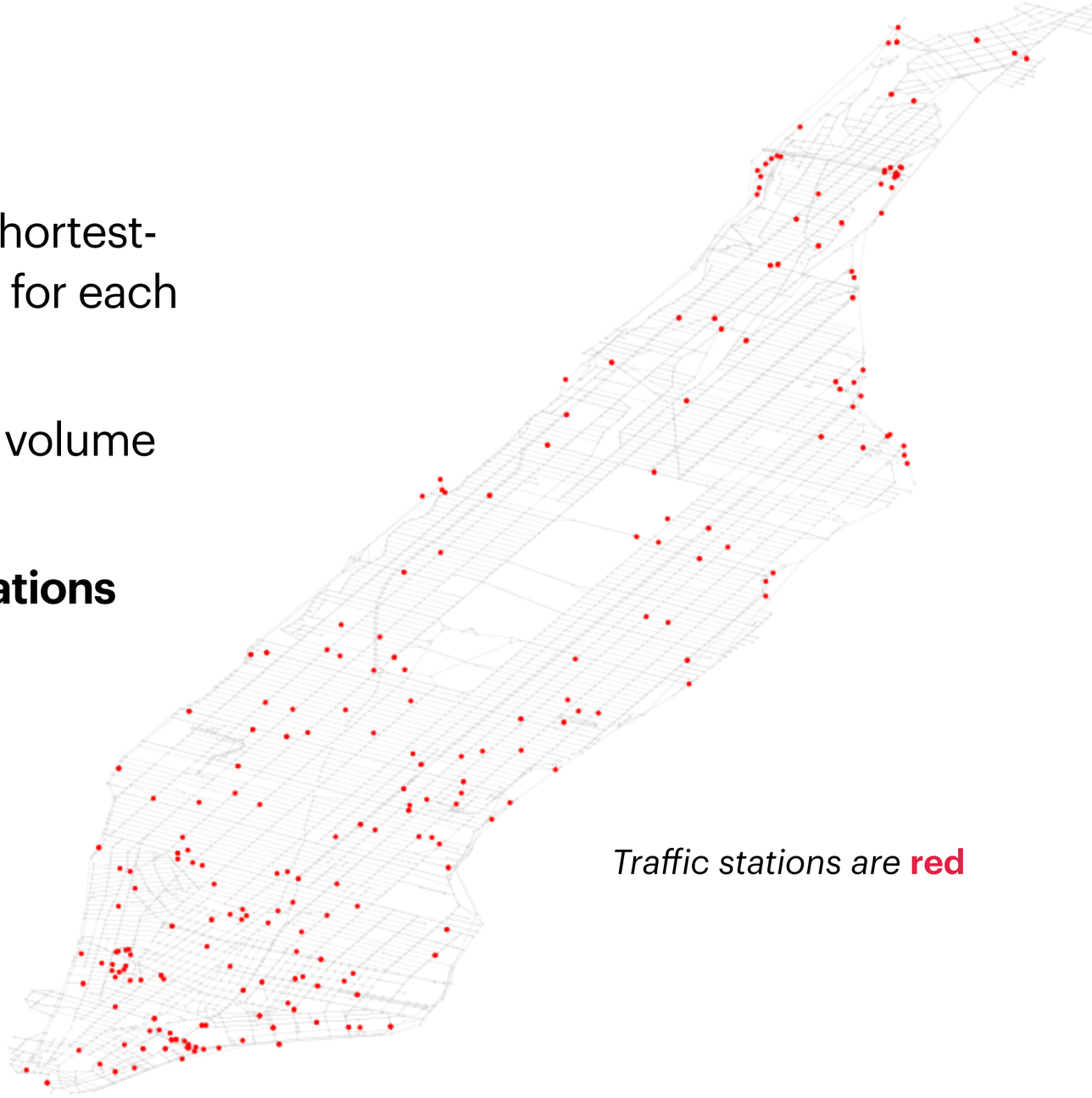


Problem Setup

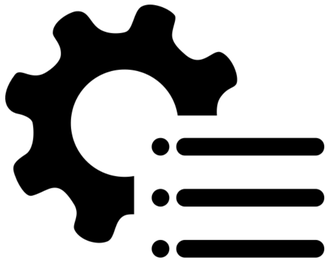
Contents

Insufficient Data

- In order to make predictions with a dynamic shortest-path algorithm, we need hourly traffic volume for each road segment (edge) in the network
- However, NYSDOT only receives hourly traffic volume from a **small subset** of road segments
- Hourly traffic volume is collected by **traffic stations**



Traffic stations are **red**



Problem Setup

Contents

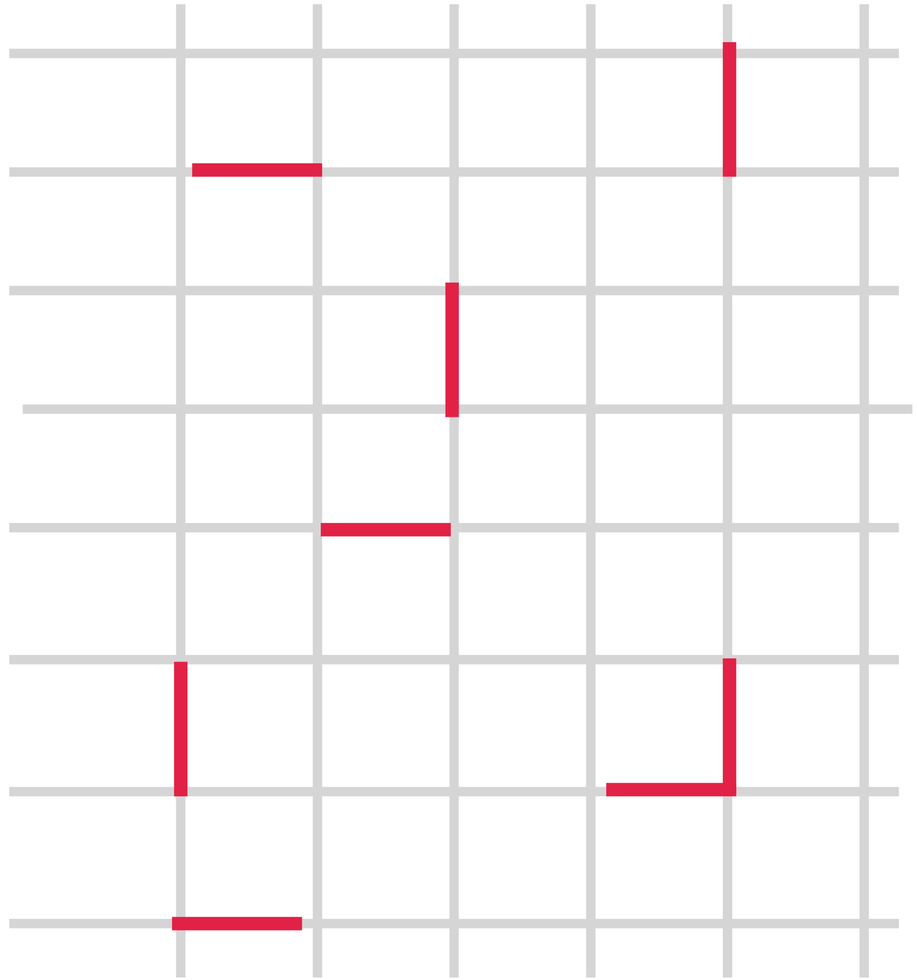
Machine Learning Solution?

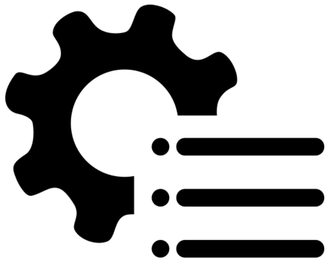
Research Question:

- Can we develop a Machine Learning (ML) algorithm that **generalizes** the NYSDOT data to **all road segments** in Manhattan?
- Can a model make accurate predictions on unforeseen road segments by learning the traffic patterns in ~300 recorded examples?

Main Tasks:

1. Generate & preprocess a **supervised learning dataset**
   - Input – road segment attributes
   - Output – hourly traffic volume
2. Evaluate the performance of **ML algorithms** on the dataset





Problem Setup

Contents

Technologies Used

Programming:

- Python & Jupyter Notebook

Python libraries:

- **General use**
  - ✓ Numpy, Pandas
- **Data visualization**
  - ✓ Matplotlib, Pandas
- **Graph handling**
  - ✓ Networkx
- **Machine Learning**
  - ✓ Scikit-Learn, Pytorch

Dataset Generation & Preprocessing

Contents

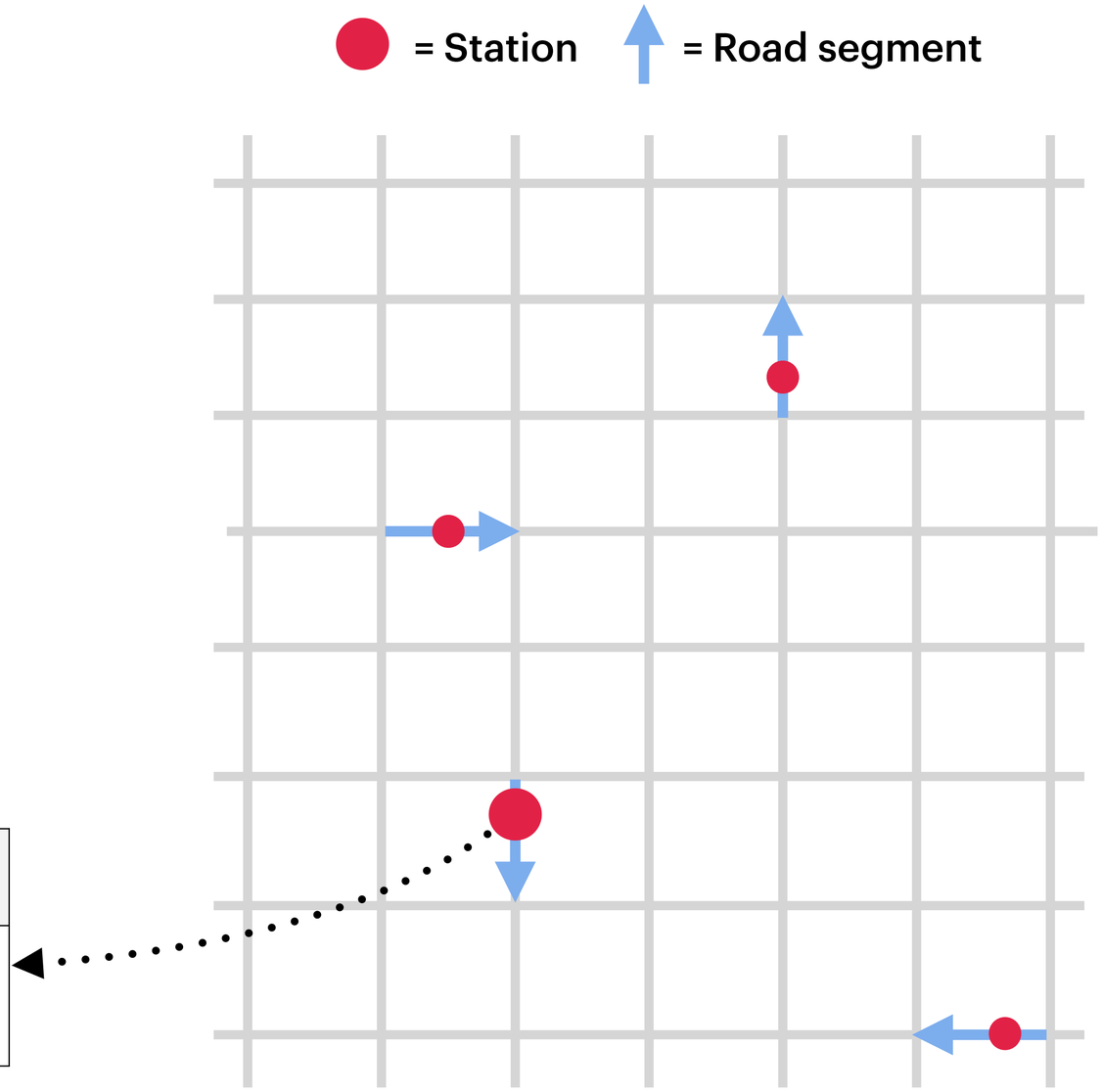
Traffic Station Overview

Traffic Station: a device that counts the number of passing vehicles and reports hourly traffic volume.

Station details:

- Each station belongs to one road segment
- Total of 310 stations in Manhattan
- A 24-hour volume breakdown for each station is reported

12am	1am	2am	...	9pm	10pm	11pm
656	352	252	...	1265	1068	878



Traffic Station Overview

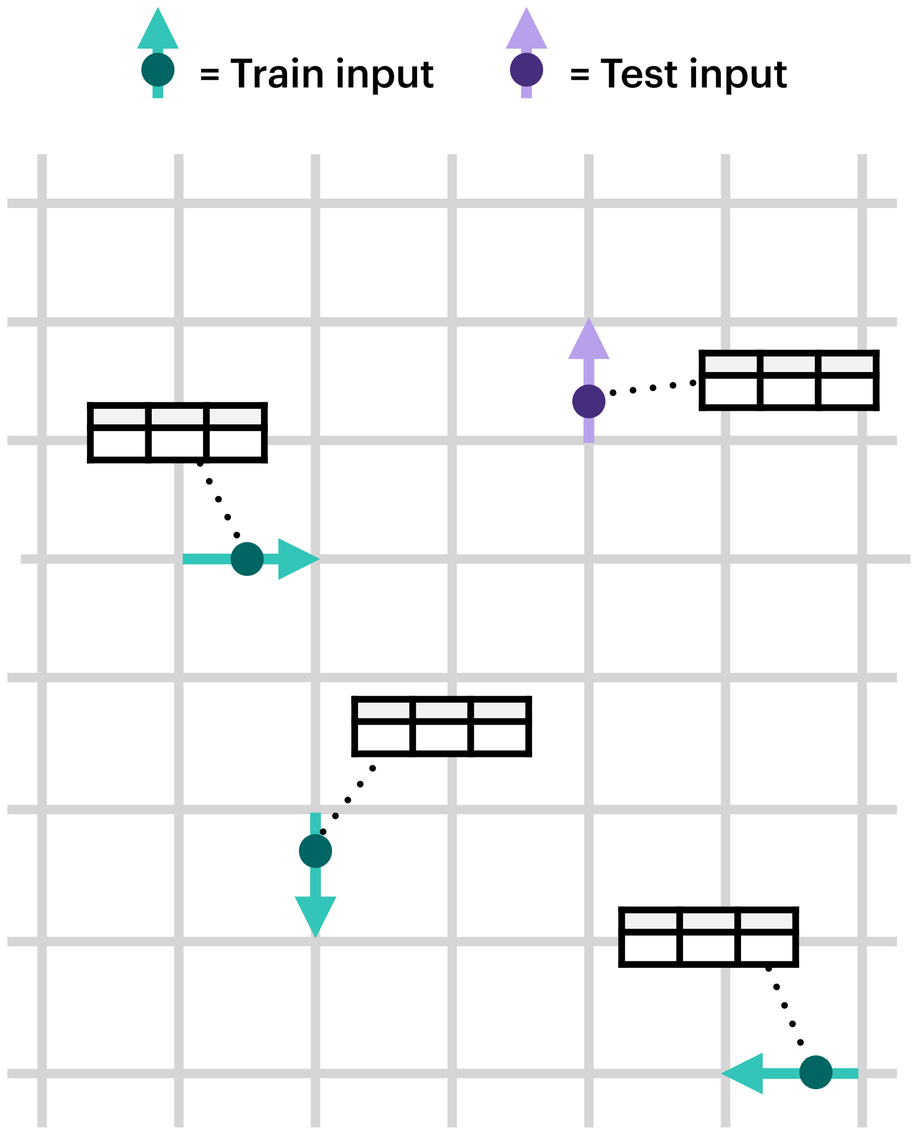
Main idea: create a function that receives any road segment and predicts a 24-hour volume vector.

Procedure:

- Train a ML model on pairs of road segment attributes (input) and 24-hour traffic volume (“ground-truth” output)
- Test the model on pairs **not** used for training

Next steps:

- For each station, find the road segment that contains it
- Pair the road segment attributes with the corresponding station 24-hour volume



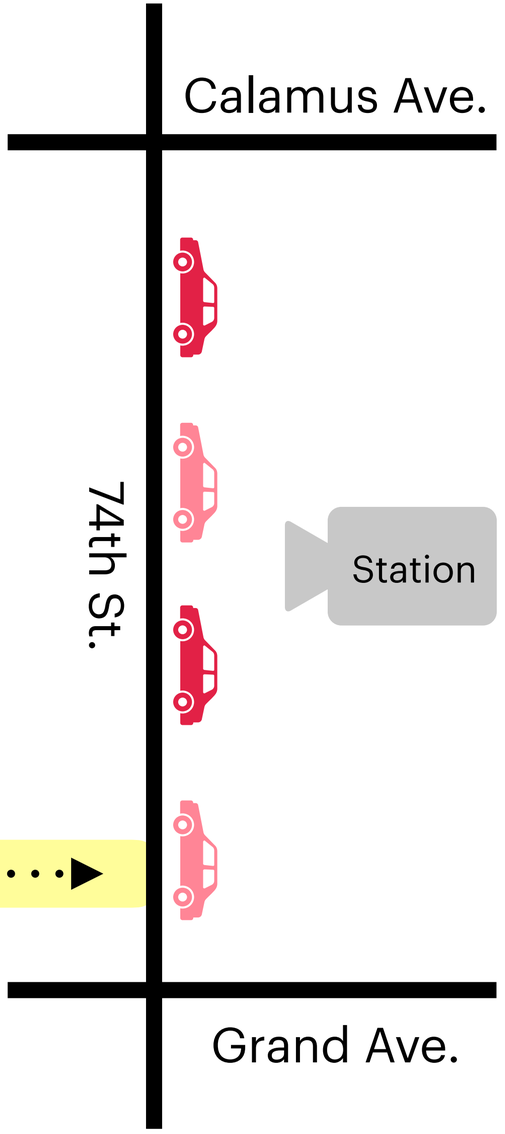
Dataset Generation & Preprocessing

Contents

Station Database Breakdown

Traffic station information taken from the ***NYSDOT public database***.

	RCSTA	Roadway Name	Station Start	Station End	Federal Direction	Latitude	Longitude
0	44080	E 13TH ST	AVENUE C	AVENUE D	Eastbound	40.72735	-73.97518
1	44080	E 13TH ST	AVENUE C	AVENUE D	Westbound	40.72735	-73.97518
2	44080	E 13TH ST	AVENUE C	AVENUE D	Combined Total	40.72735	-73.97518
3	44130	E 8TH ST	FIFTH AVE	3RD AVENUE	Eastbound	40.73031	-73.99174
4	44130	E 8TH ST	FIFTH AVE	3RD AVENUE	Combined Total	40.73031	-73.99174
...	...	...	...	...	...	...	...
324	14250	SOUND VIEW AVE	ROSEDALE AVE	STORY AVE	Southbound	40.82272	-73.86753
325	14250	SOUND VIEW AVE	ROSEDALE AVE	STORY AVE	Combined Total	40.82272	-73.86753
326	54250	74TH ST	GRAND AVE	CALAMUS AVE	Northbound	40.73180	-73.88825
327	54250	74TH ST	GRAND AVE	CALAMUS AVE	Southbound	40.73180	-73.88825
328	54250	74TH ST	GRAND AVE	CALAMUS AVE	Combined Total	40.73180	-73.88825



Road Segment Data Breakdown

Road segment information taken from ***OpenStreetMap***.

Feature	Total #	Data example
Node (Intersection)	4426	('42459137', {'y': '40.7755735', 'x': '-73.9603796', 'osmid': '42459137', 'highway': 'traffic_signals'})
Edge (Road segment)	9609	('42449589', '42433292', {'maxspeed': '25 mph', 'lanes': '2', 'name': 'Mercer Street', 'length': '128.705418668', 'oneway': 'True', 'highway': 'residential', 'osmid': '25199883', 'key': 0, 'distance': 128.705418668})

Simple rendering of map with ***Networkx***

Mapping Stations to Road Segments

Attempt #1- by road names:

1. Input: Station ID (RCSTA + Direction)
2. From NYSDOT data, find
   - Start intersection (“Roadway name”, “Station start”)
   - End intersection (“Roadway name”, “Station end”)
3. Search OSM data for all the edges that correspond to each road name. From here you can derive the two intersection IDs
4. Find the particular edge that contains both intersection IDs: this is your road segment

Road name match example:

Database	Road name
NYDOT	E 79TH ST
OSM	East 79th Street

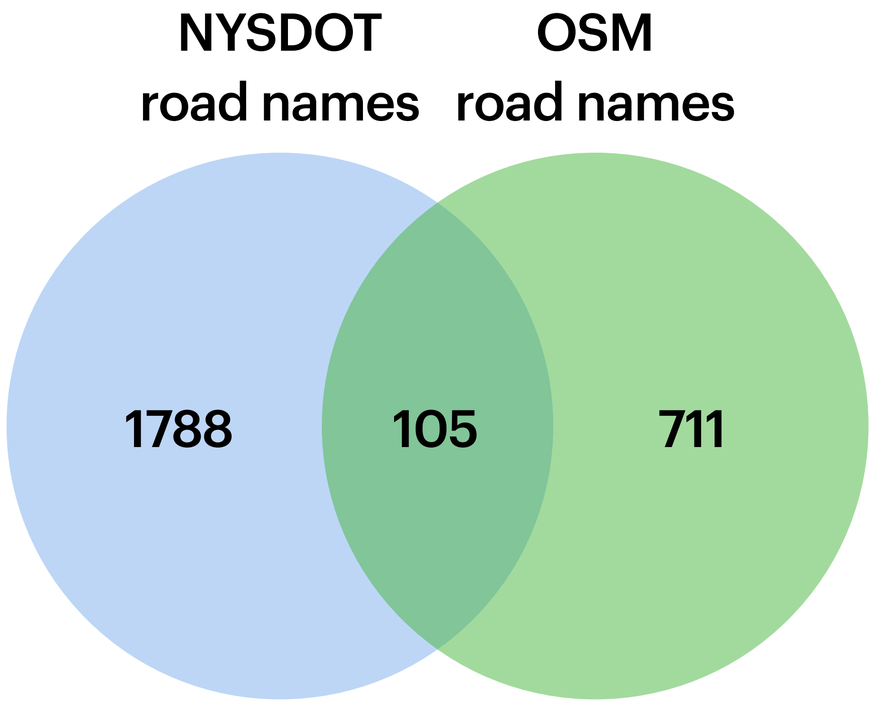
Token conversions:

- N,S,E,W ⇒ North, South, East, West
- Uppercase ⇒ Capitalize first letter only
- St, Ave, Rd ⇒ Street, Avenue, Road

Mapping Stations to Road Segments

Attempt #1- by road names:

1. Input: Station ID (RCSTA + Direction)
2. From NYSDOT data, find
   - Start intersection (“Roadway name”, “Station start”)
   - End intersection (“Roadway name”, “Station end”)
3. Search OSM data for all the edges that correspond to each road name. From here you can derive the two intersection IDs
4. Find the particular edge that contains both intersection IDs: this is your road segment

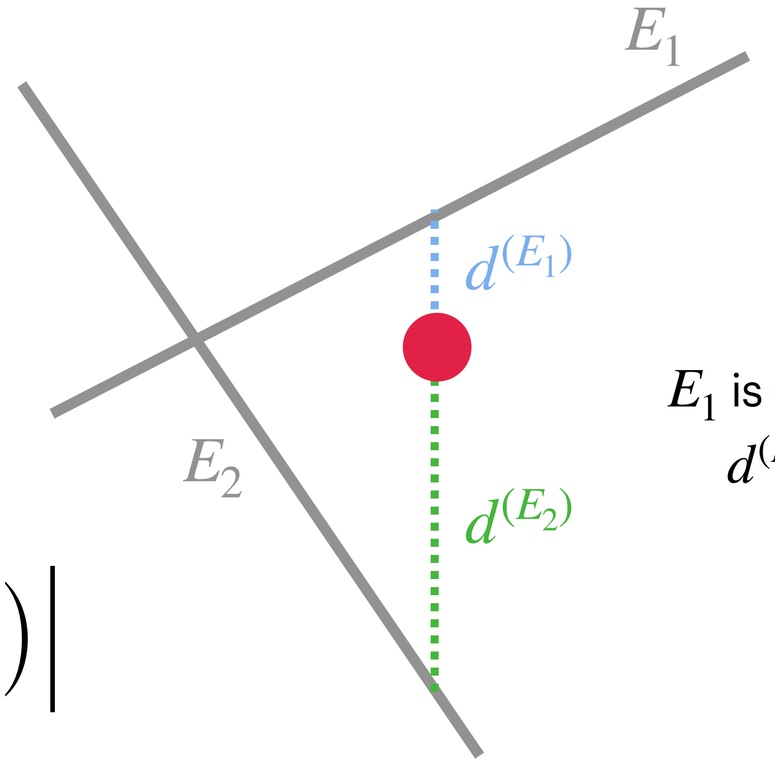


Mapping Stations to Road Segments

Attempt #2- by latitude & longitude:

1. Input: Station ID (RCSTA + Direction)
2. From NYSDOT data, find latitude & longitude
3. Determine the similarly-directed road segment with the following formula:

$$\text{edge}^* = \min_{E \in \text{edges}} d^{(E)} := \left| y_s - \left(m^{(E)}(x_s - x_1^{(E)}) + y_1^{(E)} \right) \right|$$



E_1 is chosen since $d^{(E_1)} < d^{(E_2)}$

where...

$(x_1^{(E)}, y_1^{(E)})$: long, lat of edge E's start intersection

$(x_2^{(E)}, y_2^{(E)})$: long, lat of edge E's end intersection

$$m^{(E)} = \frac{y_2^{(E)} - y_1^{(E)}}{x_2^{(E)} - x_1^{(E)} + \epsilon}, \ \epsilon \text{ : random noise}$$

(x_s, y_s) : long, lat of station

Mapping Stations to Road Segments

Attempt #2- by latitude & longitude:

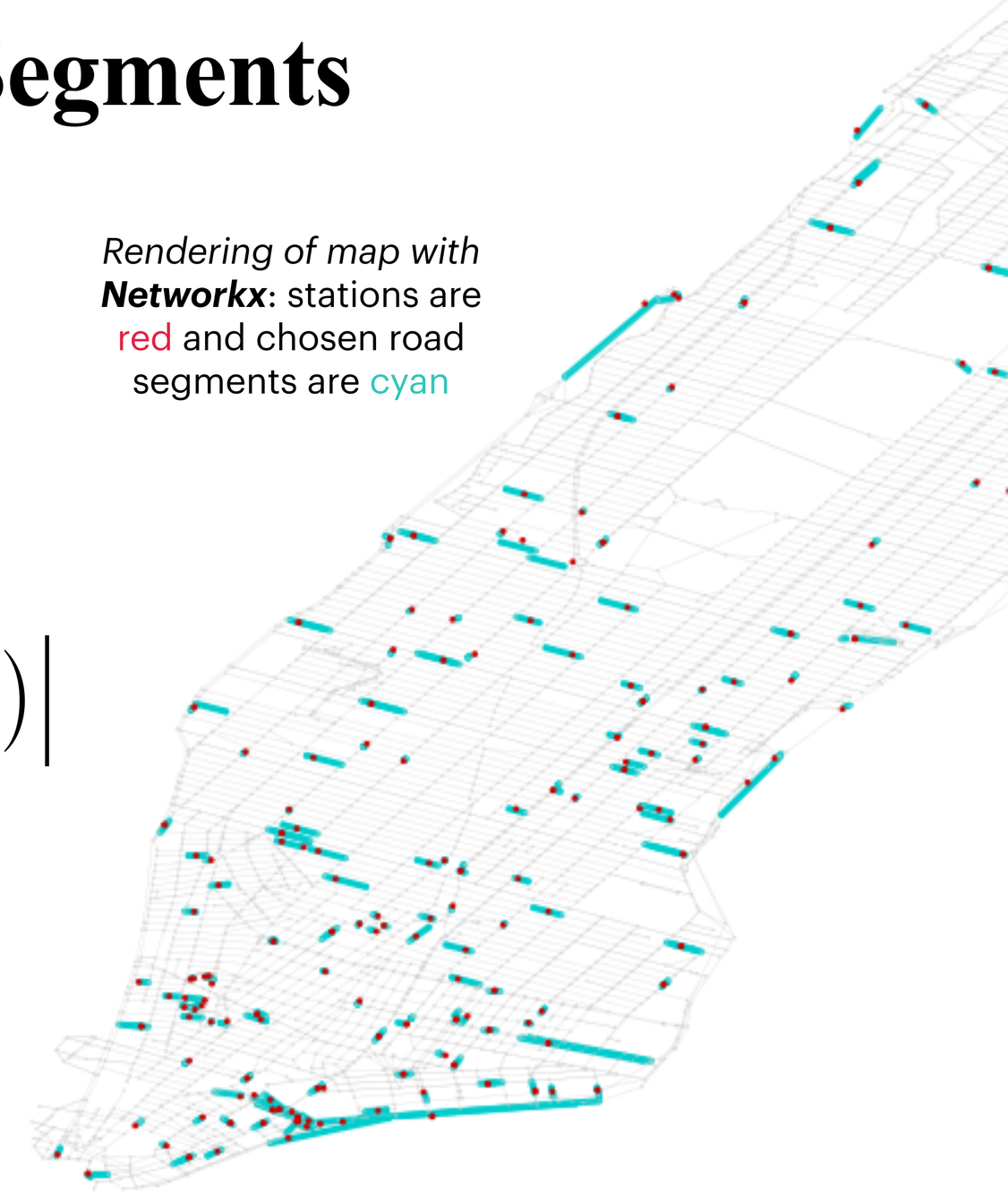
1. Input: Station ID (RCSTA + Direction)
2. From NYSDOT data, find latitude & longitude
3. Determine the similarly-directed road segment with the following formula:

$$\text{edge}^{*} = \min_{E \in \text{edges}} d^{(E)} := \left| y_s - \left(m^{(E)}(x_s - x_1^{(E)}) + y_1^{(E)} \right) \right|$$

Final Results:

- Appears to be an accurate approximation of the true station → road segment mapping

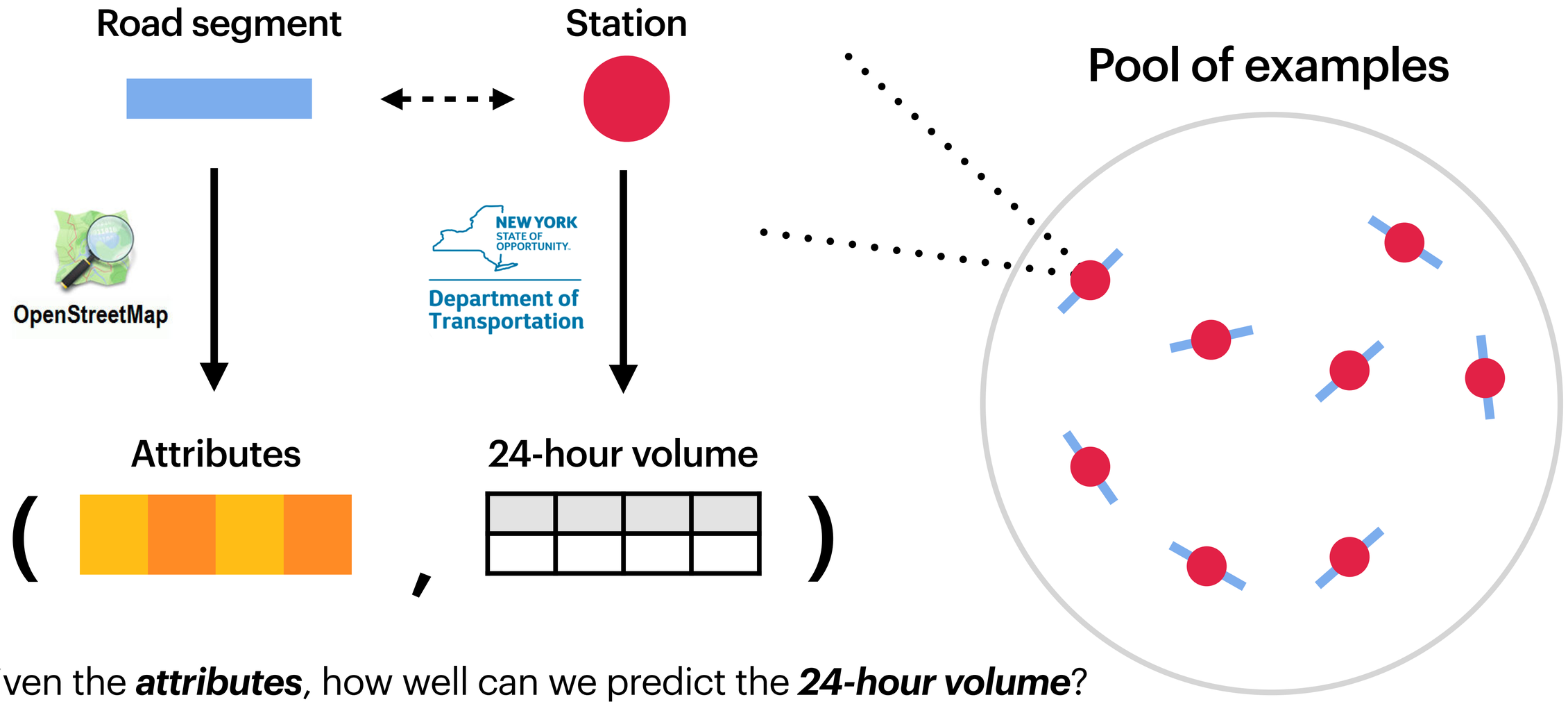
Rendering of map with ***Networkx***: stations are red and chosen road segments are cyan



Dataset Generation & Preprocessing

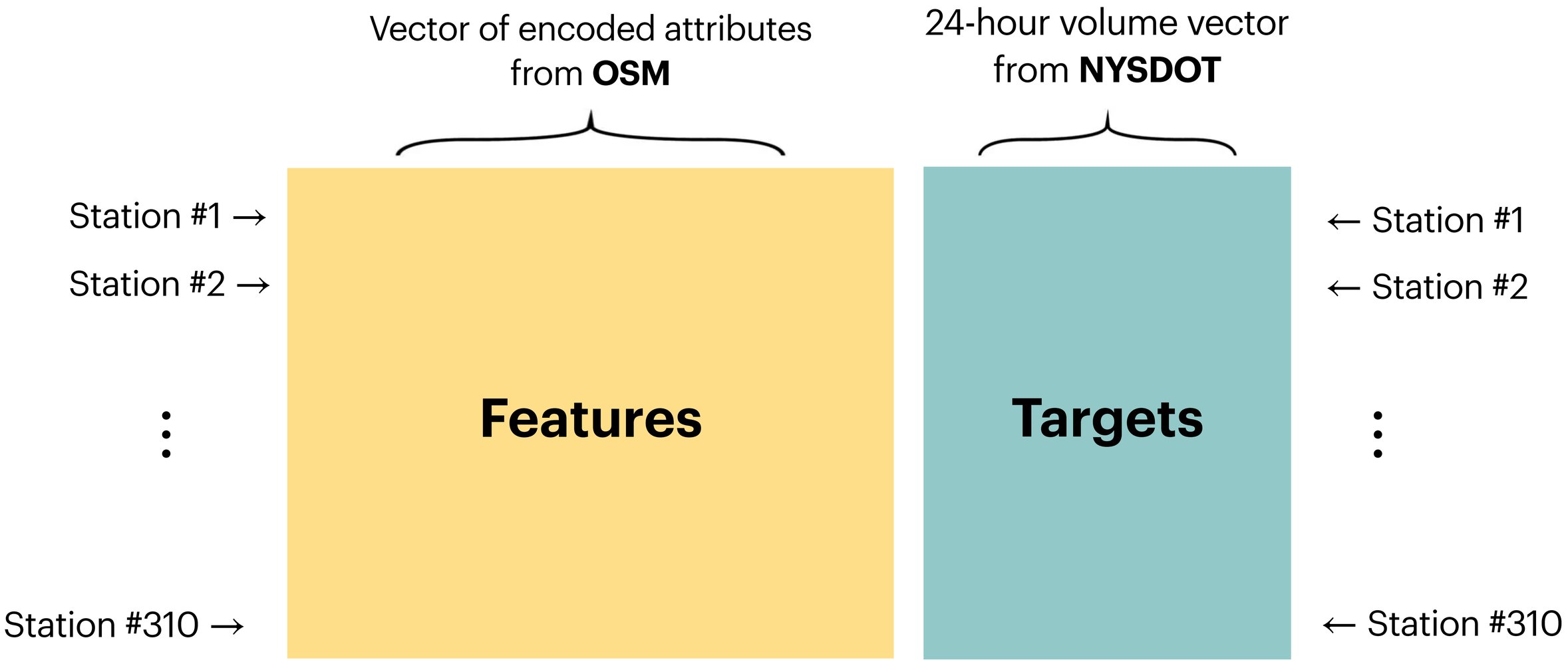
Contents

Dataset Generation Pipeline



Given the ***attributes***, how well can we predict the ***24-hour volume***?

Dataset Generation Pipeline

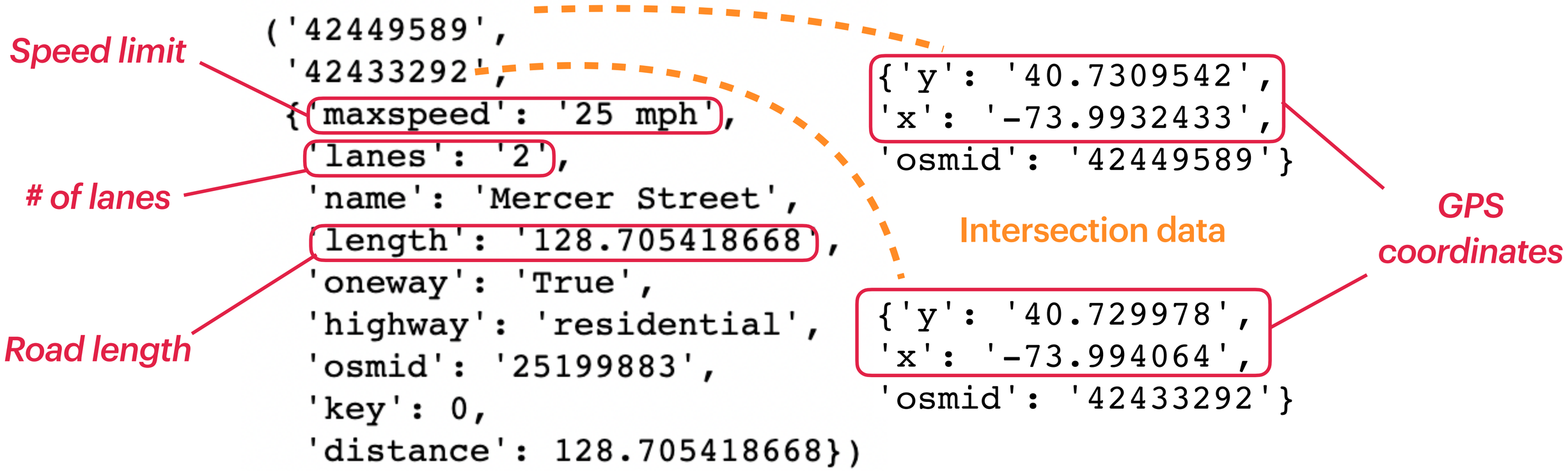


Dataset Generation & Preprocessing

Contents

Identifying Relevant Attributes From OSM

OSM data of road segment:

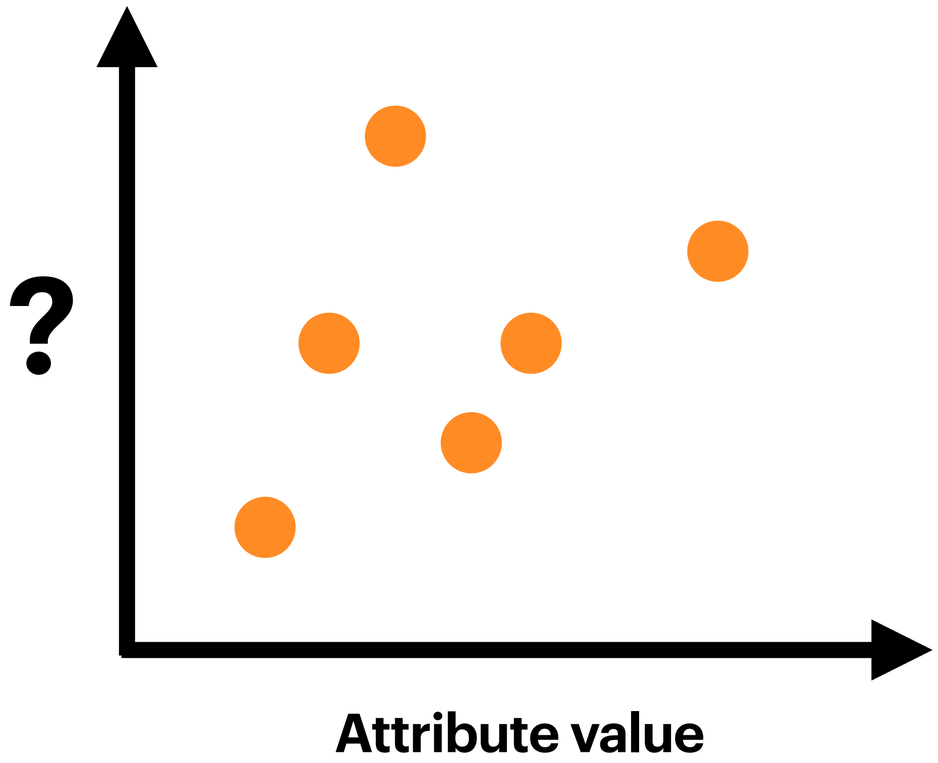


Relating Attributes to Traffic Volume

For analysis, we are interested in the predictive capability of each attribute for *traffic volume*.

Problem: the traffic volume space is *24 dimensions*. Impossible to visualize without dimensionality reduction.

Solution: reduce 24-hour volume vector down to a single value representing the whole vector, so we can plot ***attribute value vs. traffic volume***.

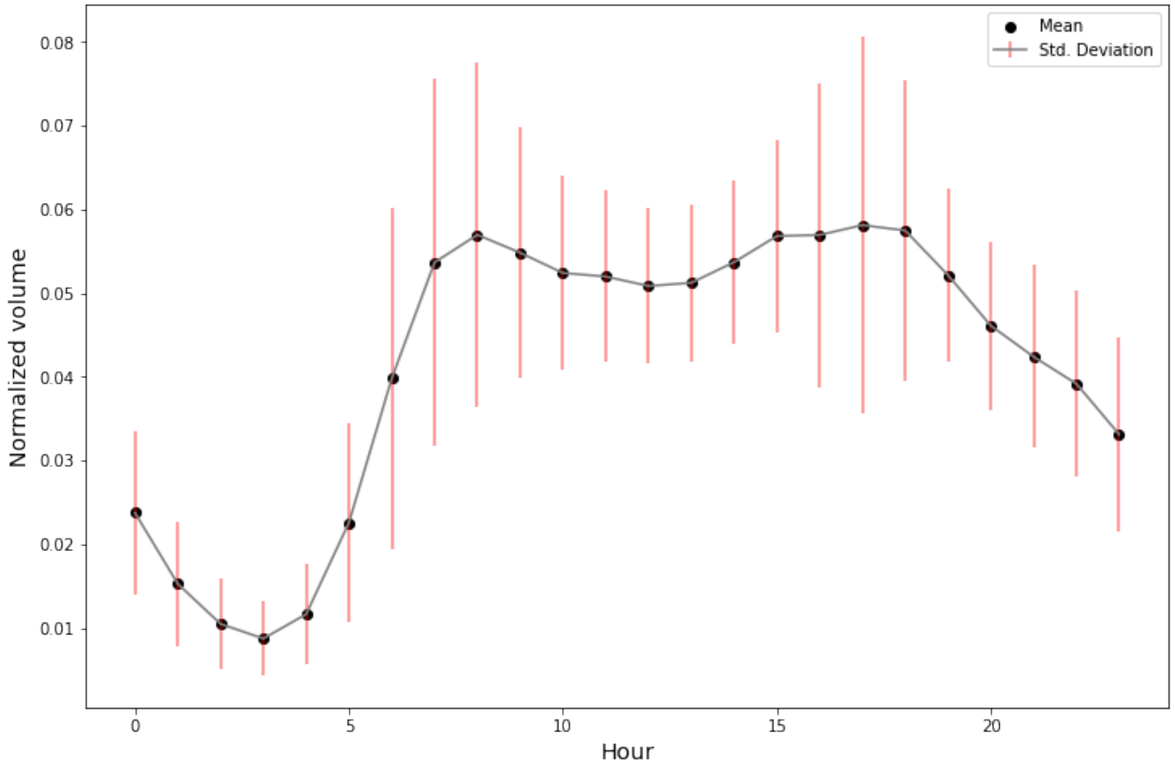


Relating Attributes to Traffic Volume

Claim: the daily volume count (*a.k.a.* vector-wise sum) is a very good single-value representation of the 24-hour traffic volume.

Evidence:

Normalization



- If we divide each 24-hour volume vector by its sum, then the hour-wise distributions are very **compact** and reflect a clear pattern in daily traffic:
  - **12am–5am:** low volume; sleeping
  - **7am-9am:** high volume; going to work
  - **4pm-6pm:** high volume; leaving work

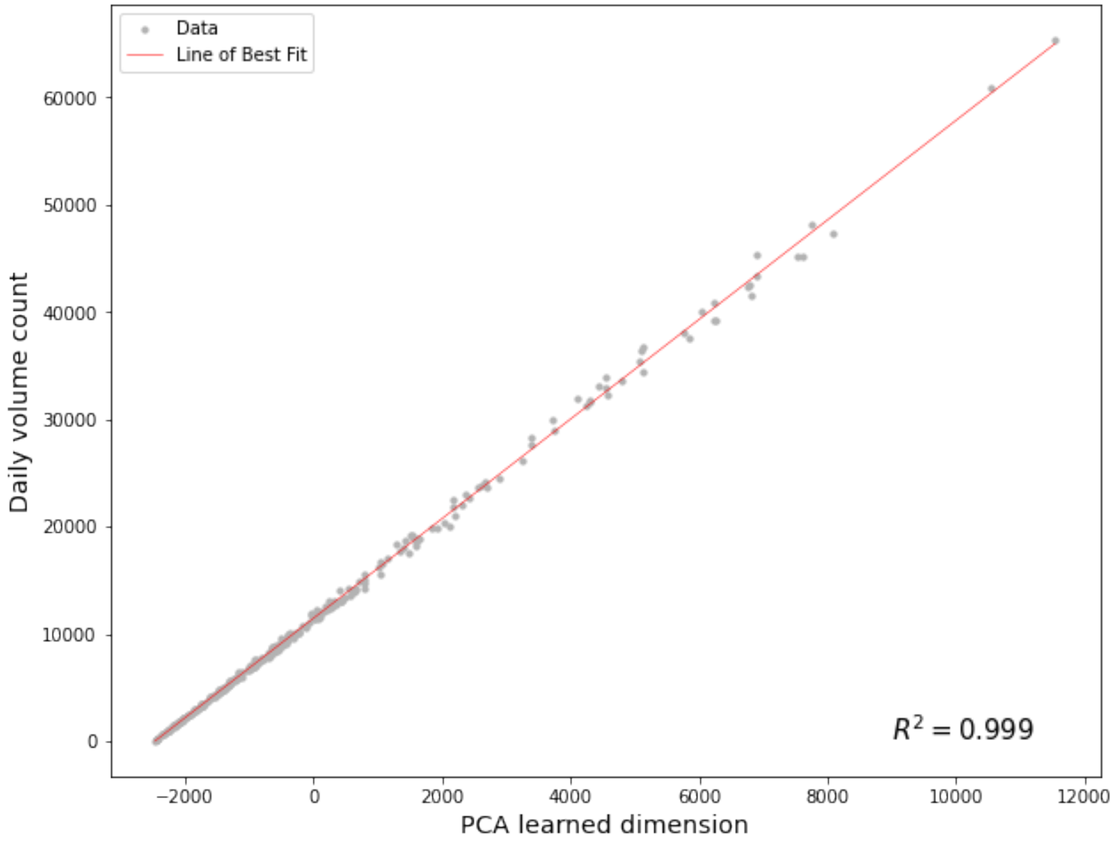
Relating Attributes to Traffic Volume

Claim: the daily volume count (*a.k.a.* vector-wise sum) is a very good single-value representation of the 24-hour traffic volume.

Evidence:

- If we use **Scikit-Learn's implementation of PCA** *[X]* to project the 24-dimension traffic volume space onto a single dimension that best preserves variance among the samples, we observe that the learned 1D distribution of values is almost perfectly ($R^2 \approx 1$) directly related to daily volume count.

Principal Component Analysis (PCA)



1. GPS Coordinate

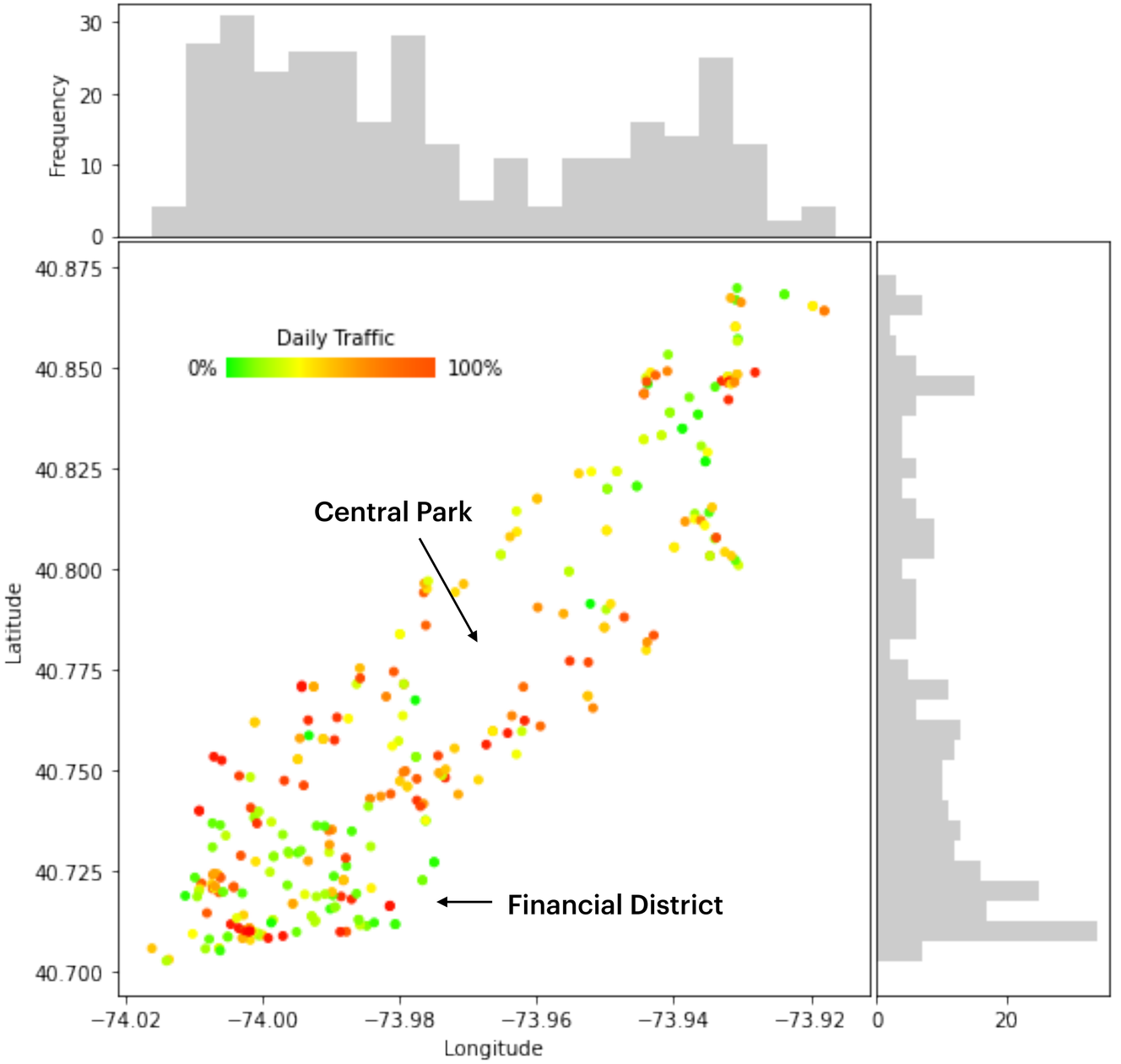
Hypothesis: Road segments within a geographic cluster are likely to reflect similar patterns in traffic volume.

Observations:

- Complex relationship between latitude-longitude and traffic volume
- Stations are (reasonably) well-distributed

Evaluation:

- Some clusters of similar volume exist
- Geographic location may be good predictor in a large capacity model



1. GPS Coordinate

Preprocessing Method:

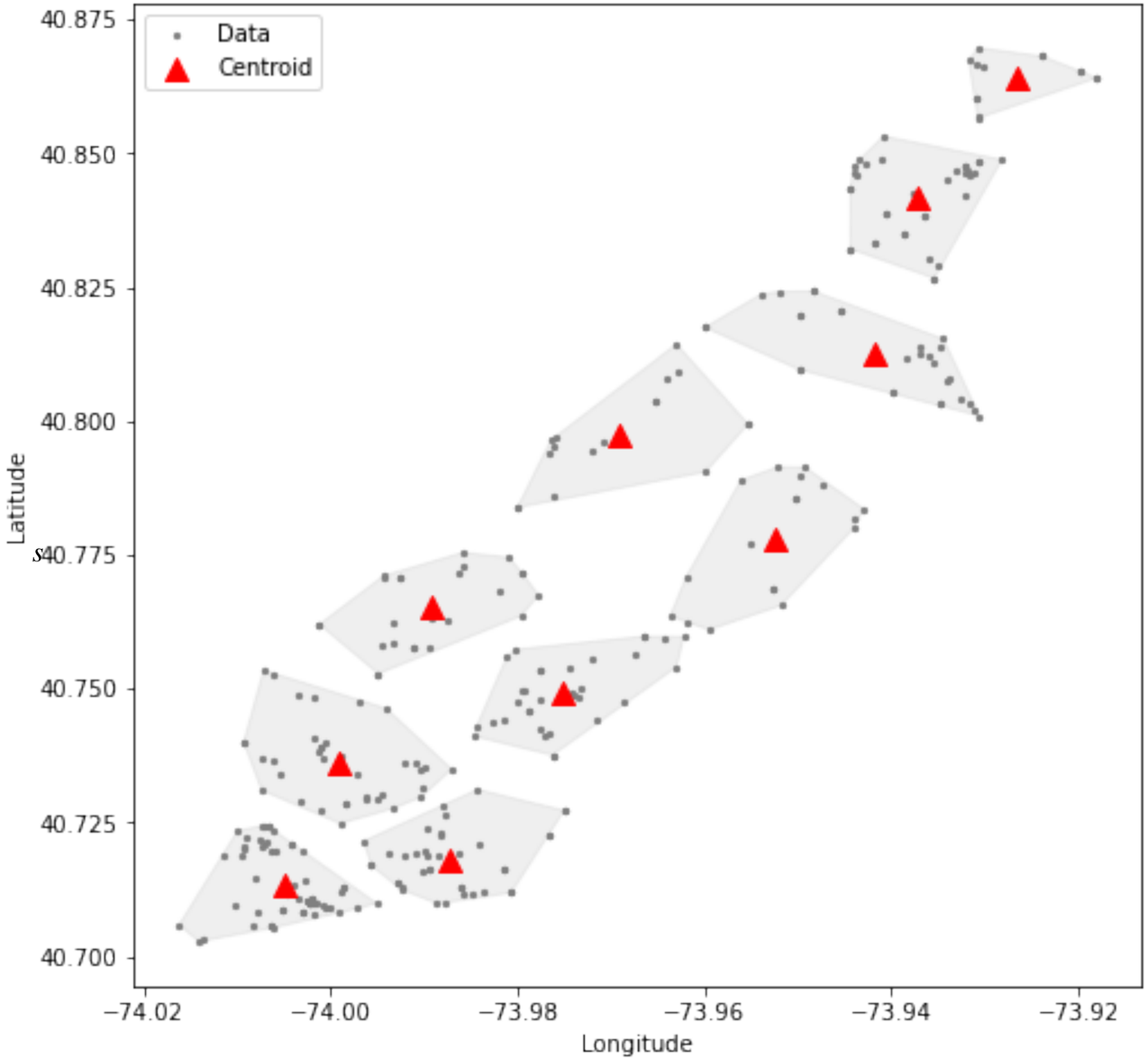
K-Means Clustering

- Algorithm finds K centroids that best represent the distribution of data
- Each centroid defines a cluster, and stations are assigned to closest centroid
- We use $K = 20$ with *SciKit-Learn's implementation [4]* for preprocessing

Encoding:

yes: **1**, no : **0**

Station →	Station in cluster **1**?	Station in cluster **2**?	…	Station in cluster **20**?



Clustering example of $K = 10$

2. Road Length (m)

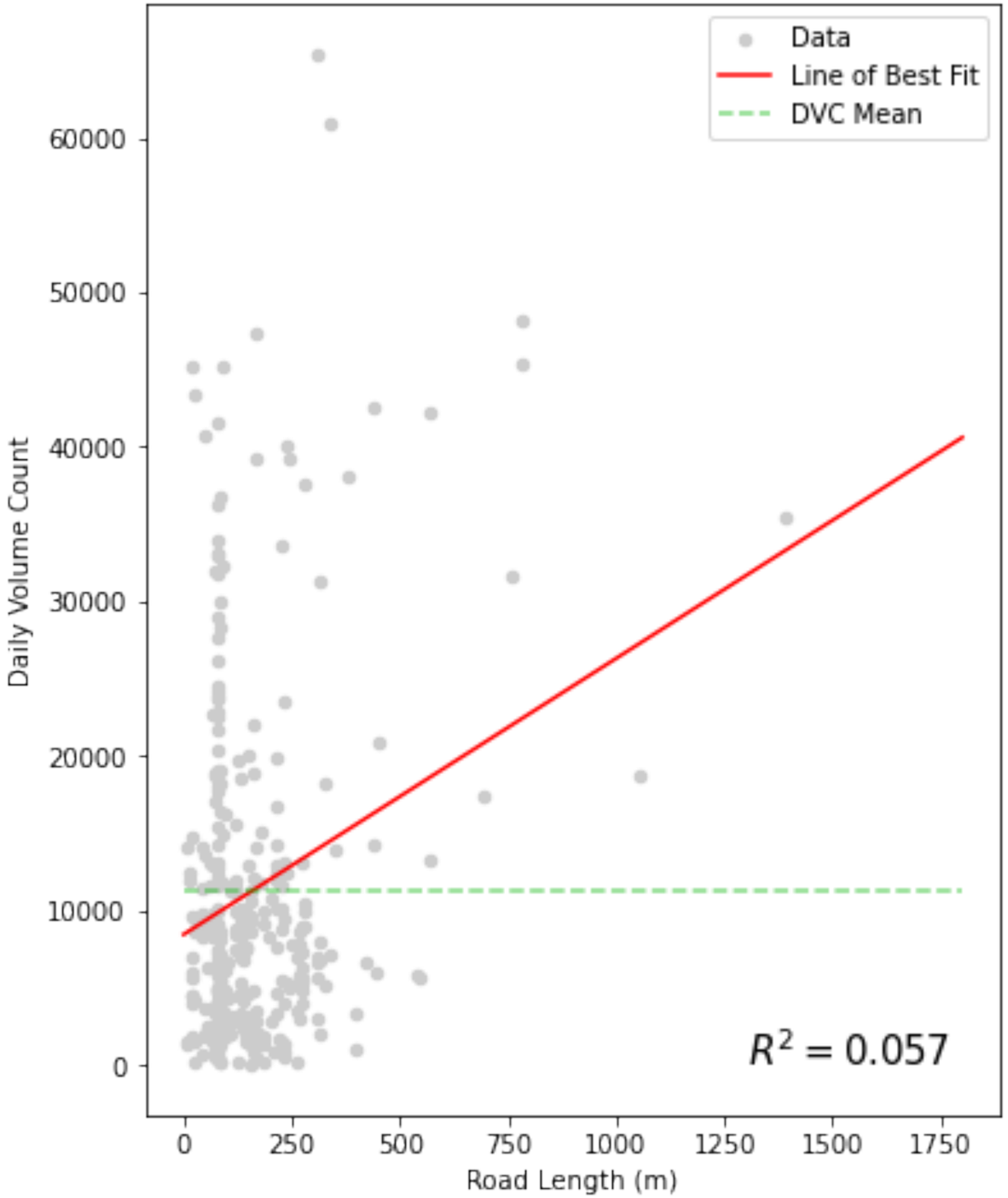
Hypothesis: Longer road segments are more important to the structure of the network, thus possess higher traffic volume levels.

Observations:

- Virtually no linear relationship between road length and traffic volume
- Outliers (long road segments) tend to reflect higher volume.

Evaluation:

- Road length, in isolation, appears to be a poor predictor of traffic volume
- May serve some use in complex regressors



2. Road Length (m)

Preprocessing Method: Normalization (w.r.t. the train set mean & std. deviation)

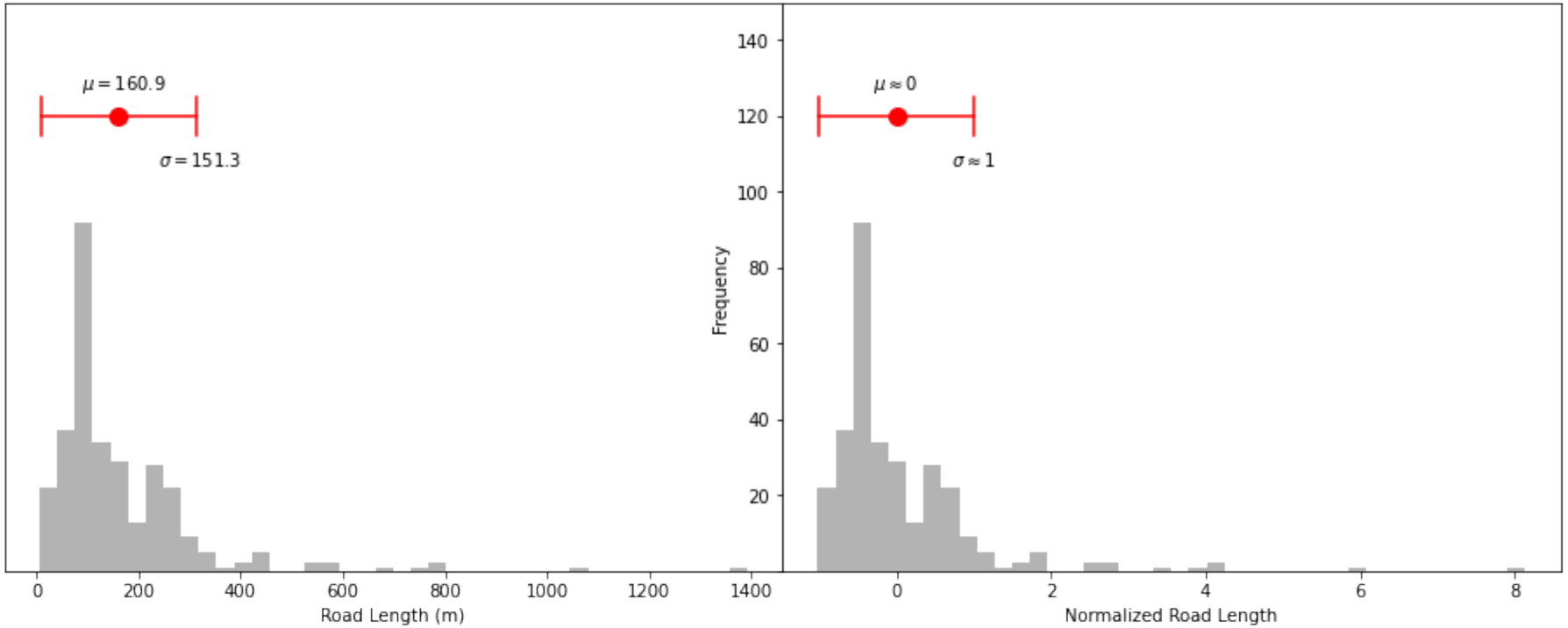
Justification: easily interpretable, boosts convergence speed in neural networks

Encoding:

Station → $(\text{road length} - \mu_{\text{train}})/\sigma_{\text{train}}$

μ_{train} : mean of road lengths in train set

σ_{train} : std. deviation of road lengths in train set



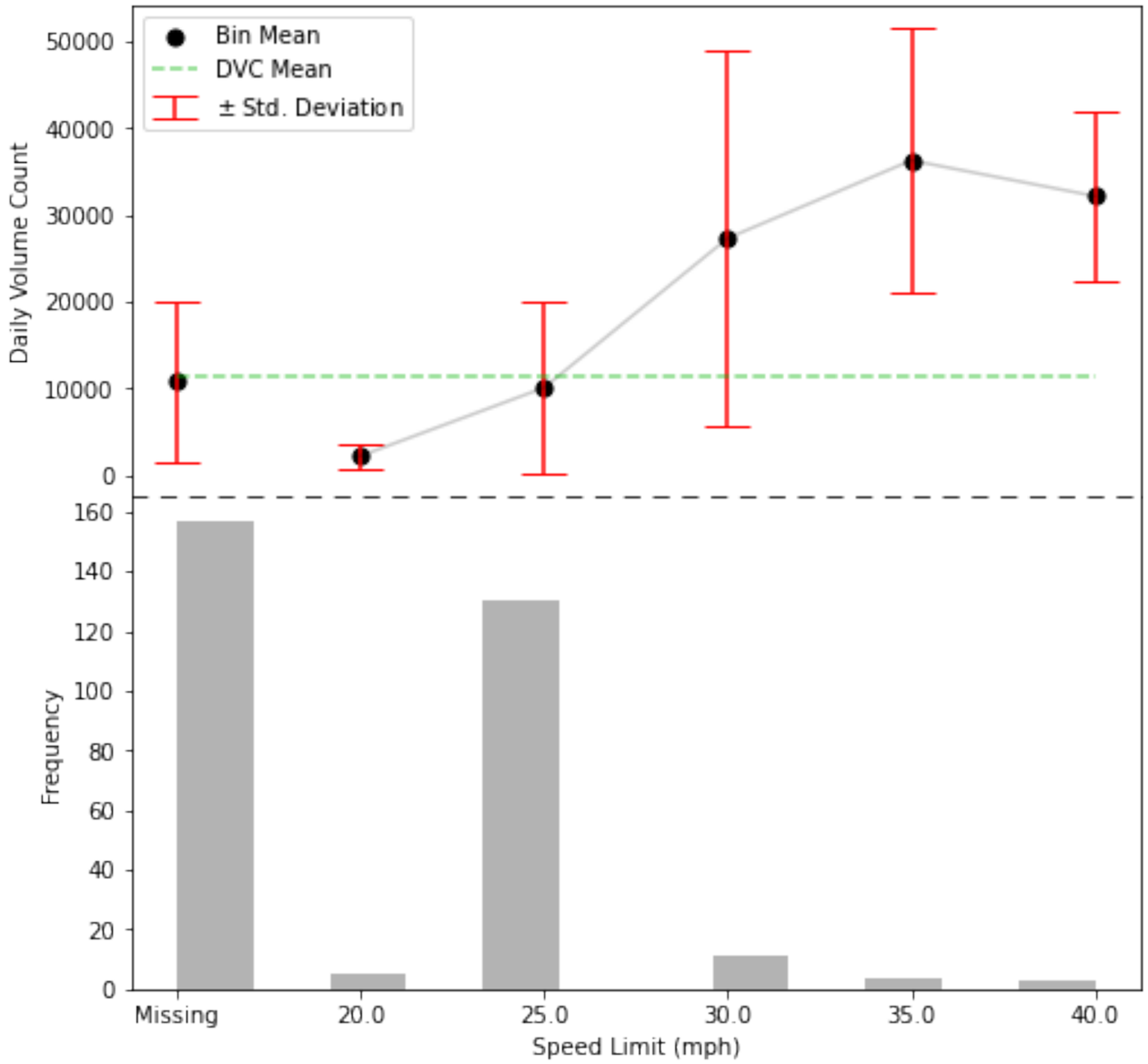
3. Speed Limit (mph)

Hypothesis: Vehicles are more likely to take roads with faster speed limits, so daily volume count should be directly related to speed limit

Observations:

- Significant amount of stations are missing a speed limit
- Traffic volume begins to decrease slightly after 35mph

Evaluation: Generally a direct relationship, and the slight decrease may be explained by the lack of 40mph examples



3. Speed Limit (mph)

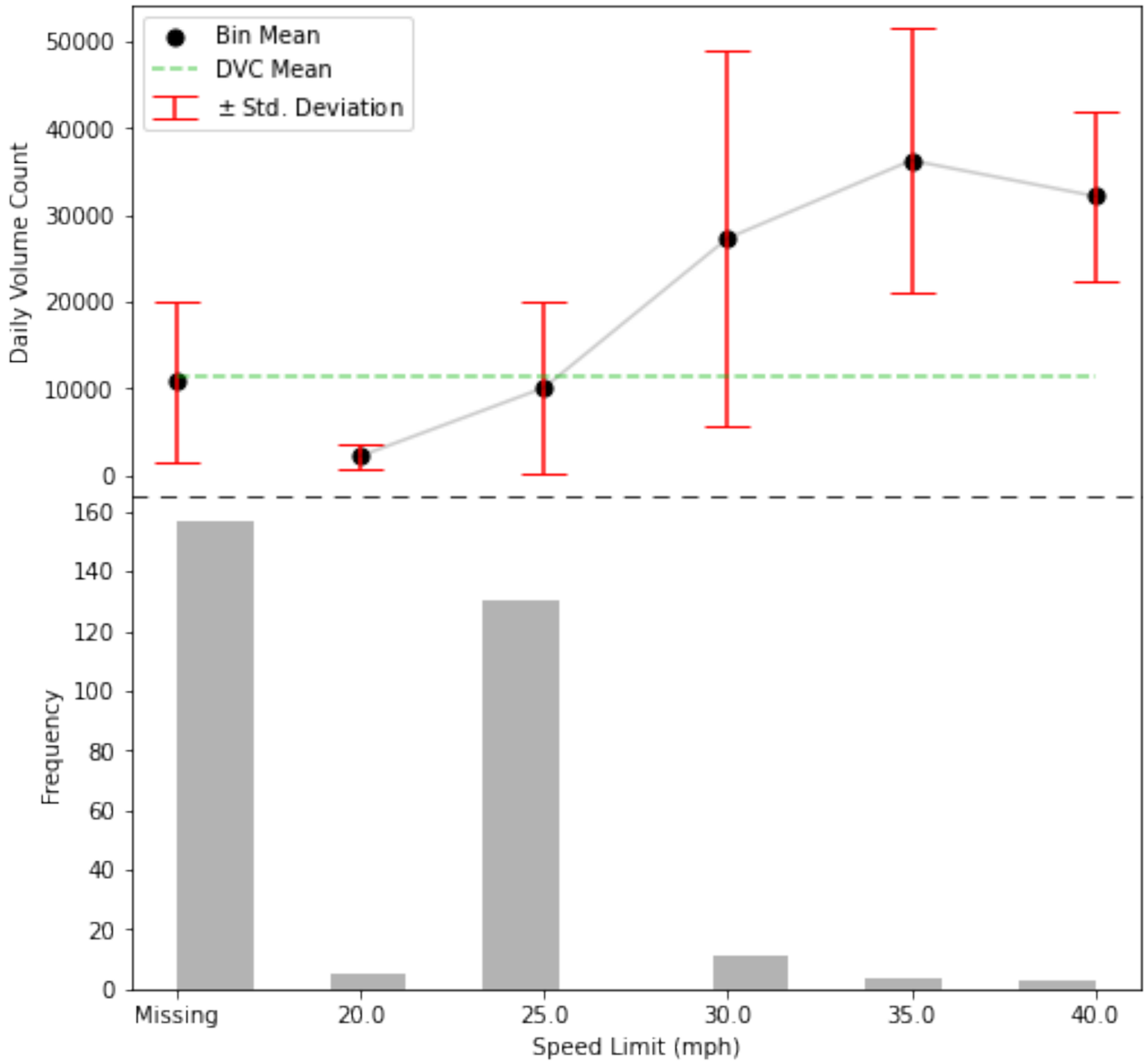
Preprocessing Method:

Handling missing data & Normalization

- The speed limit is encoded with two values: (1) a bit indicating whether or not the true value is missing, and (2) the normalized speed limit, or 0 if it's missing

Encoding:

Station →	Missing value? yes: **1**, no: **0**	**0** if missing; **normalized value** if not



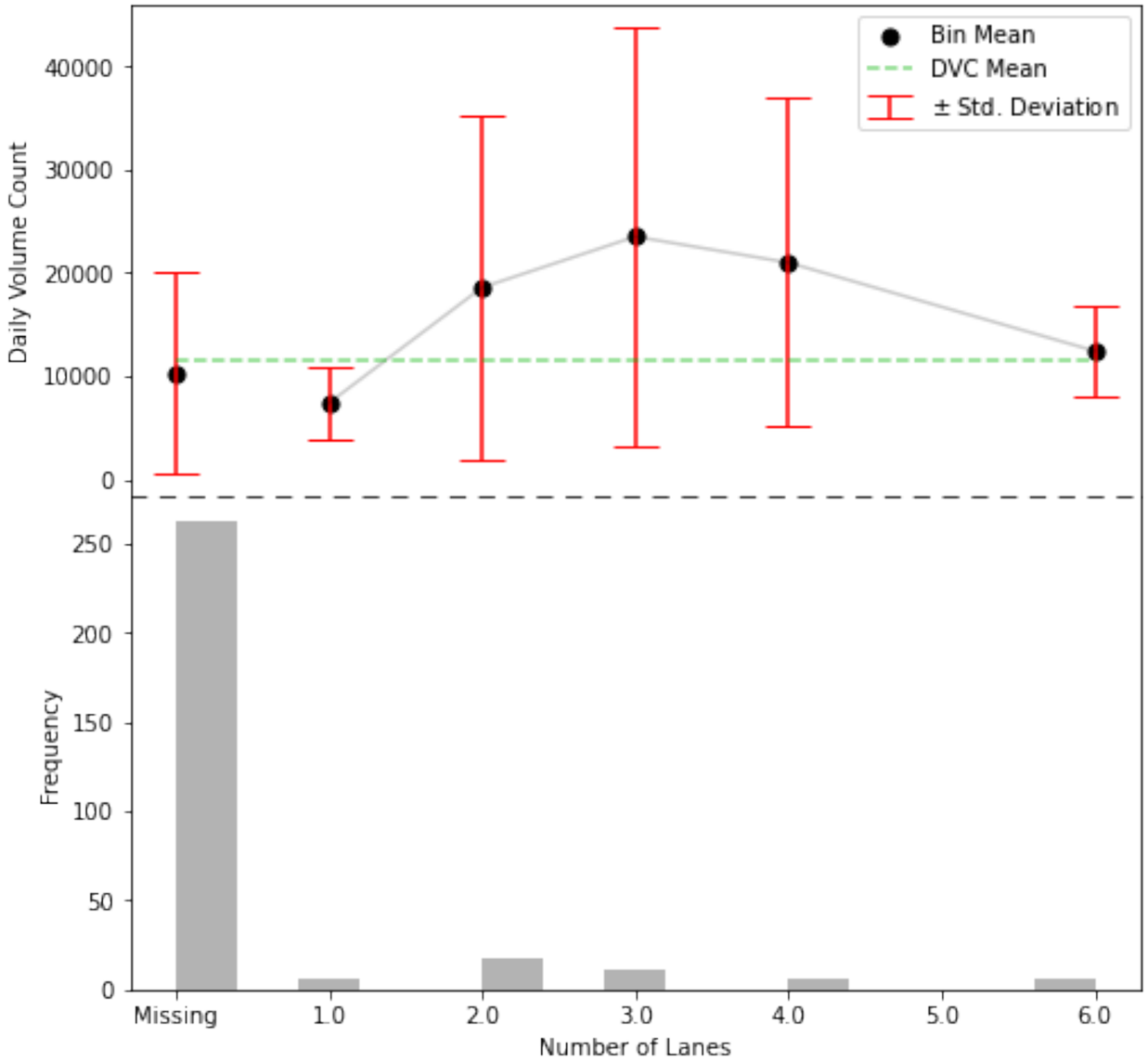
4. Number of Lanes

Hypothesis: Vehicles are more likely to take roads with more lanes, so daily volume count should be directly related to number of lanes

Observations:

- Vast majority of stations are missing a number of lanes
- Traffic volume begins to decrease after 3 lanes

Evaluation: As number of lanes increases, the traffic volume increases up until a point, then comes back down due to the sheer openness of the road.



4. Number of Lanes

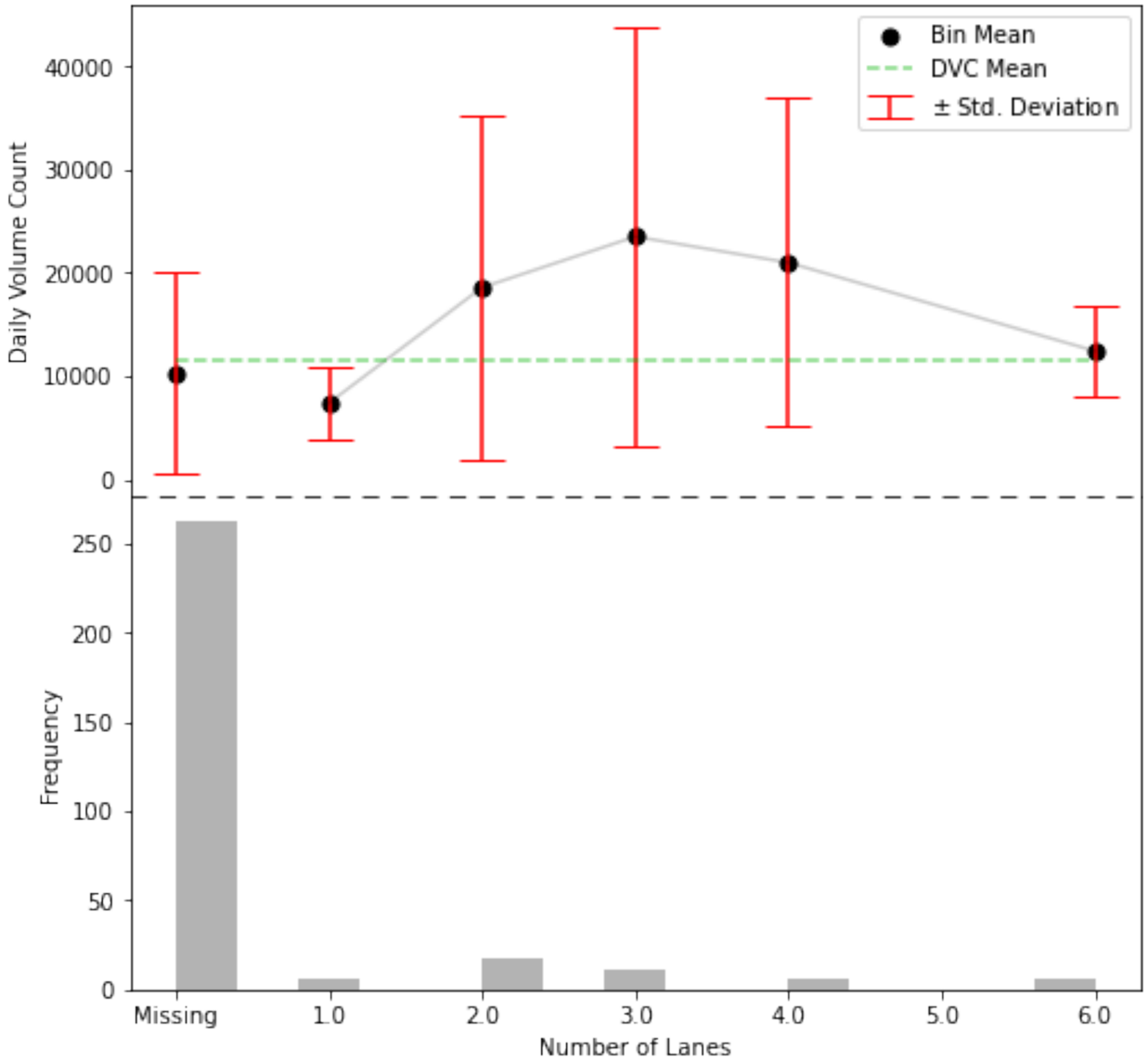
Preprocessing Method:

Handling missing data & Normalization

- The number of lanes is encoded with two values: (1) a bit indicating whether or not the true value is missing, and (2) the normalized number of lanes, or 0 if it's missing

Encoding:

Station →	Missing value? yes: **1**, no: **0**	**0** if missing; **normalized value** if not



Dataset Generation & Preprocessing

Contents

5. Direction

Calculation:

(x_1, y_1) : coordinates of starting intersection

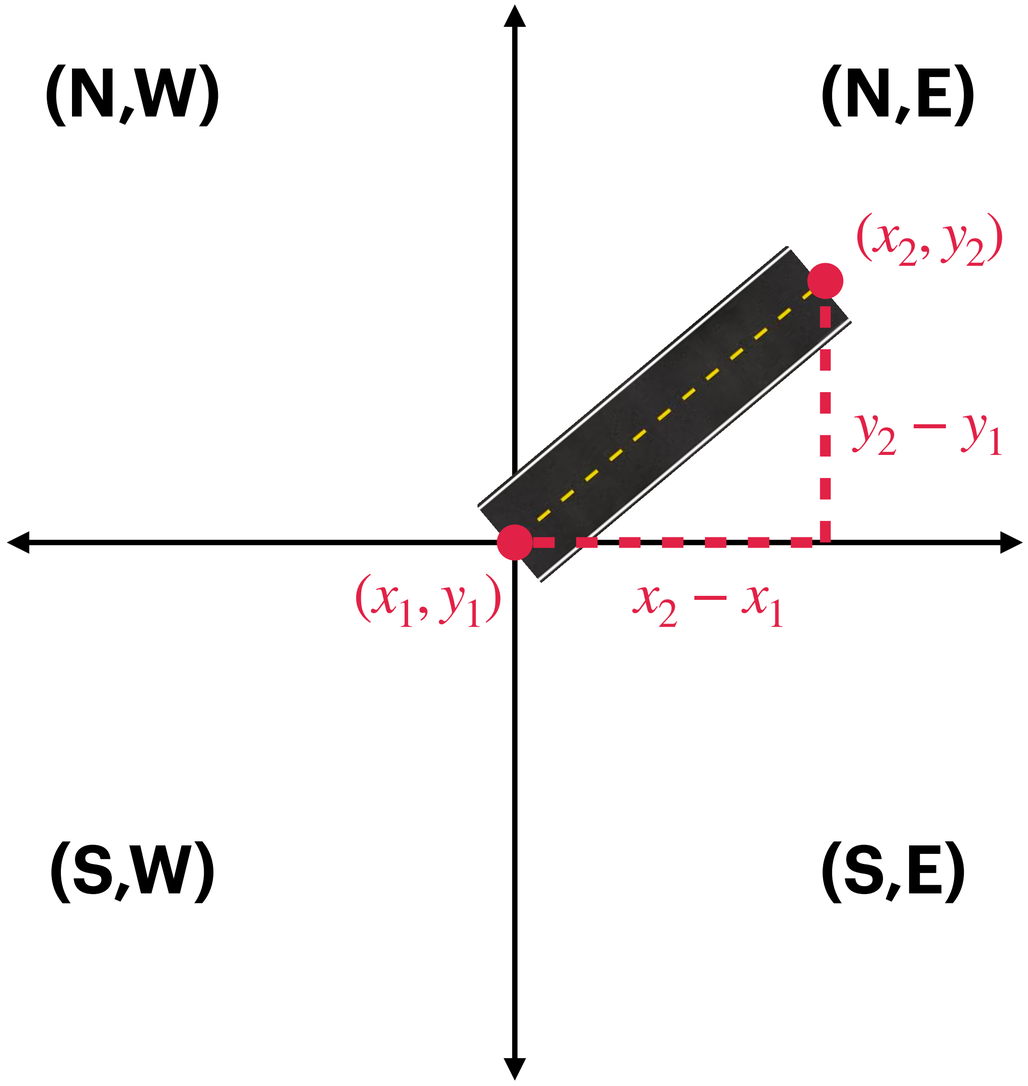
(x_2, y_2) : coordinates of ending intersection

direction = (N/S, E/W), where:

$$\text{N/S} = \begin{cases} \text{N} & \text{if } y_2 - y_1 \geq 0 \\ \text{S} & \text{otherwise} \end{cases}$$

$$\text{E/W} = \begin{cases} \text{E} & \text{if } x_2 - x_1 \geq 0 \\ \text{W} & \text{otherwise} \end{cases}$$

Hypothesis: Road segments traveling in the same direction may share similar daily patterns in traffic volume.



5. Direction

Observation:

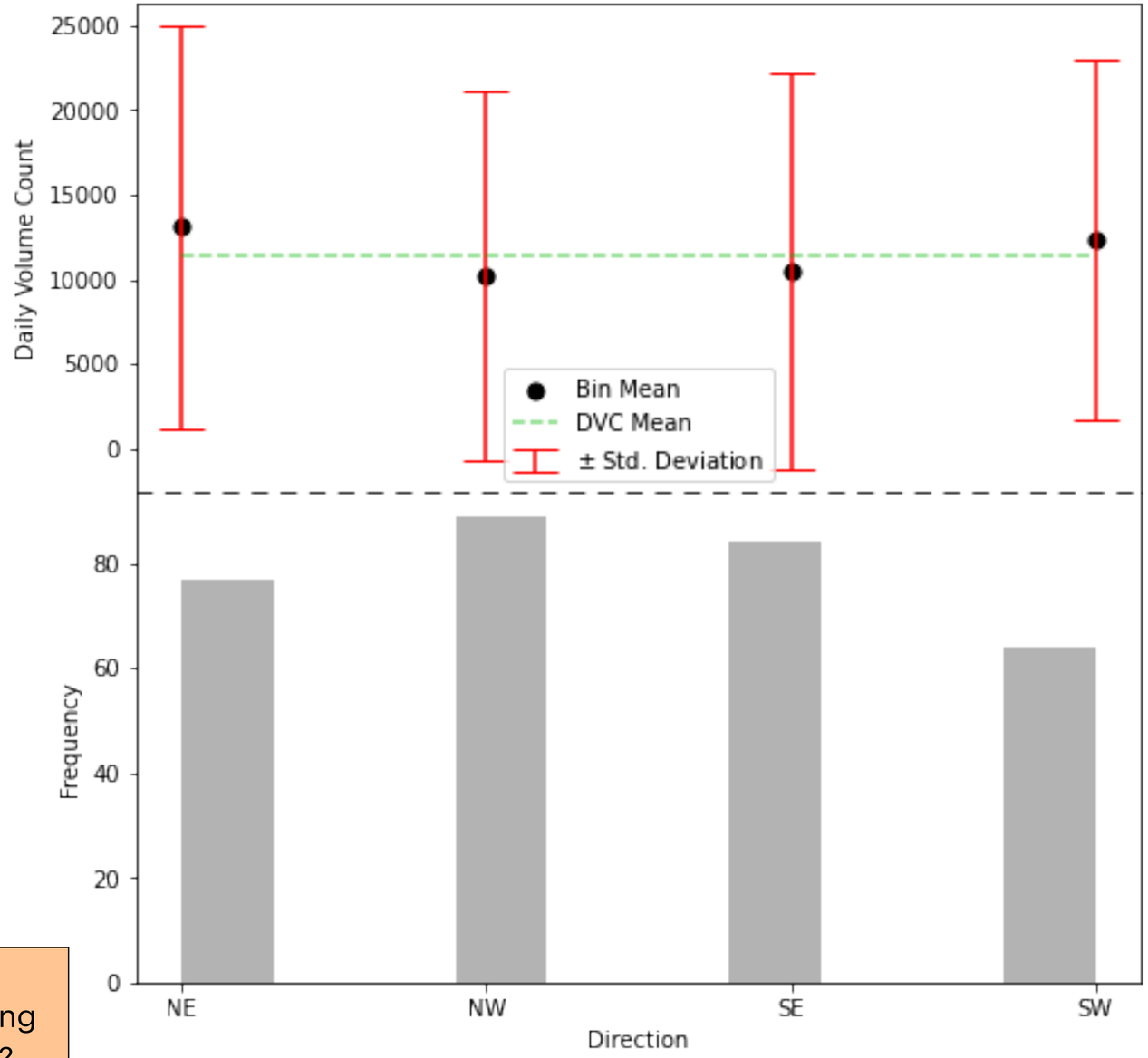
- Relatively uniform distribution
- NE and SW have slightly higher traffic volume, while NW and SE have slightly lower volume. Very minimal difference.

Evaluation: Direction not a good predictor of traffic volume in itself, but may serve to be a useful splitter in Decision Trees or Random Forests.

Preprocessing Method: None

Encoding: yes: **1**, no : **0**

Station →	Traveling North?	Traveling South?	Traveling East?	Traveling West?



6. Arctan (radians)

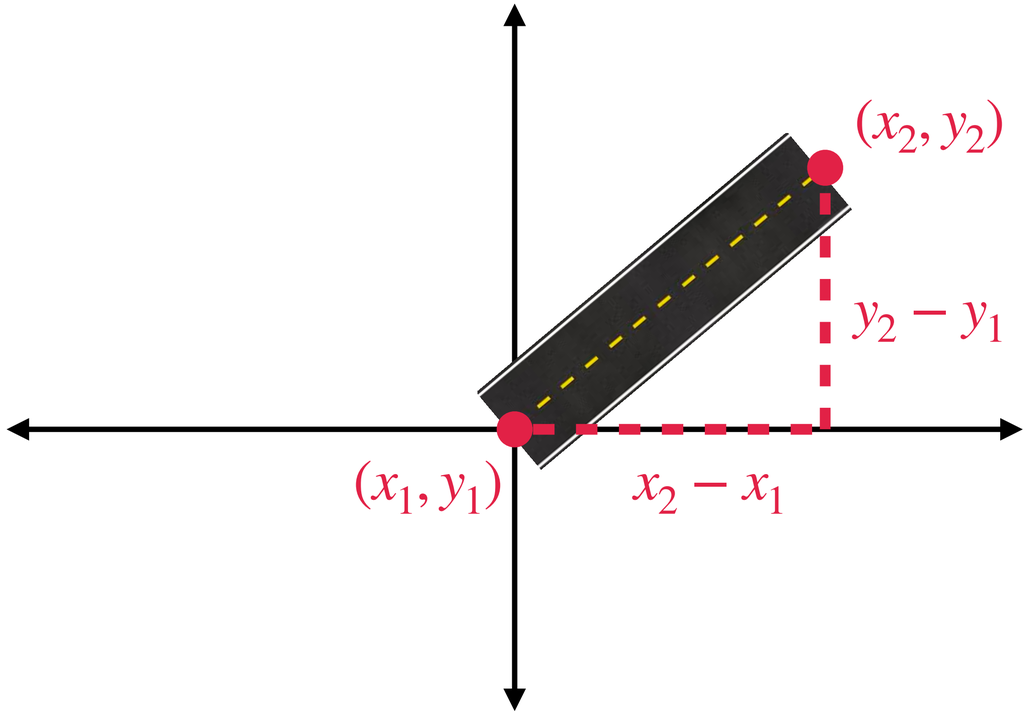
Calculation:

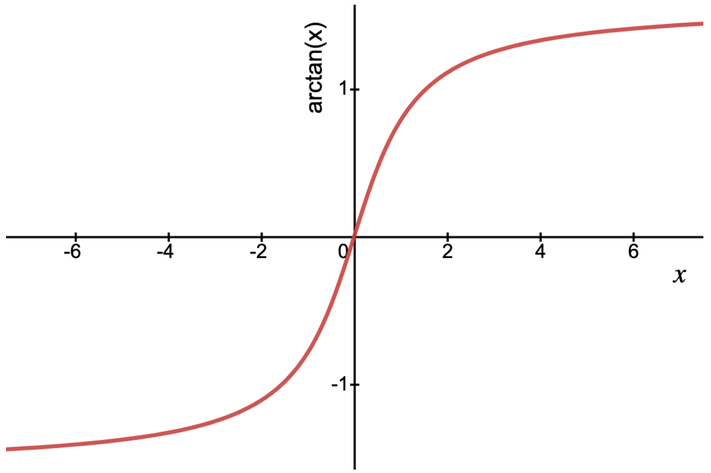
(x_1, y_1) : coordinates of starting intersection

(x_2, y_2) : coordinates of ending intersection

$$\arctan = \begin{cases} \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) & \text{if } x_1 \neq x_2 \\ \pi/2 & \text{otherwise} \end{cases}$$

Hypothesis: Road segments tilted at similar angles (independent of direction) may be more likely to possess similar patterns in traffic volume.





6. Arctan (radians)

Observations:

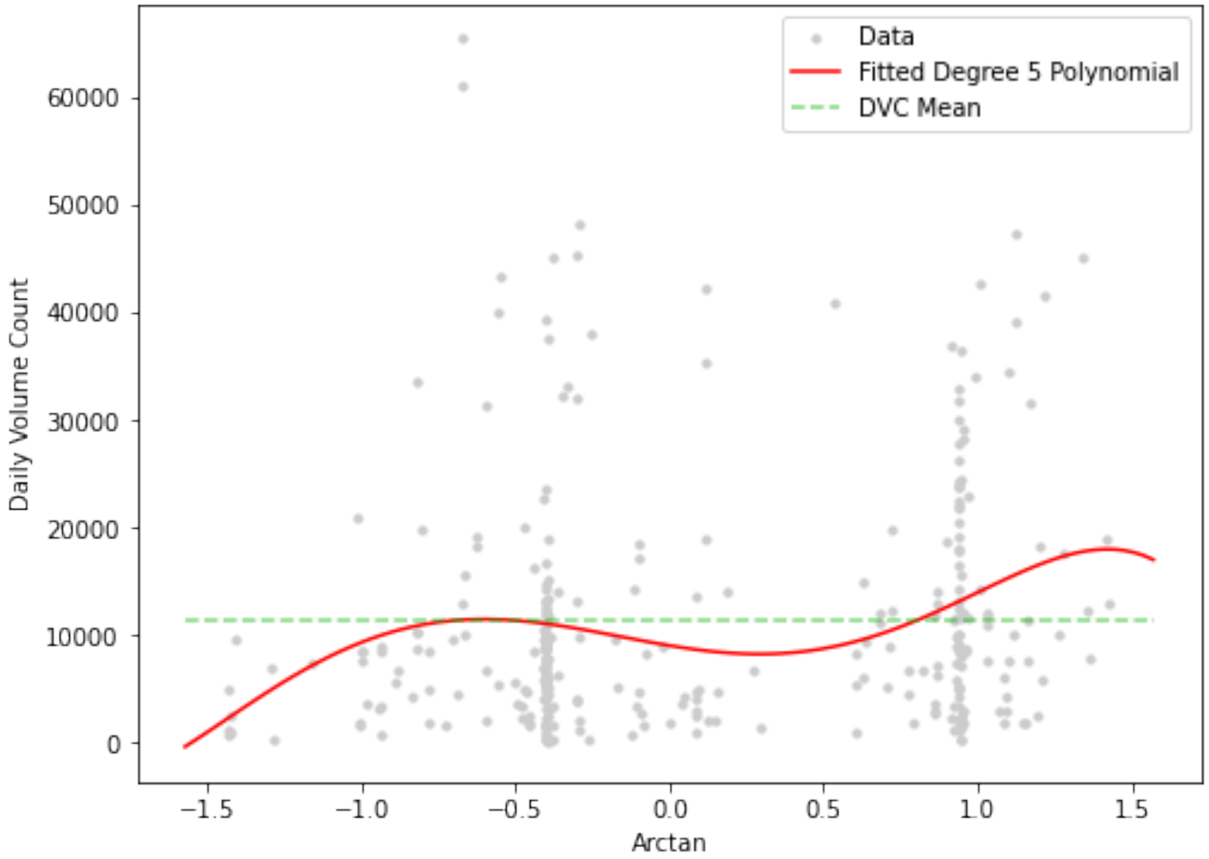
- Examples heavily clustered at -0.4 and 0.9
- Road segments in these clusters possess slightly higher traffic volume count

Evaluation: Road segments in one of the -0.4 or 0.9 clusters are slightly more likely to have a higher traffic volume count.

Preprocessing Method: None

Encoding:

Station → Arctan value



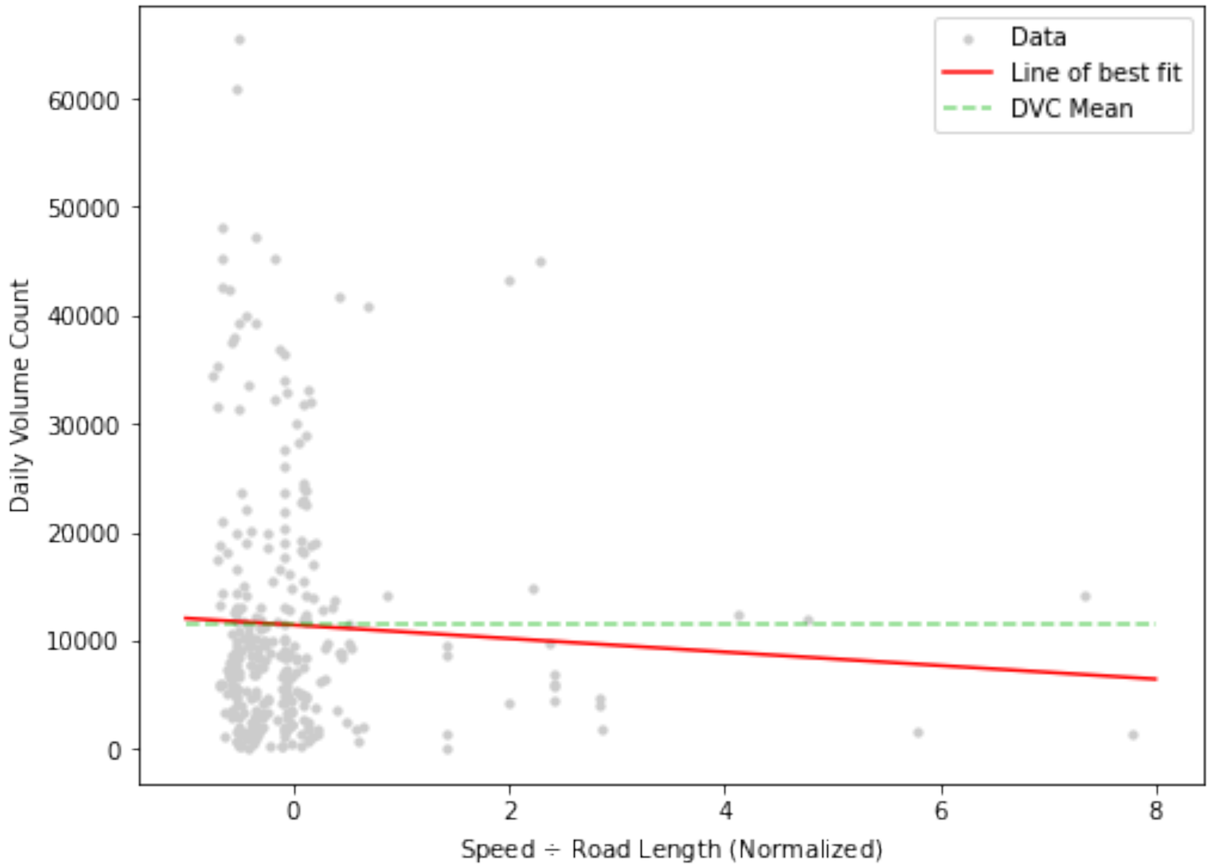
7. Speed Limit ÷ Road Length

Definition: A metric proportional to $1/\text{time}$

Calculation:

$$s \div r = \begin{cases} \dfrac{\text{speed\_limit}}{\text{road\_length}} & \text{if speed\_limit exists} \\ \dfrac{20}{\text{road\_length}} & \text{otherwise} \end{cases}$$

Hypothesis: Since vehicles would prefer to take faster roads, it would make sense that a metric inversely proportional to time is indirectly related to traffic volume.



7. Speed Limit ÷ Road Length

Observations:

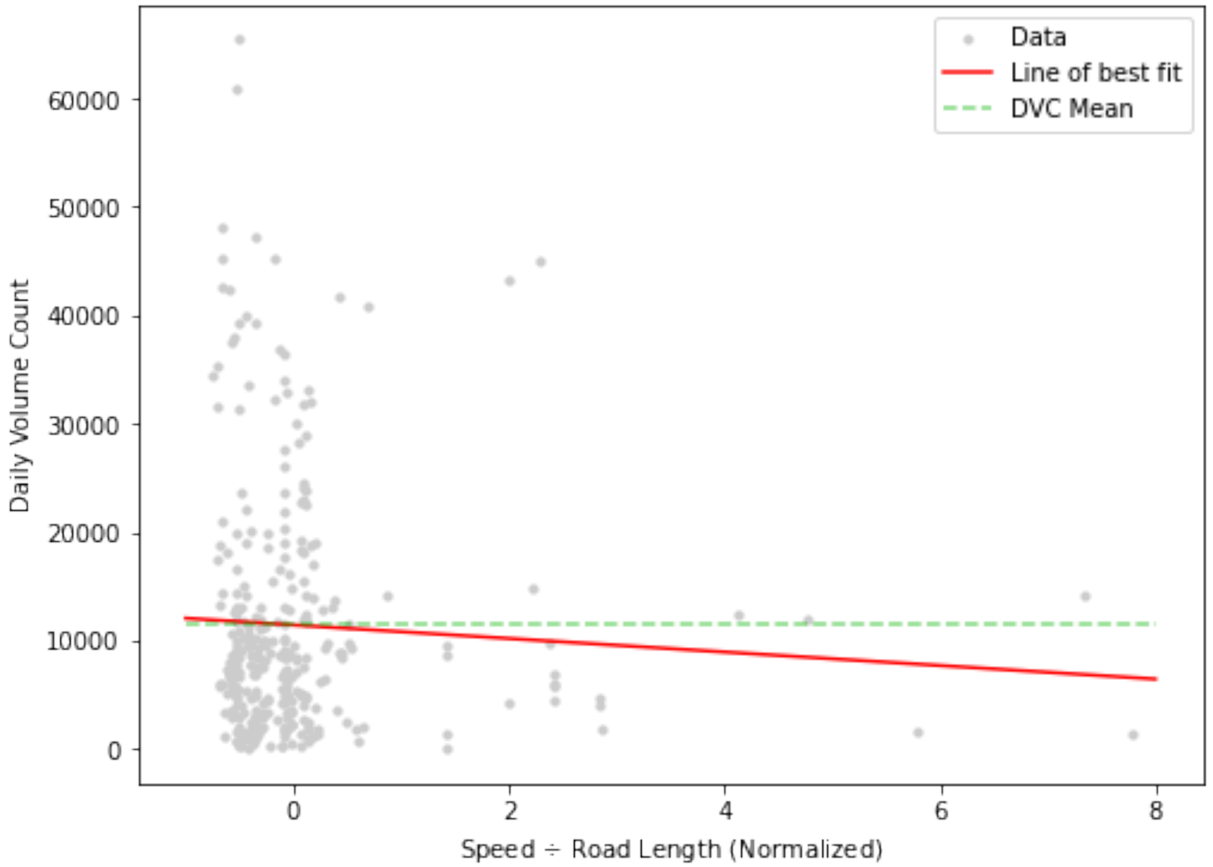
- S÷R values within a standard deviation of the mean appear to have no pattern
- Outliers (> +σ) appear to have lower traffic volume

Evaluation: Looks to be a poor predictor of traffic volume, unless an outlier happens to be sampled

Preprocessing Method: Normalization

Encoding:

Station → $(S/R - \mu_{train})/\sigma_{train}$



8. Road Length × Number of Lanes

Definition: A metric representing the amount of space on a road segment

Calculation:

$$r \times \ell = \begin{cases} \text{road\_length} \times \text{num\_lanes} & \text{if num\_lanes exists} \\ \text{road\_length} & \text{otherwise} \end{cases}$$

Hypothesis: Long road segments with many lanes should attract lots of vehicles, thus possess higher levels of traffic volume. There should be a direct relationship between this metric and traffic volume.

Lots of space

Little space

8. Road Length × Number of Lanes

Observations:

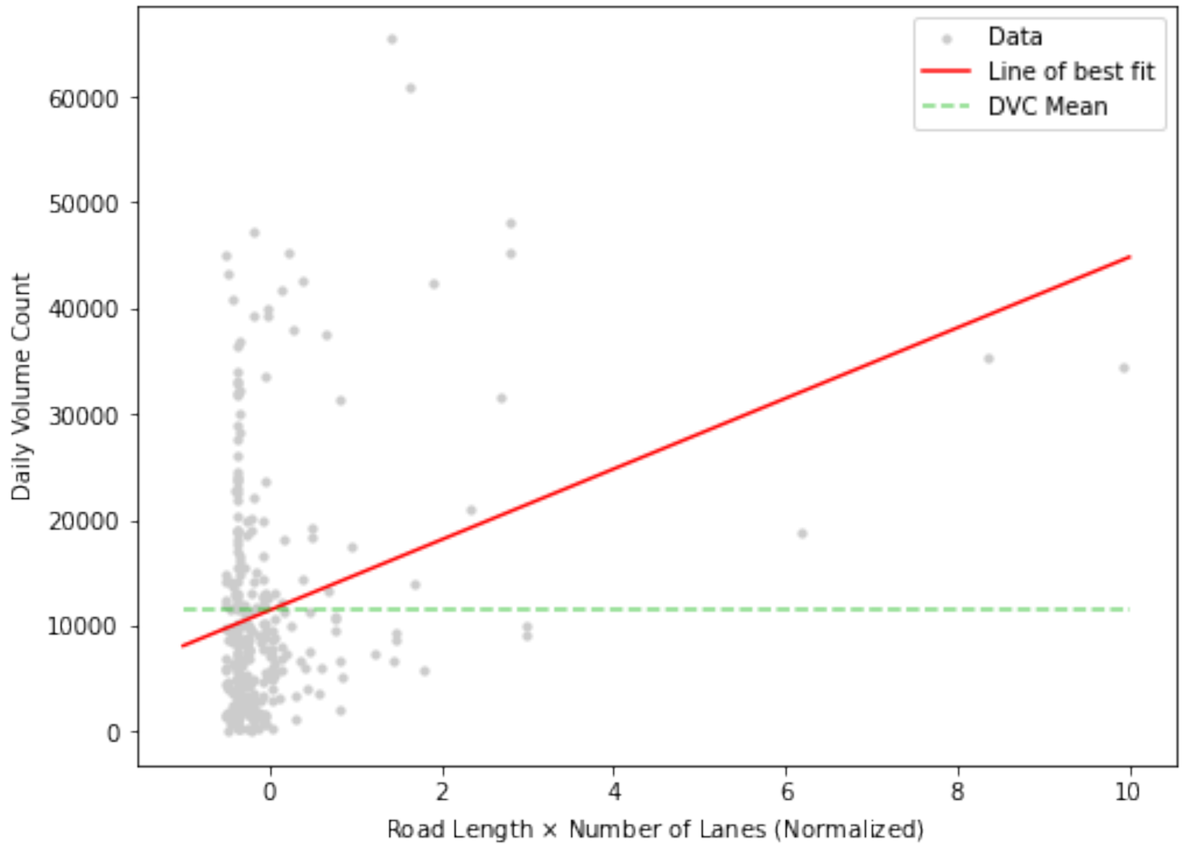
- R×L values within a standard deviation of the mean appear to have no pattern
- Outliers (> +σ) appear to have higher traffic volume

Evaluation: This metric seems to be a realivetly poor predictor of traffic volume, unless an outlier is sampled

Preprocessing Method: Normalization

Encoding:

Station → $(R^*L - \mu_{train})/\sigma_{train}$



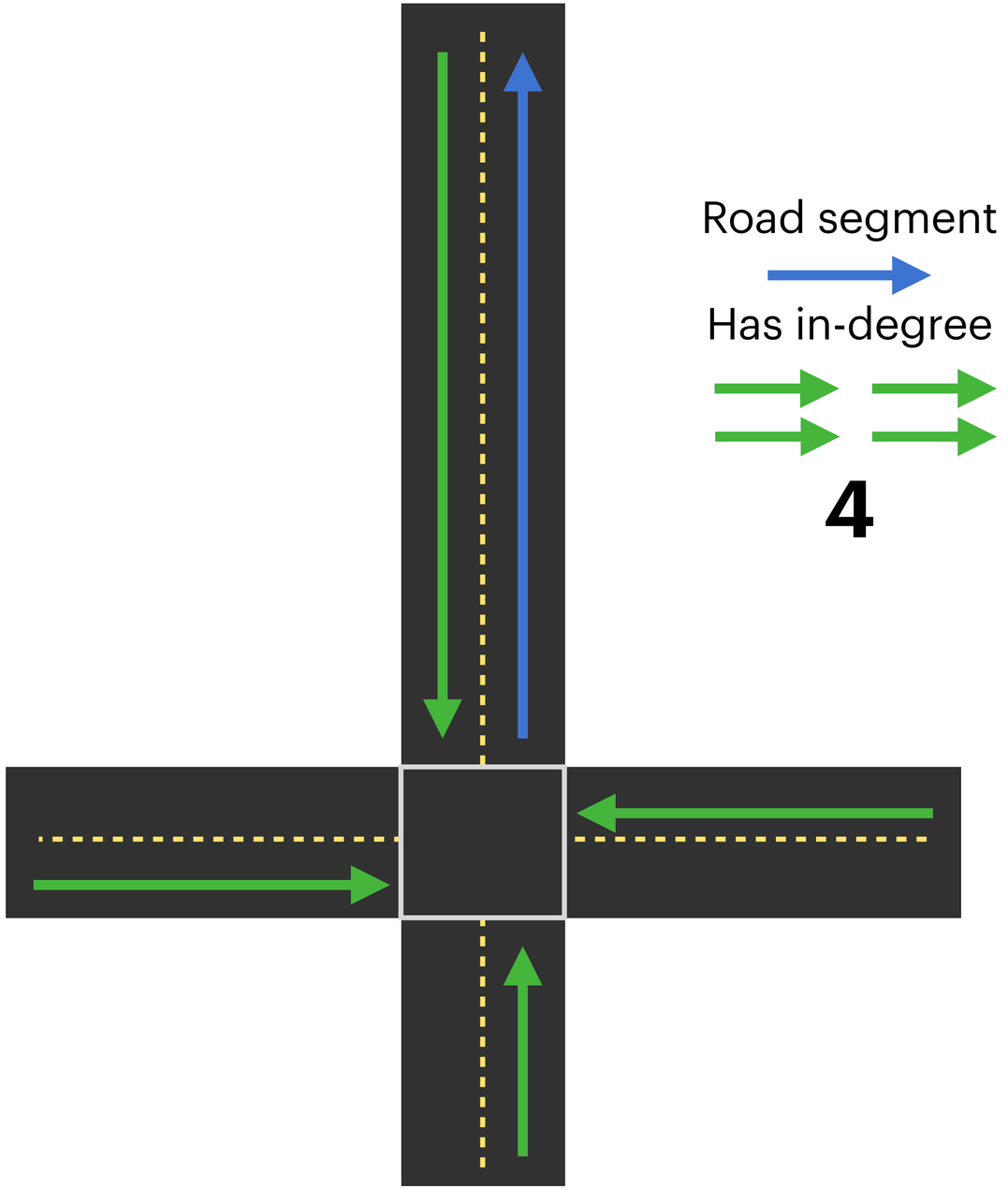
9. In-Degree

Definition: The number of road segments (1) adjacent to, and (2) directed at the starting intersection of the road segment.

Computation:

Networkx function **in_degree** [5]

Hypothesis: Road segments with higher in-degrees have more incoming traffic, thus receive higher levels of traffic volume.



9. In-Degree

Observations:

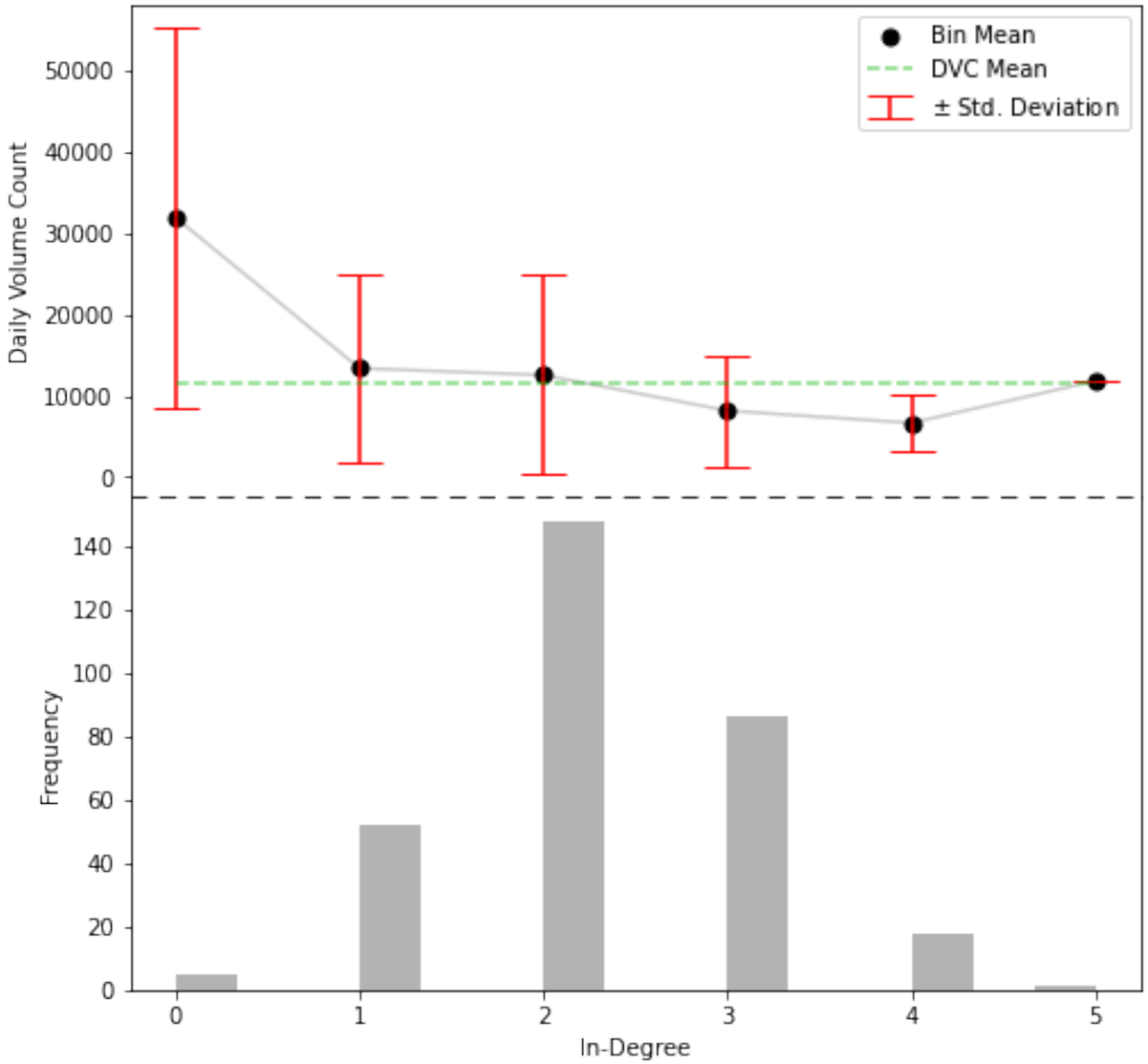
- Symmetric distribution centered at 2
- Apparent inverse relationship between in-degree and traffic volume

Evaluation: The hypothesis looks to be the opposite of reality; as the in-degree increases, the traffic volume appears to decrease

Preprocessing Method: Normalization

Encoding:

Station → $(\text{ID} - \mu_{\text{train}})/\sigma_{\text{train}}$



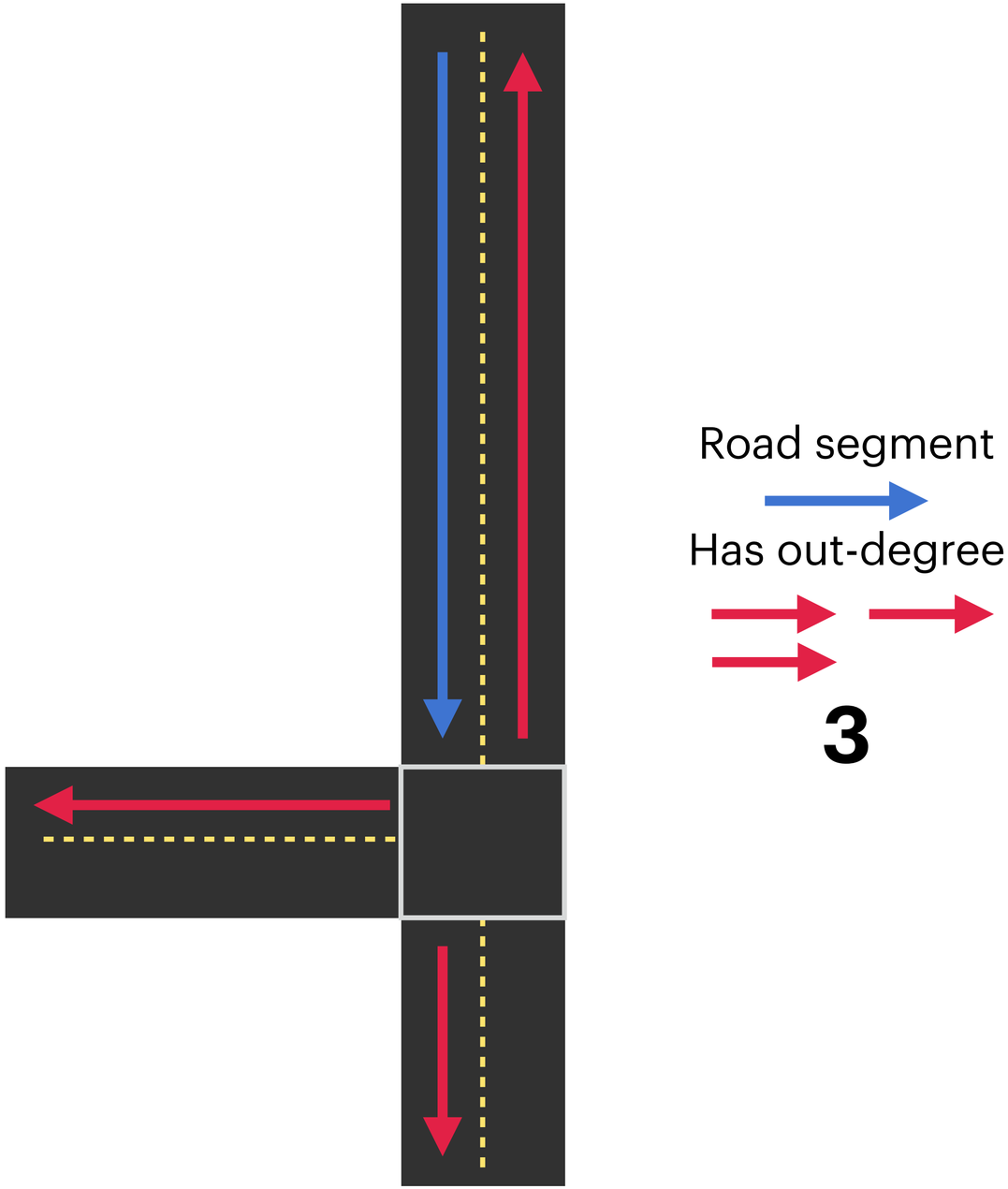
10. Out-Degree

Definition: The number of road segments (1) adjacent to, and (2) directed away from the ending intersection of the road segment.

Computation:

Networkx function **out_degree** [6]

Hypothesis: Road segments with higher out-degrees have more places to go, thus receive more traffic and consequently higher levels of traffic volume.



10. Out-Degree

Observations:

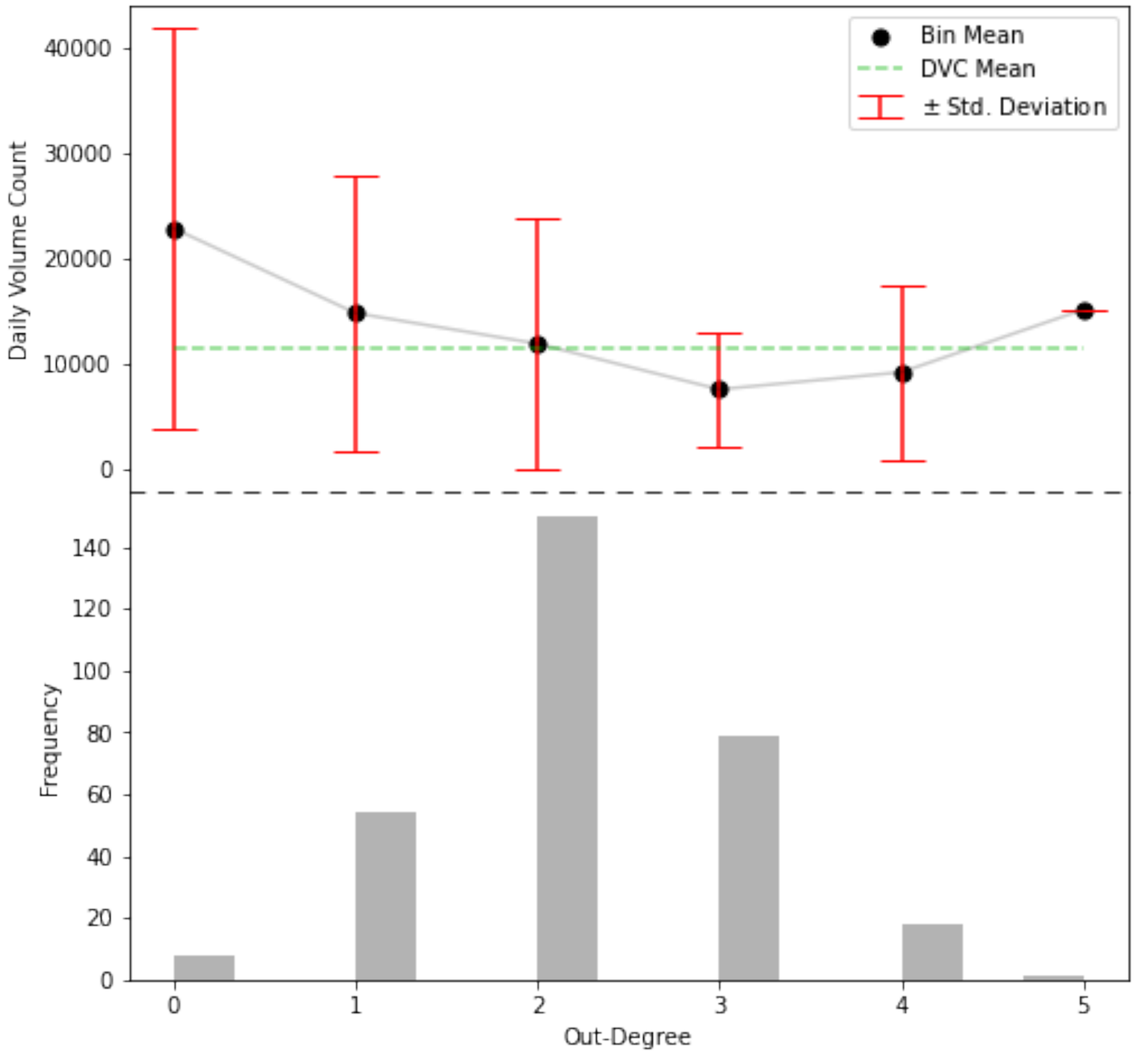
- Symmetric distribution centered at 2
- Apparent inverse relationship between out-degree and traffic volume, until 4

Evaluation: The hypothesis looks to be the opposite of reality; as the out-degree increases, the traffic volume appears to decrease up to 4, then increases

Preprocessing Method: Normalization

Encoding:

Station → $(\text{OD} - \mu_{\text{train}})/\sigma_{\text{train}}$



11. Betweenness Centrality

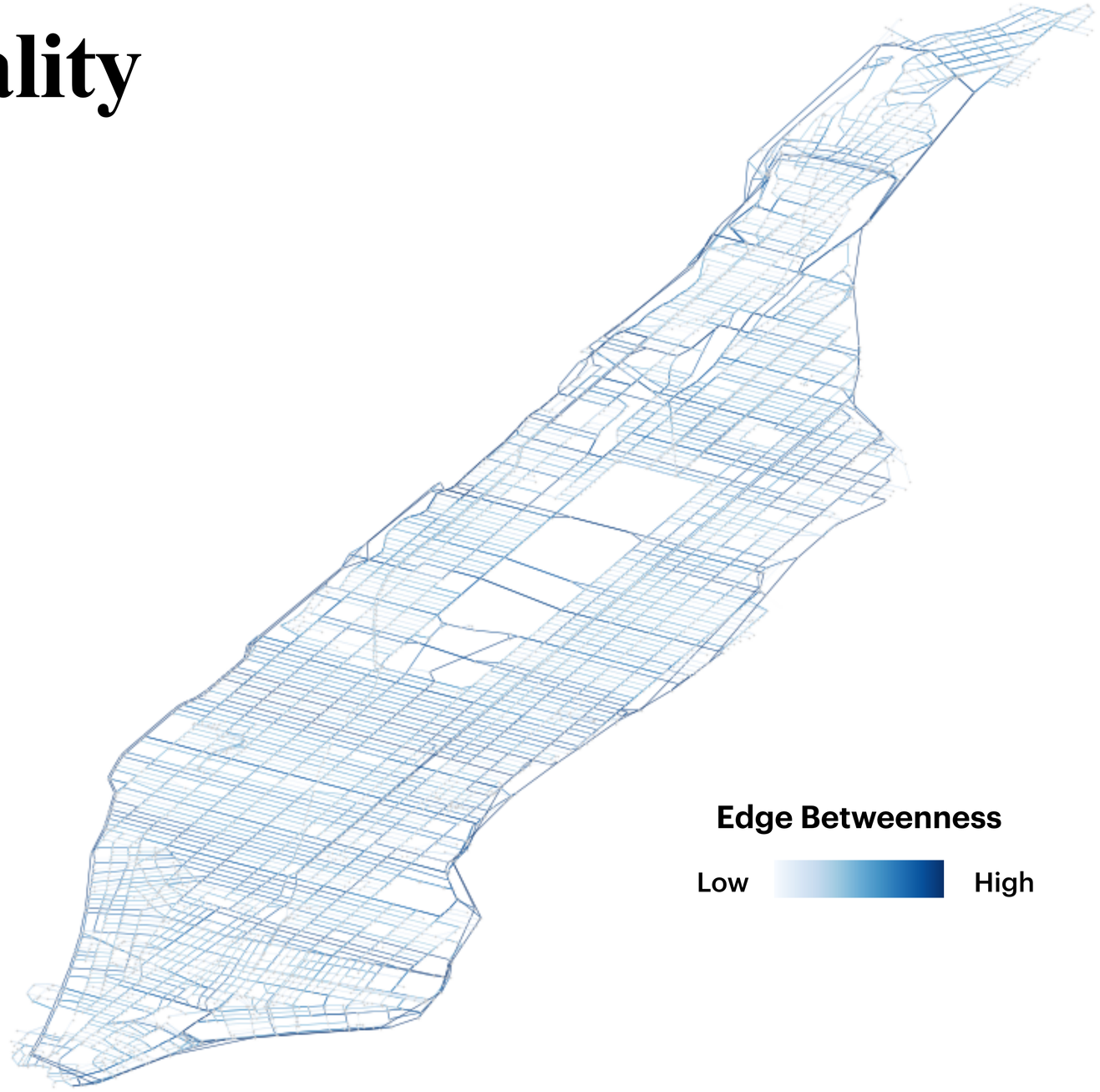
Definition: The percentage of shortest paths that run through a road segment

Calculation:

$$\text{bet}(e) = \sum_{s \neq t} \frac{\sigma(s,t,e)}{\sigma(s,t)}$$

$\sigma(s,t,e) =$ number of shortest paths between s and t including edge e

Hypothesis: If a road segment has a high betweenness centrality, then it's important to the road network structure, so it should receive higher levels of traffic volume.



11. Betweenness Centrality

Observations:

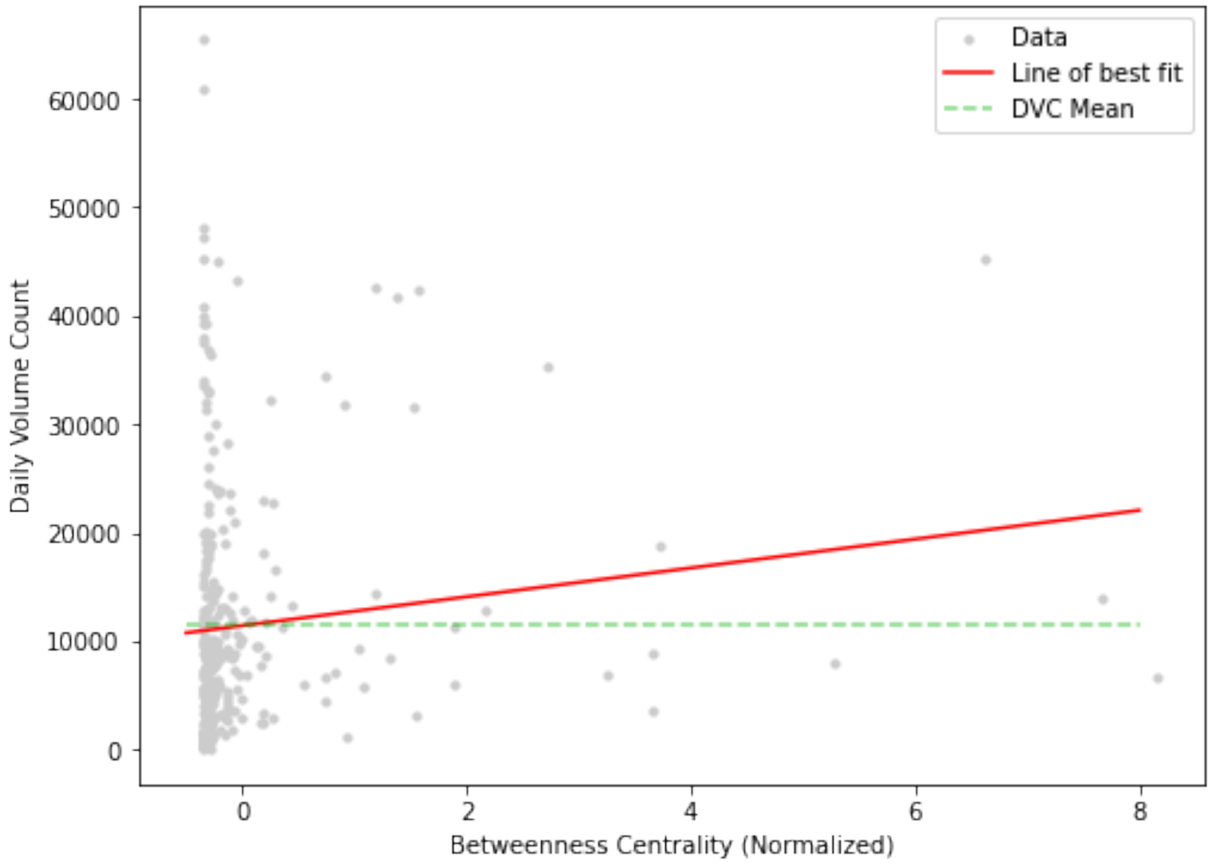
- Values below the mean appear to be very spread out
- Values above the mean appear to have higher traffic volume

Evaluation: Not a great predictor of traffic volume overall, but some pattern emerges with data above the mean

Preprocessing Method: Normalization

Encoding:

Station → $(\text{bet} - \mu_{\text{train}})/\sigma_{\text{train}}$



12. Closeness Centrality

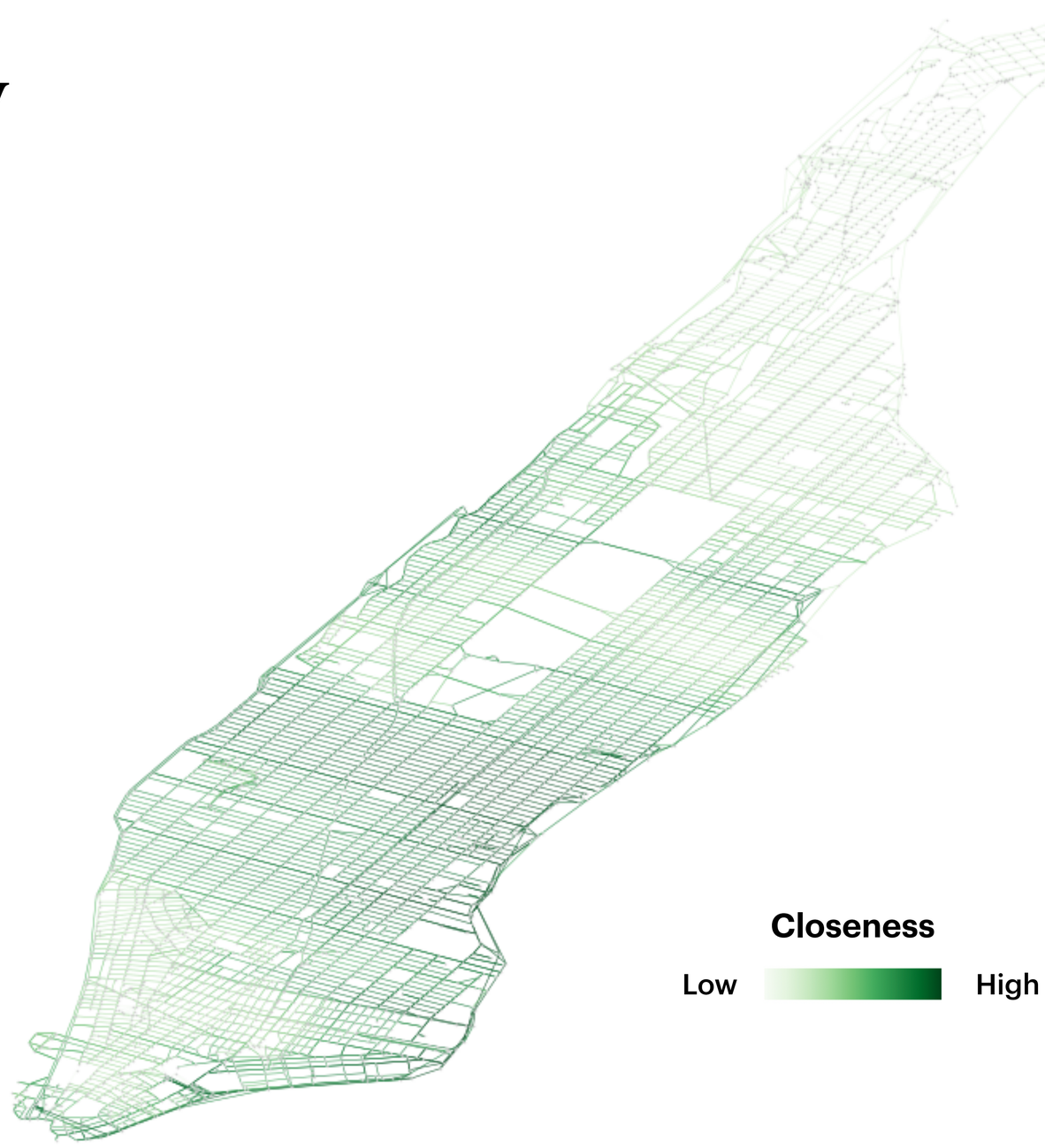
Definition: The average of the shortest paths between the road segment and every other road segment

Calculation:

$$\text{clos}(e) = \frac{1}{|E| - 1} \sum_{e' \neq e} d(e, e')$$

$d(e, e') =$ the length of the shortest path between edges e and e'

Hypothesis: If a road segment has a high closeness centrality, then it's important to the road network structure, so it should receive higher levels of traffic volume.



12. Closeness Centrality

Observations:

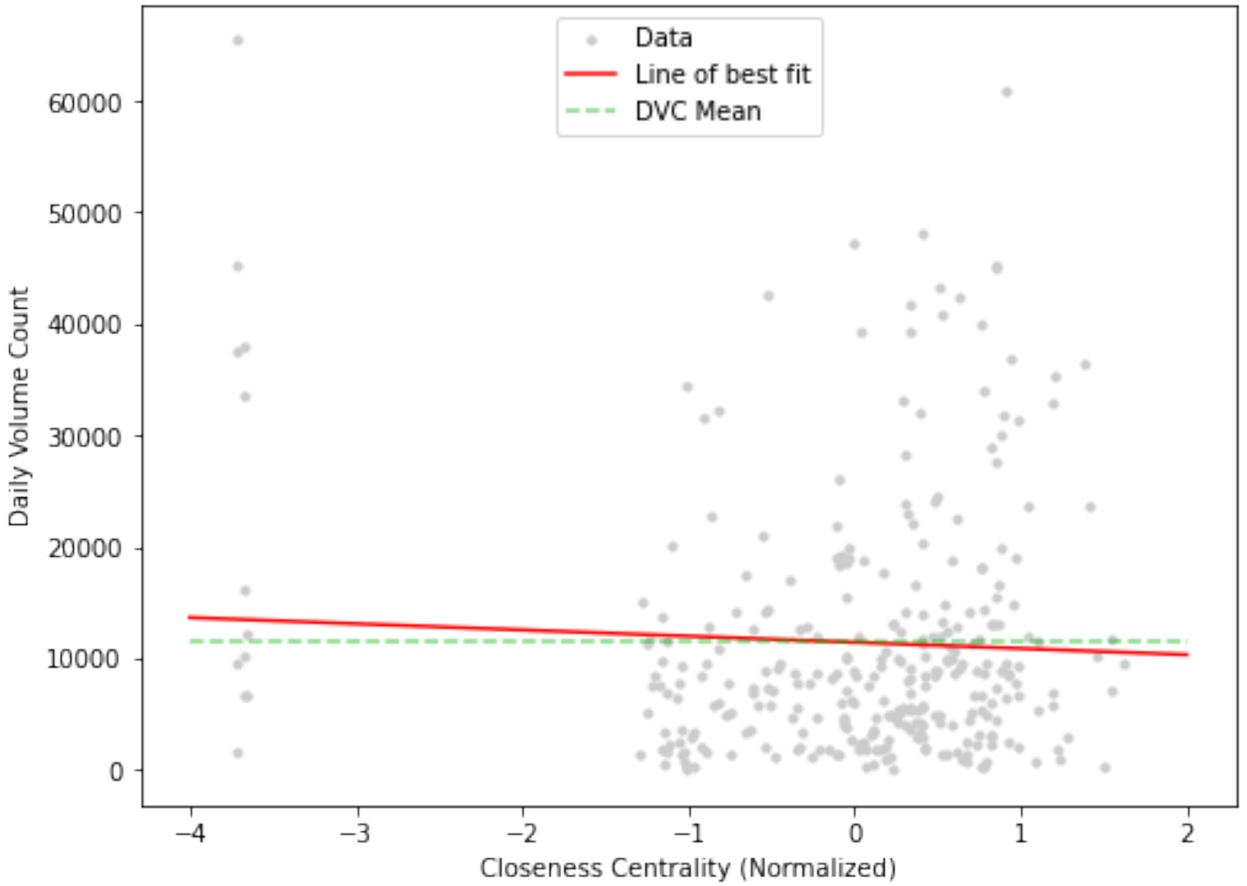
- Values close to the mean appear to follow no pattern
- Outliers (< -σ) appear to have higher traffic volume

Evaluation: Very poor predictor of traffic volume in isolation, although a slight pattern emerges with outliers below the mean

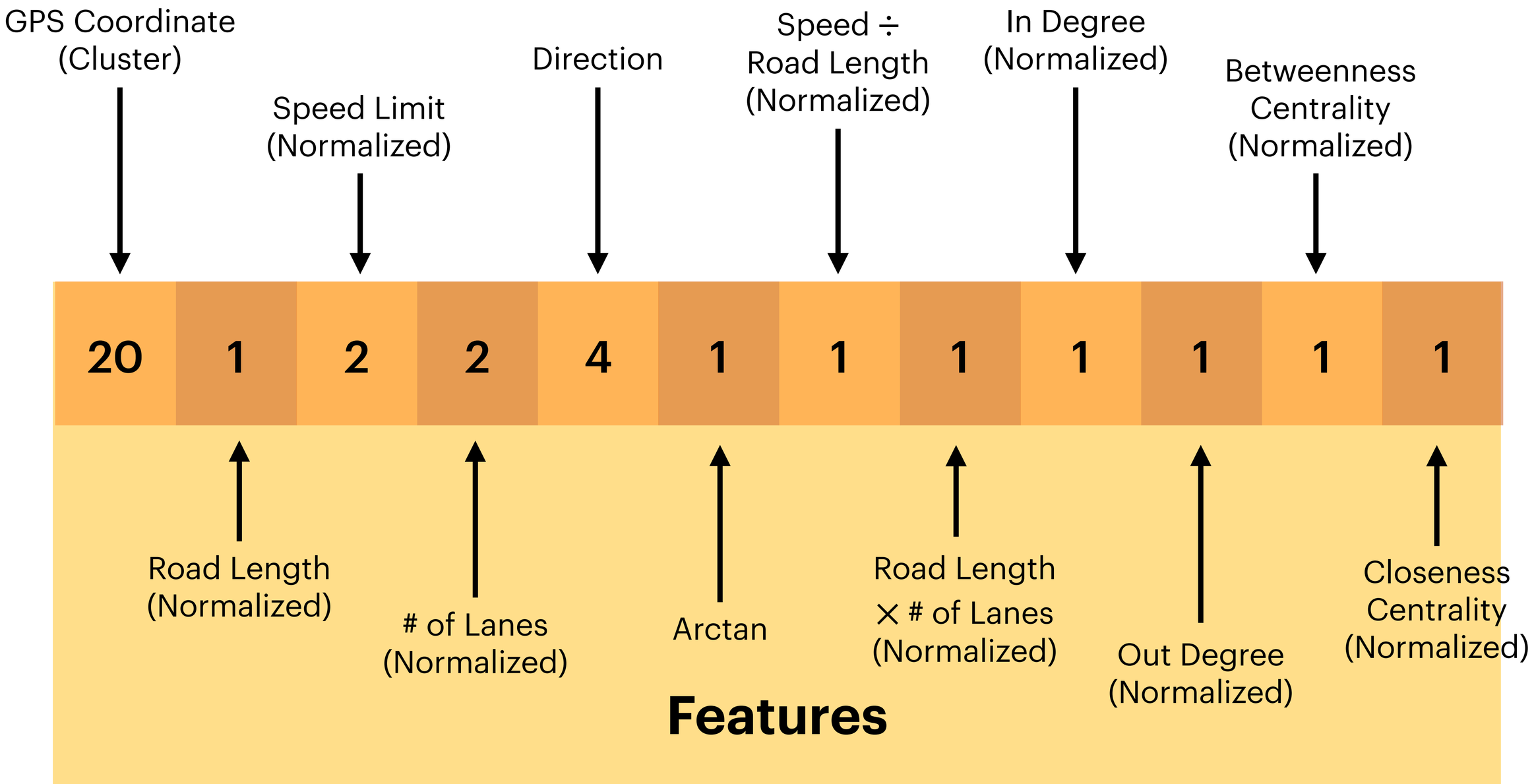
Preprocessing Method: Normalization

Encoding:

Station → $(\text{clos} - \mu_{\text{train}})/\sigma_{\text{train}}$



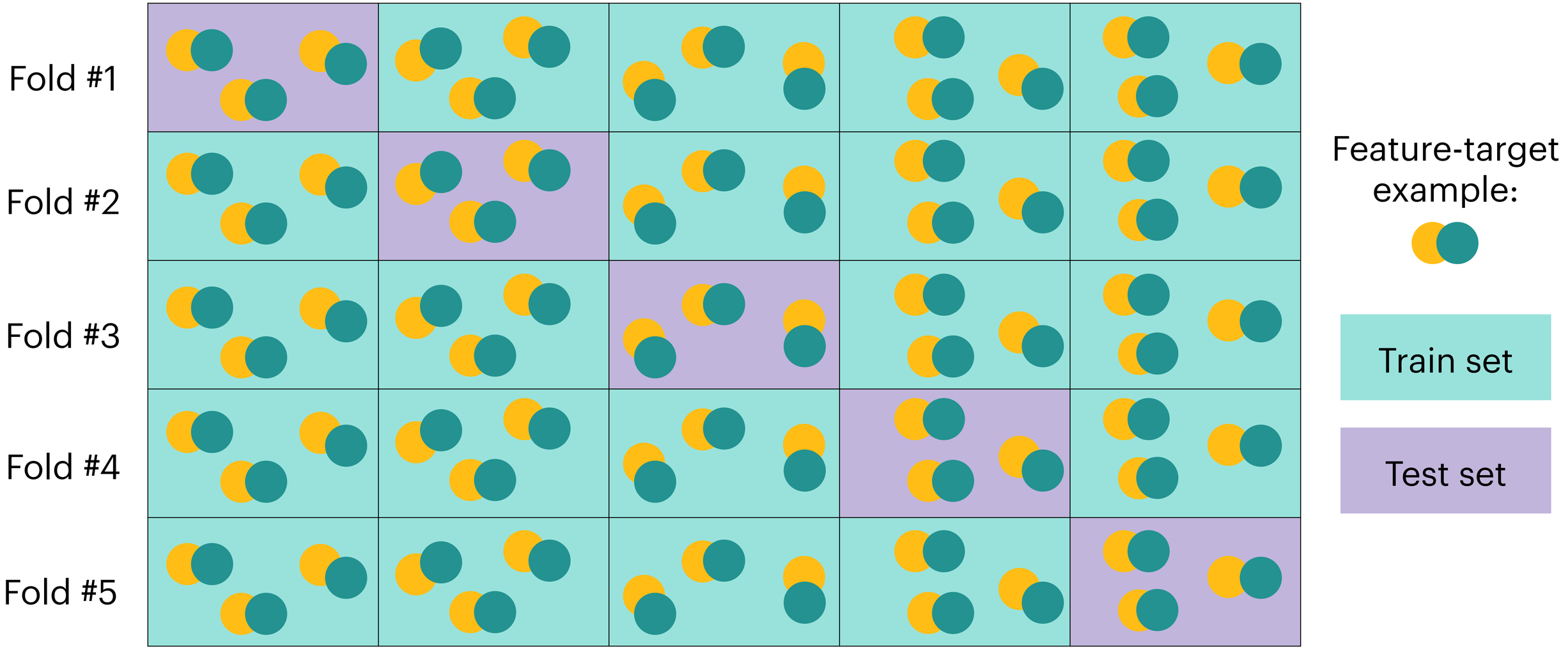
Feature Vector Encoding

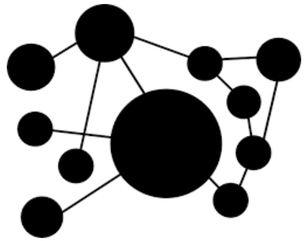


Dataset Generation & Preprocessing

Contents

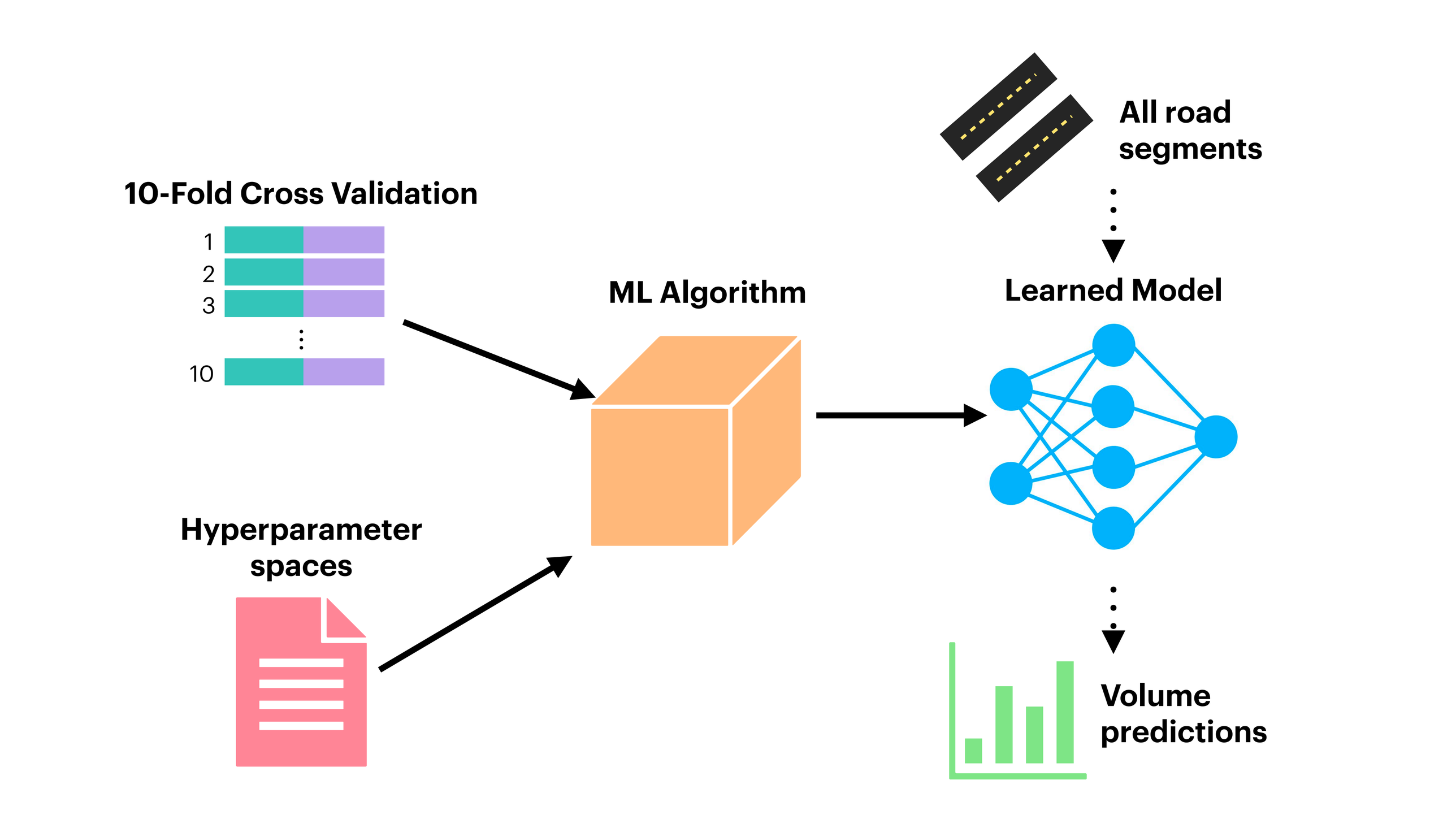
5-Fold Cross Validation Example



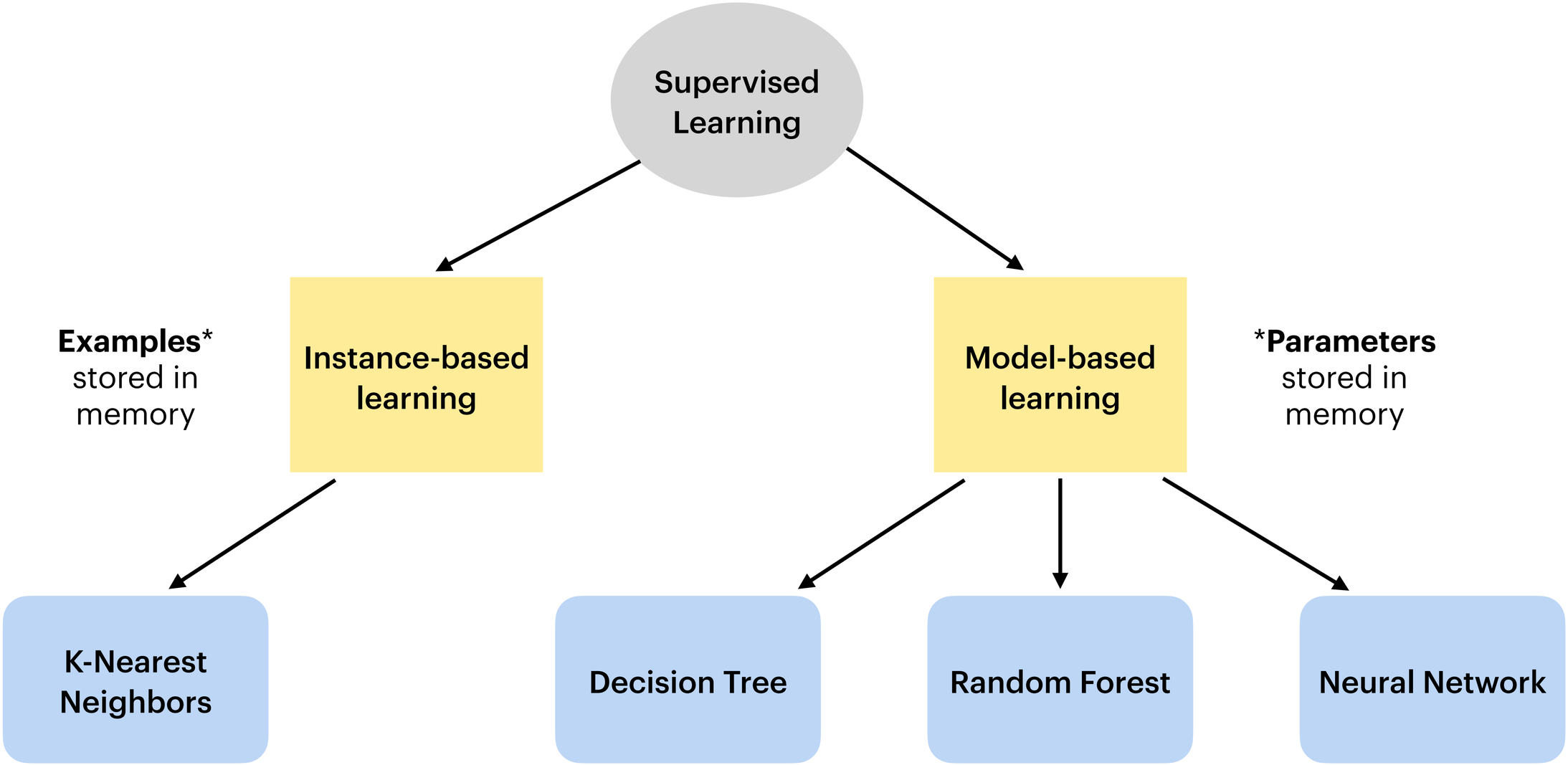


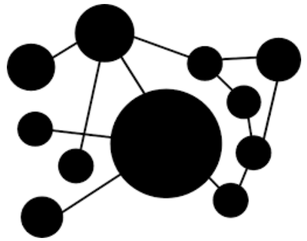
Predictive Modeling

Contents


10-Fold Cross Validation
1
2
3
10
Hyperparameter spaces
ML Algorithm
All road segments
Learned Model
Volume predictions

Hierarchy of ML Algorithms





Predictive Modeling

Contents

Overview of the KNN Algorithm

Training:

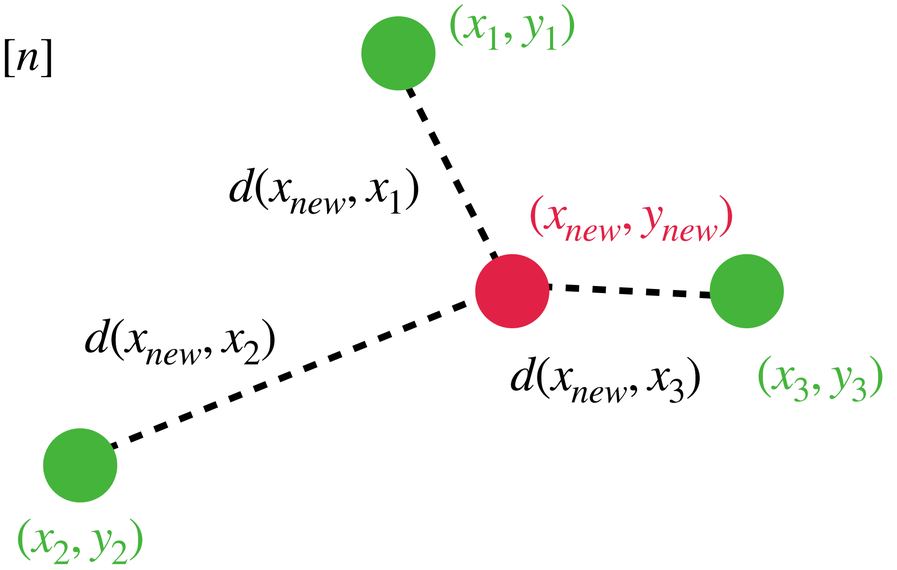
- No training; instead, store each train example $\{(x_i, y_i)\}_{i \in [n]}$ in memory for testing

Testing:

- Algorithm defined as follows:

1. Input: feature x_{new} (goal is to predict y_{new})
2. Calculate $d(x_{new}, x_i)$ for each x_i in the train set and some distance metric d. Find the K examples yielding the smallest distances: $(x_1^*, y_1^*), \ldots, (x_1^*, y_K^*)$
3. Set $\hat{y}_{new} =$ mean or median of $y_1^*, y_2^*, \ldots, y_K^*$

$K = 2$ **Example:**



$$\hat{y}_{new} = \frac{1}{2}(y_1 + y_3)$$

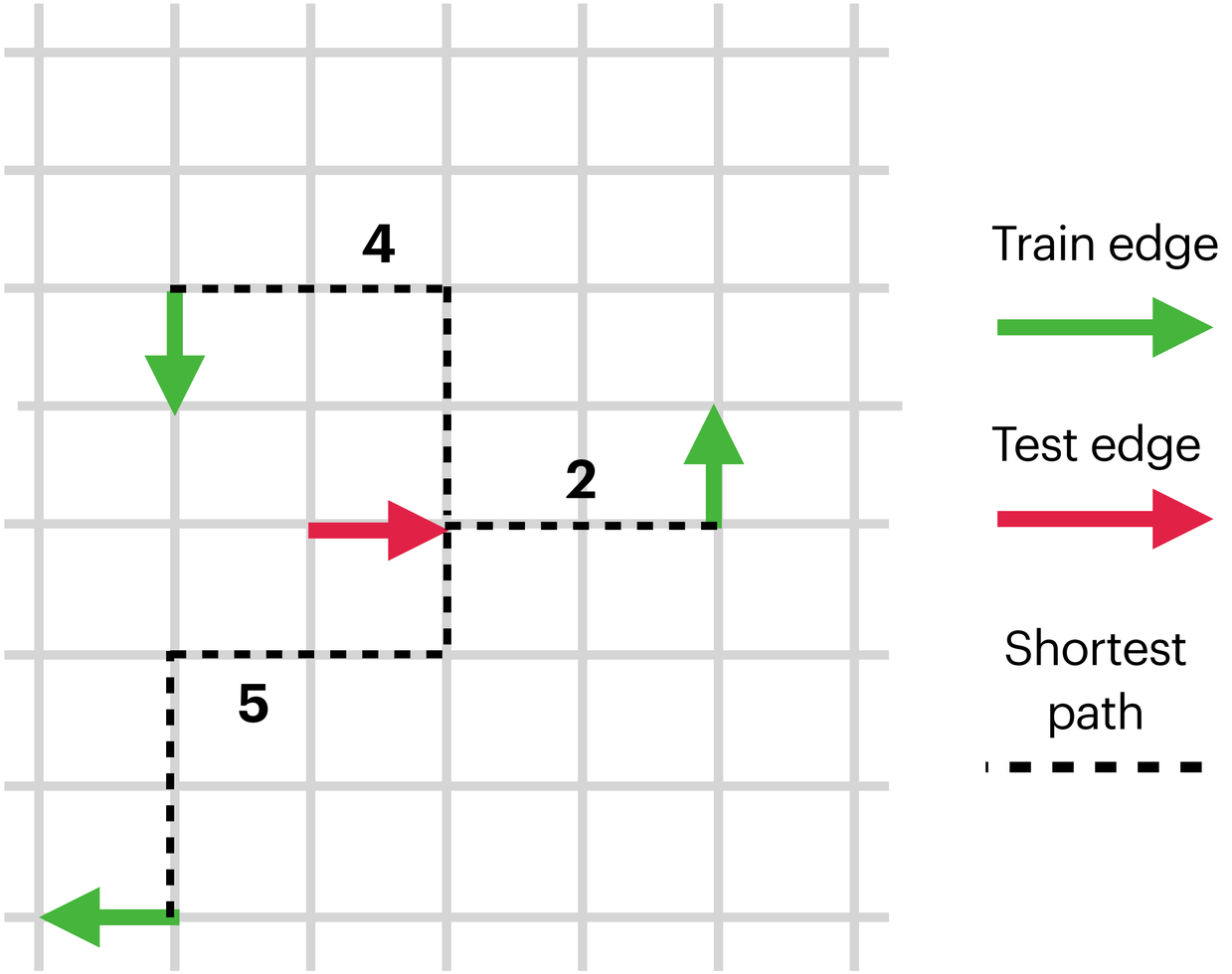
Review of Distance Metric d

Conventional use:

- Euclidean distance (L2 Norm)
  - $d(x_1, x_2) = \|x_1 - x_2\|_2$

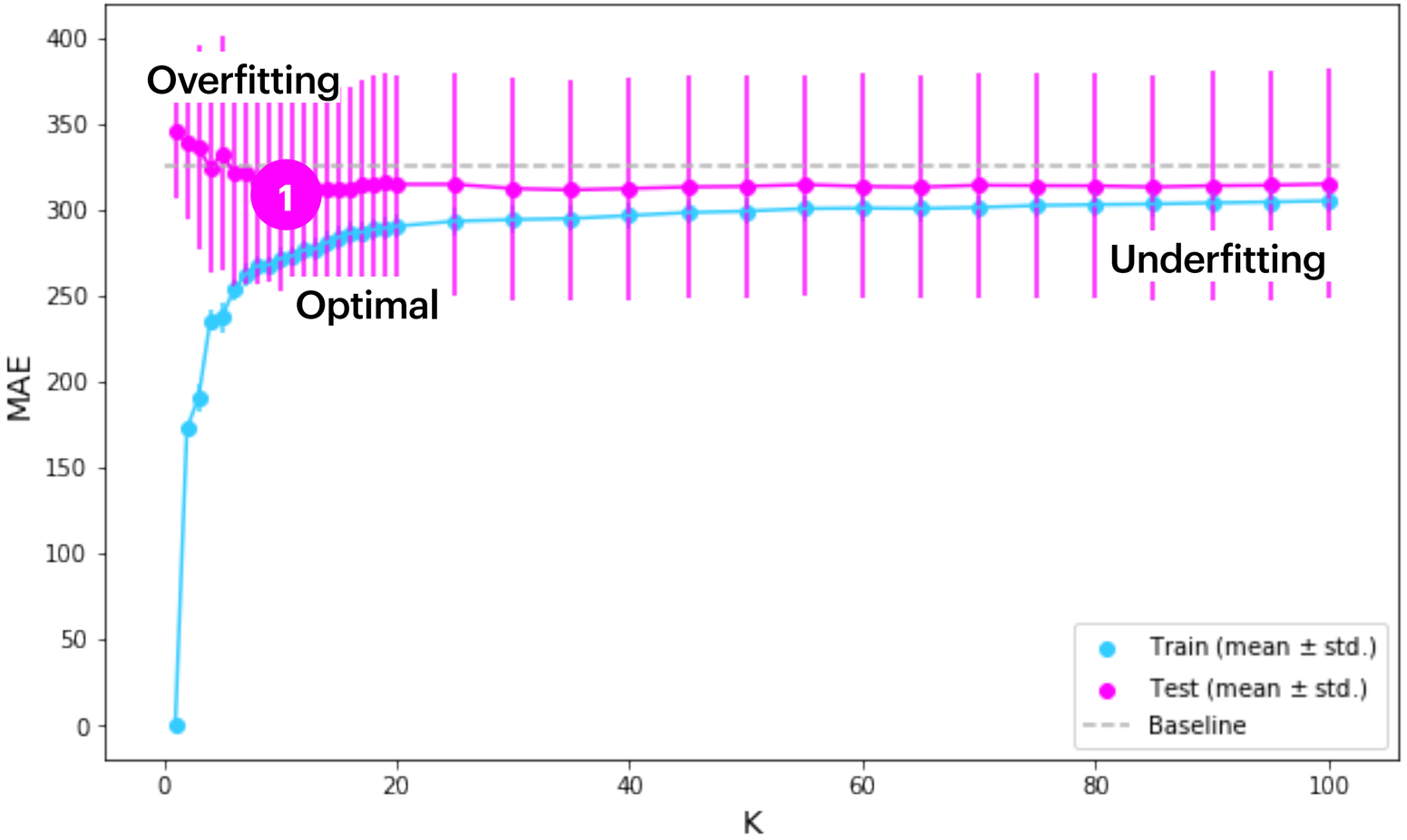
Our use:

- Length of shortest path in network between edges
- Features = directed edges
- $d(e_1, e_2) =$ length of shortest path between edges e_1 and e_2
- Implemented with NetworkX's **single_source_djikstra** function [7]



Hyperparameter Tuning & Results

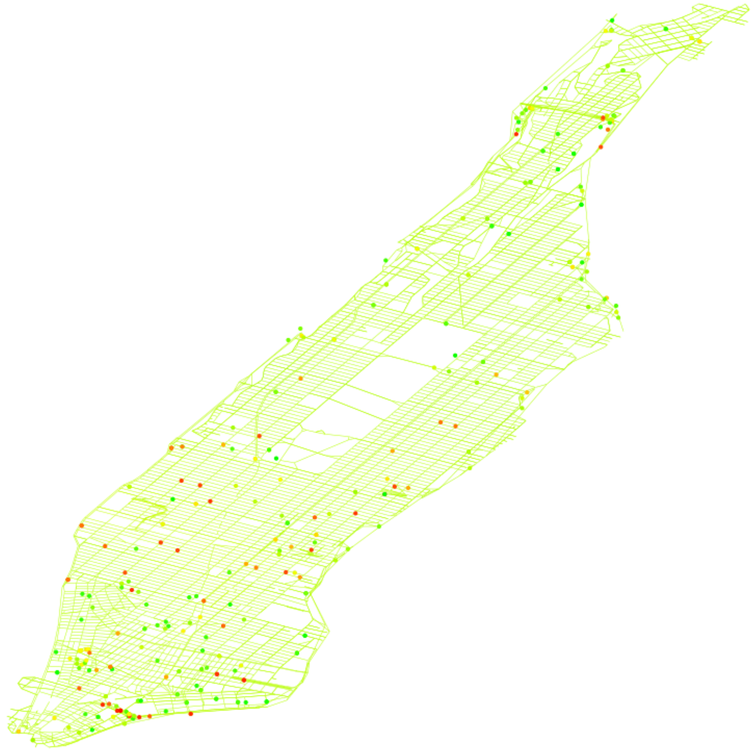
Hyperparameter Search: $K \in 1$ to 20 in steps of 1, 25 to 100 in steps of 5



Best Result:

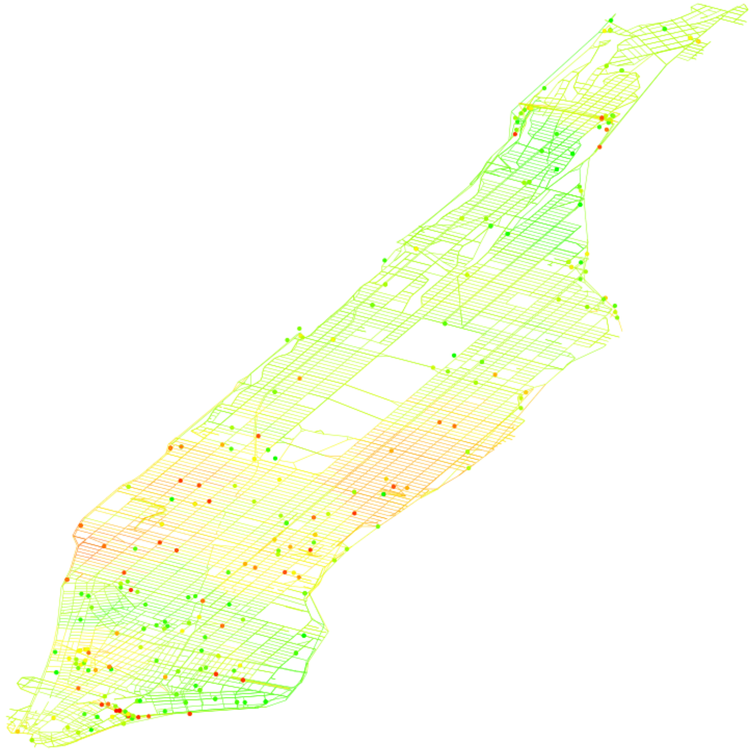
1 K = 10 **: 309.0**

Animations of Traffic Predictions



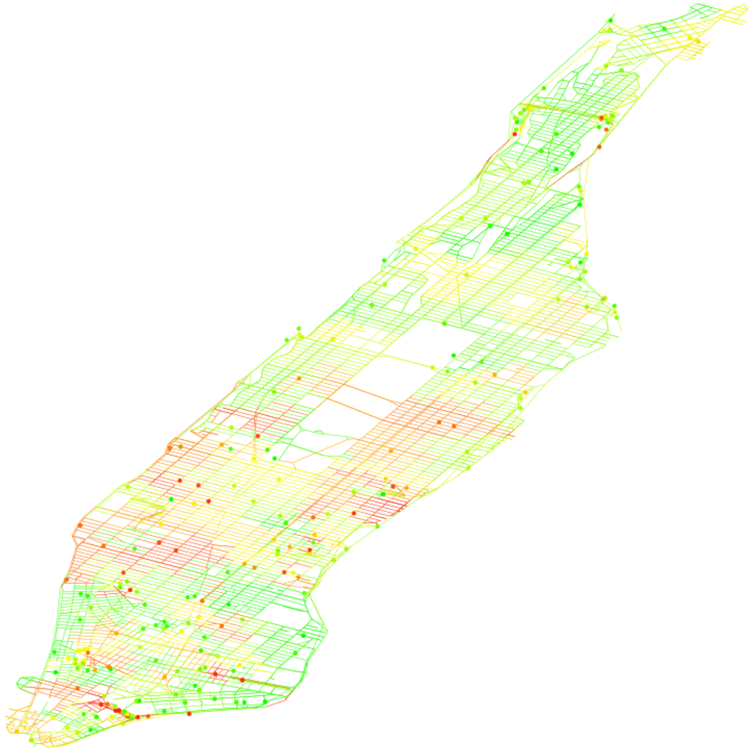
Underfitting

K = 150



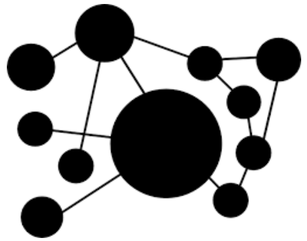
Optimal

K = 10



Overfitting

K = 1



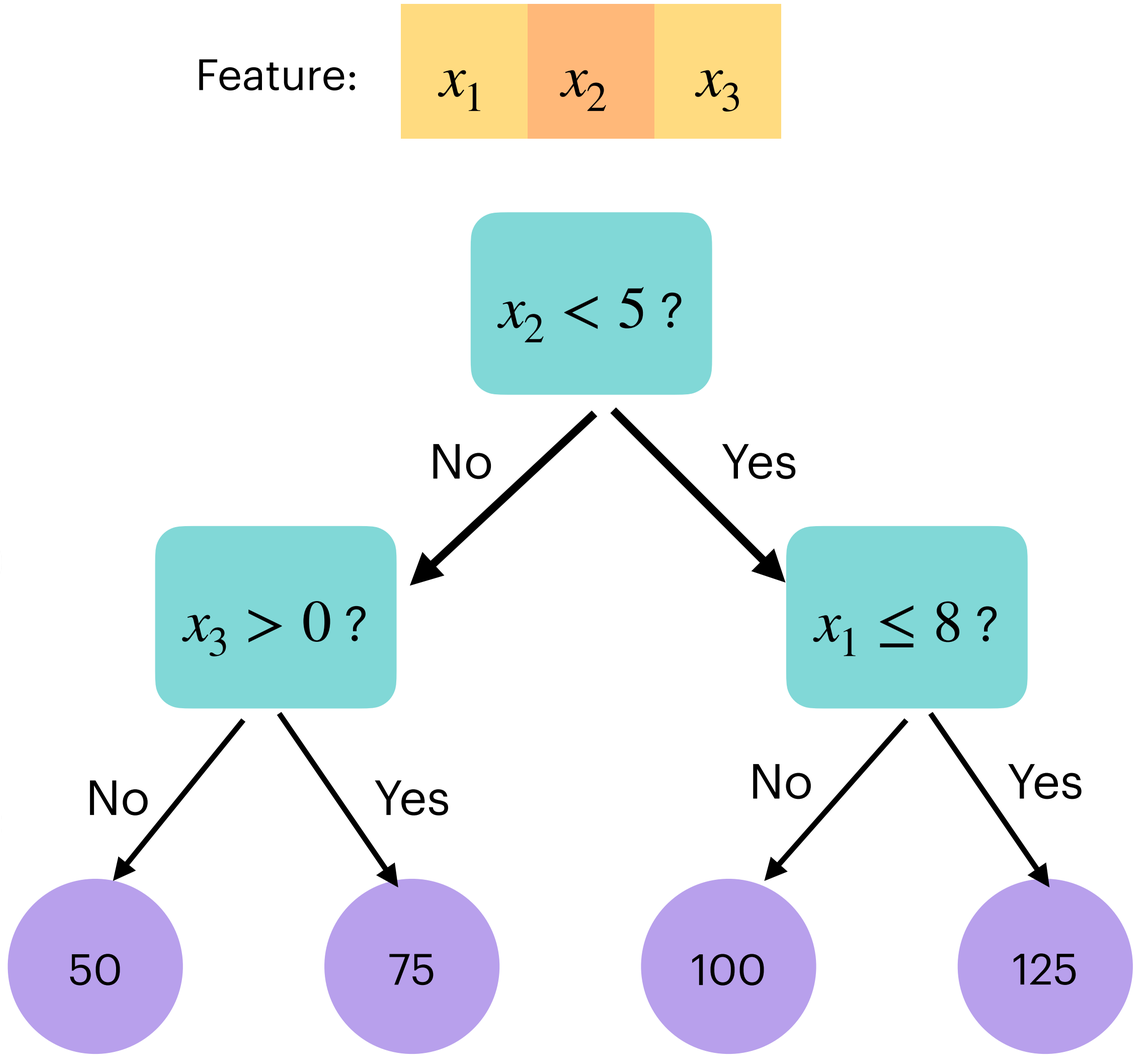
Predictive Modeling

Contents

Overview of Decision Tree Algorithm

Training:

- Builds a tree where the **nodes** are splits (decisions) and the **leafs** are predictions.
- From the root, each node attempts to find the **best attribute split**
  - **Best**: looks for single attribute split that maximizes information gain (greedy approach)
  - **Random*:** chooses random attribute split out of ones that achieve information gain
- Leafs are made by averaging over the volumes of examples left



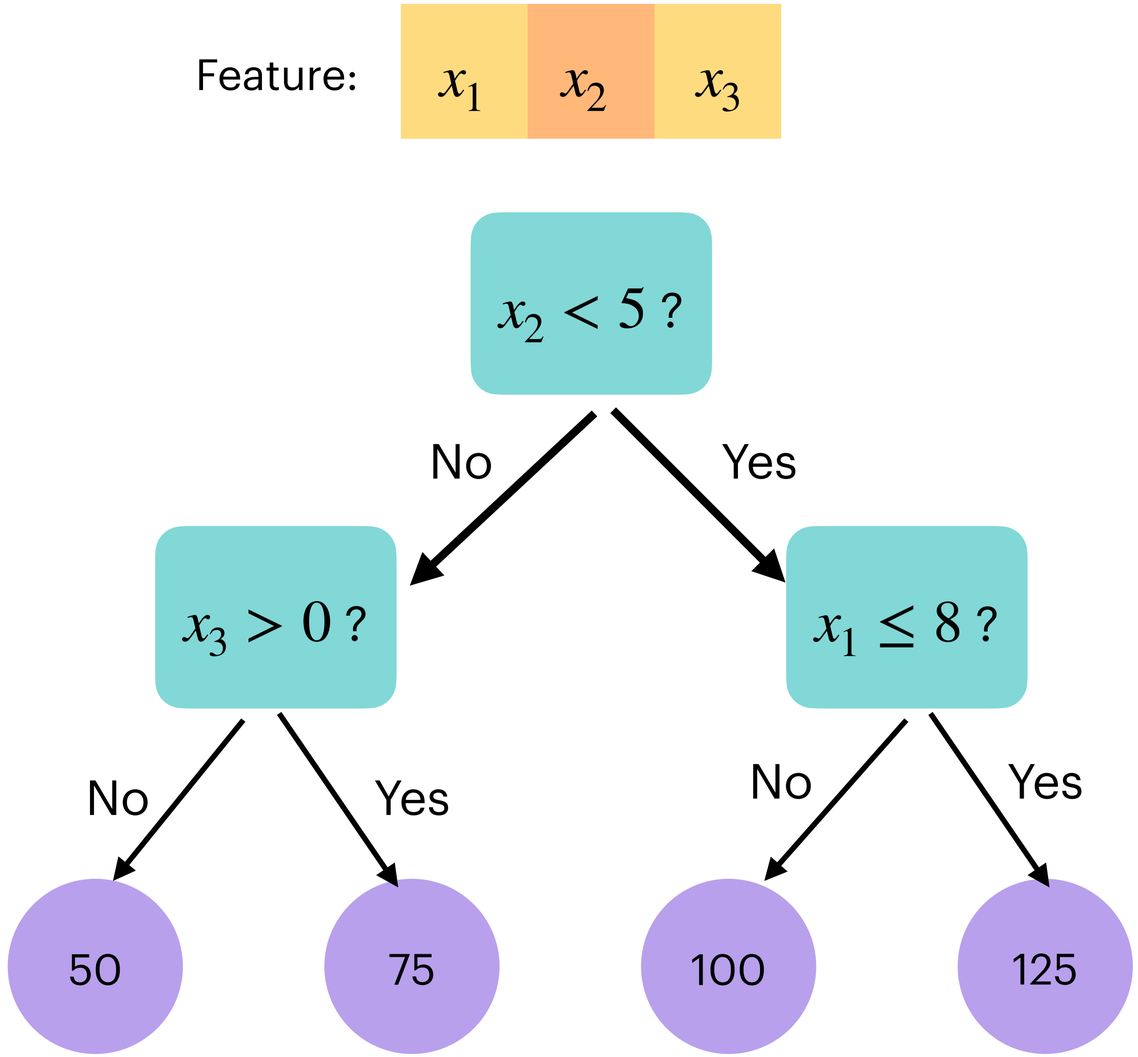
Overview of Decision Tree Algorithm

Regularization (Pre-Pruning) Methods:

1. Placing a limit on the depth of the tree
   - Decreasing **max_depth** hyperparameter
2. Restricting the amount of attributes considered for splitting
   - Decreasing **max_features** hyperparameter

Implementation:

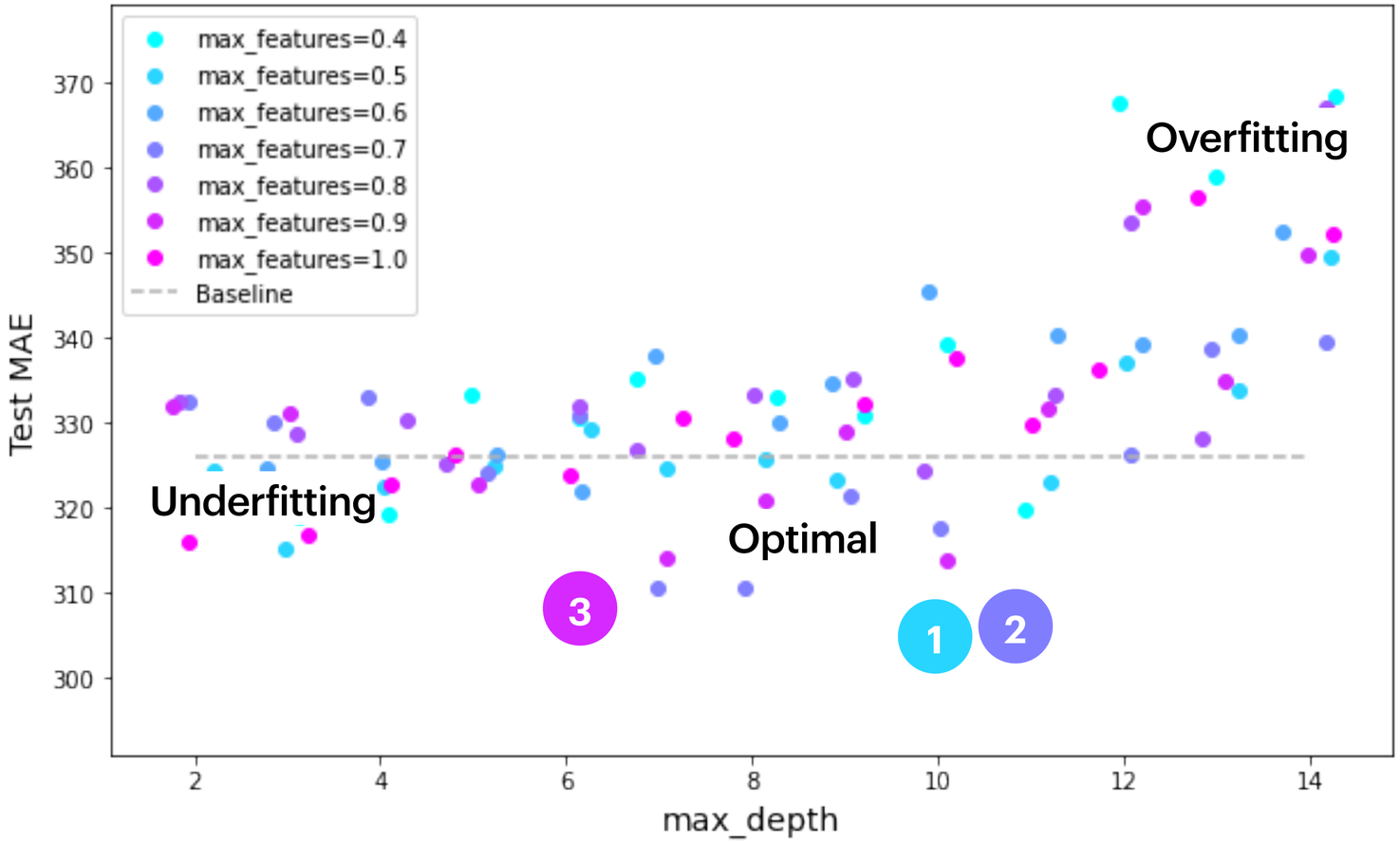
- SciKit-Learn's DecisionTreeRegressor class [8] handles multi-output regression
- Criterion: **MAE**, Splitting: **Random**



Hyperparameter Tuning & Results

Hyperparameter Search:

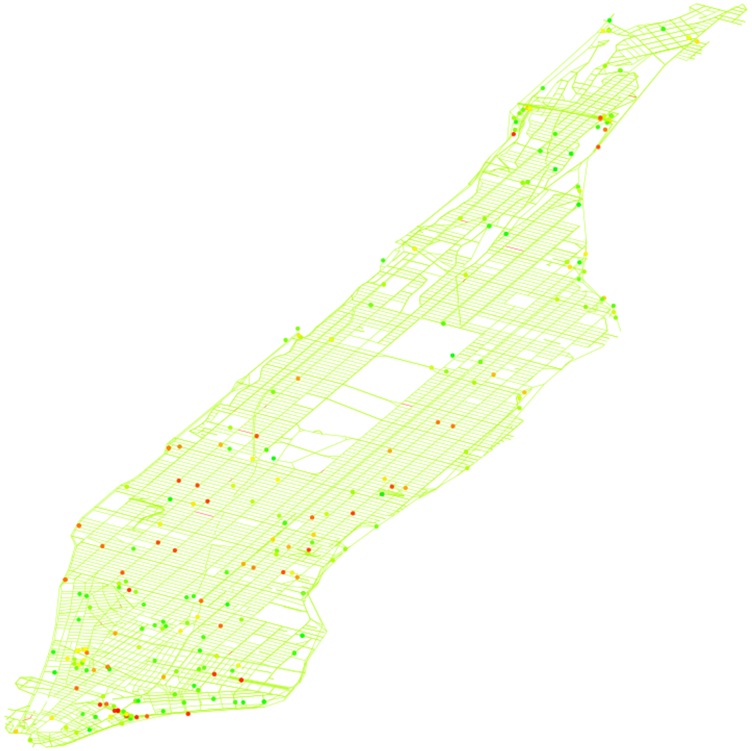
- max_depth $\in$ 2 to 15 in steps of 1
- max_features $\in \{0.4,0.5,0.6,0.7,0.8,0.9\}$



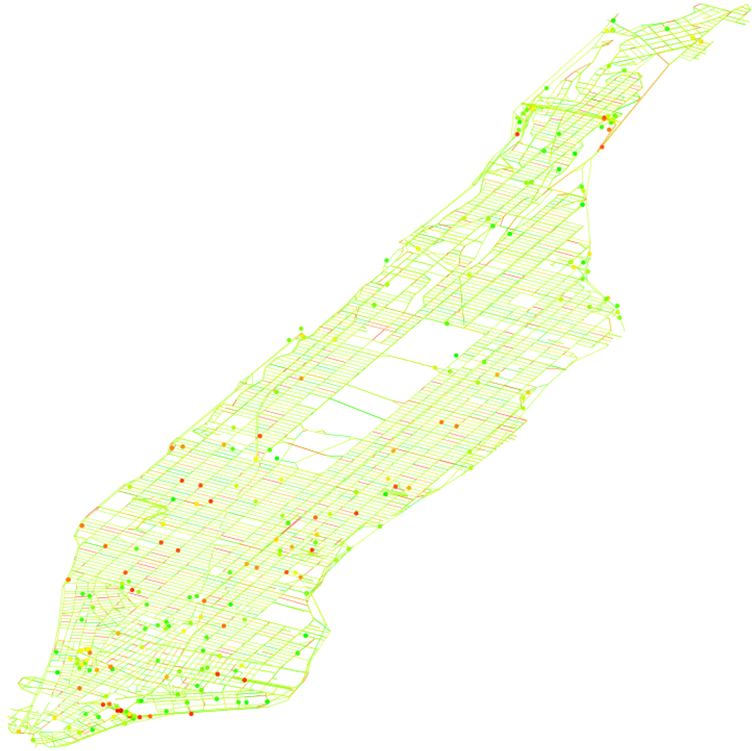
Best Results:

1. max_depth = 10, max_features = 0.5 : **305.5**
2. max_depth = 11, max_features = 0.7 : **305.8**
3. max_depth = 6, max_features = 0.9 : **307.7**

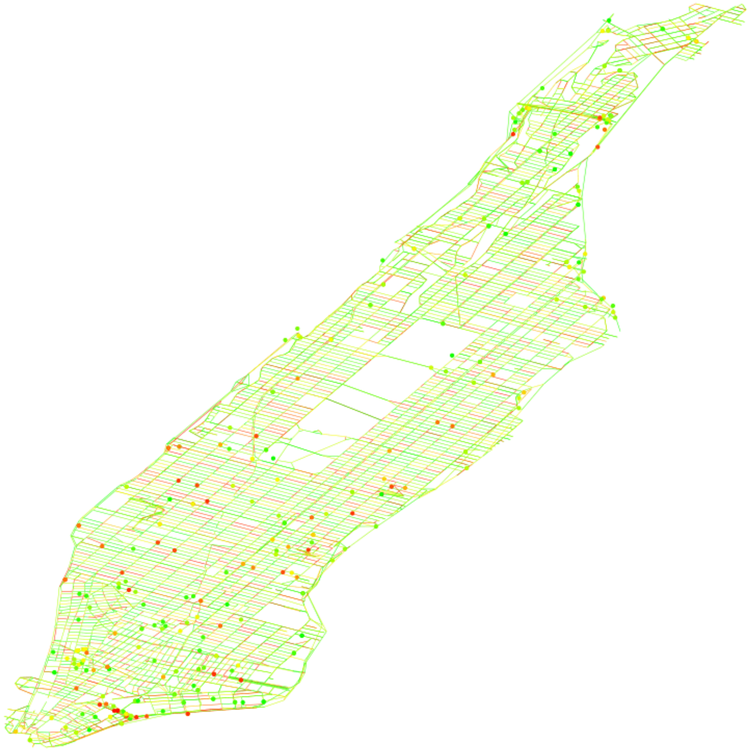
Animations of Traffic Predictions



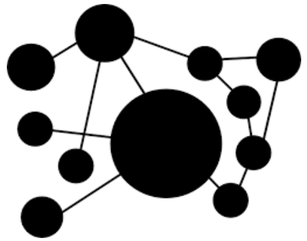
Underfitting
max_depth = 1
max_features = 1



Optimal
max_depth = 10
max_features = 0.5



Overfitting
max_depth = 20
max_features = 1



Predictive Modeling

Contents

Overview of Random Forest Algorithm

Training:

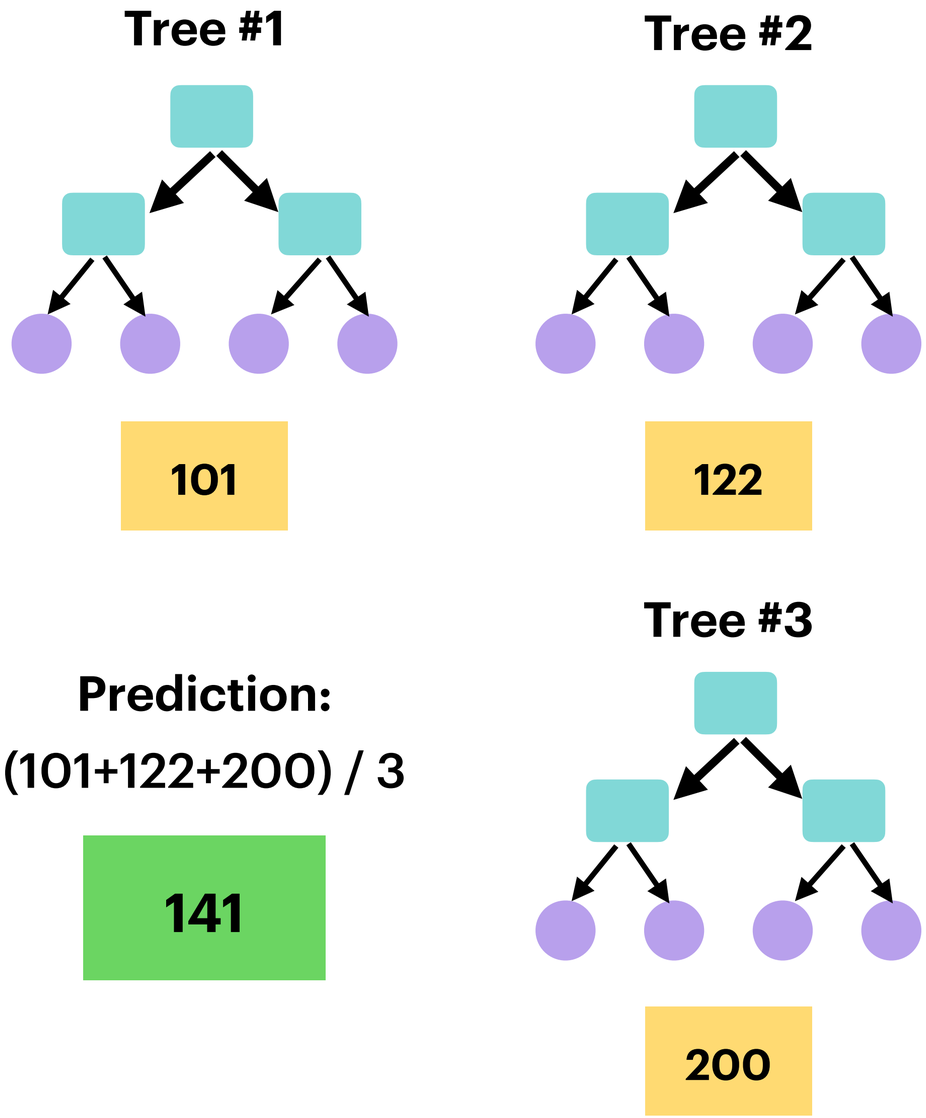
- A group of trees are built using **random** splitting
- Each tree is trained to predict the multi-output

Testing:

- The final prediction is the **mean** or **median** of each tree's individual prediction

Regularization methods:

1. Adding more trees to the forest
   - Increasing **n_estimators** hyperparameter
2. Placing a limit on the depth of each tree
   - Decreasing **max_depth** hyperparameter



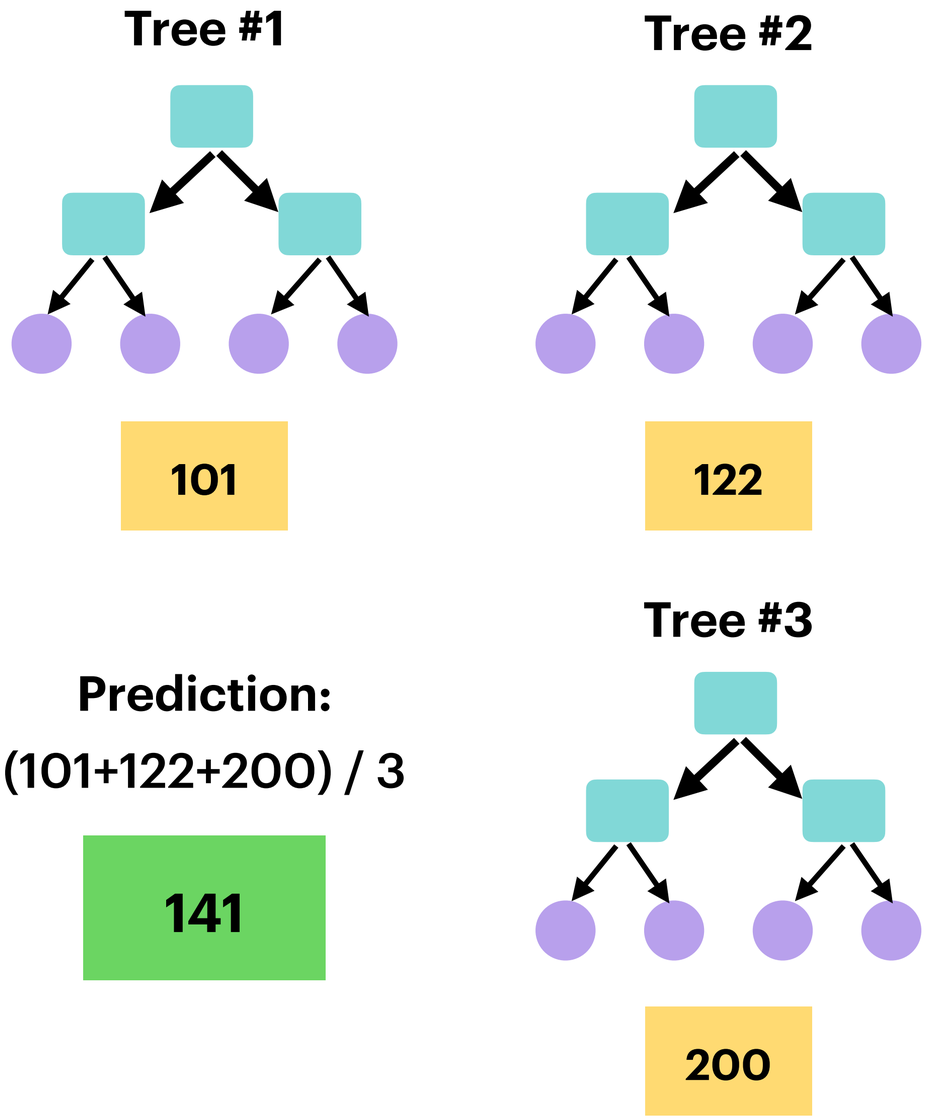
Overview of Random Forest Algorithm

Regularization methods:

1. Adding more trees to the forest
   - Increasing **n_estimators** hyperparameter
2. Placing a limit on the depth of each tree
   - Decreasing **max_depth** hyperparameter

Implementation:

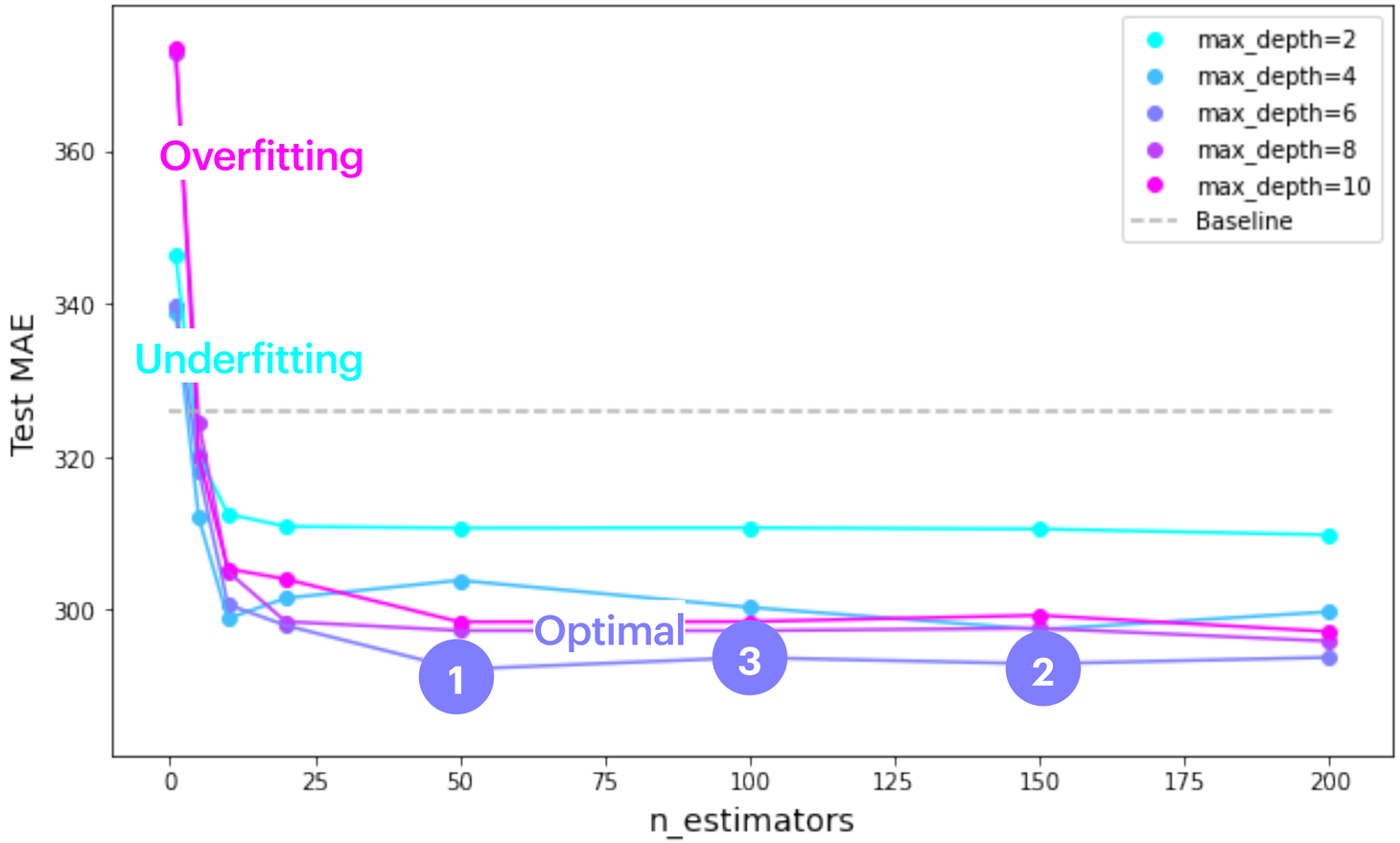
- SciKit-Learn's RandomForestRegressor class [9] handles multi-output regression
- Criterion: **MAE**



Hyperparameter Tuning & Results

Hyperparameter Search:

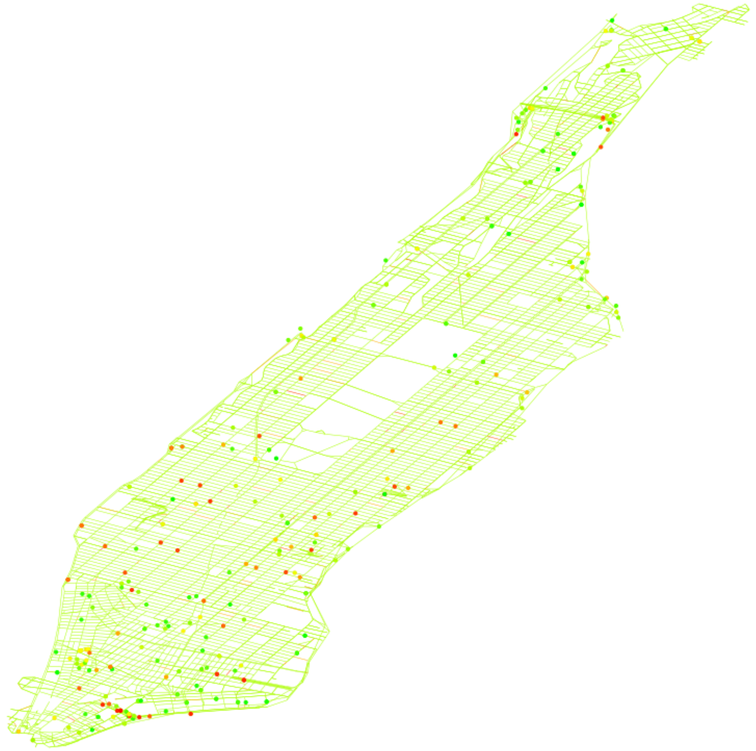
- n_estimators $\in \{2,5,10,20,50,100,150,200\}$
- max_depth $\in 2$ to 10 in steps of 2



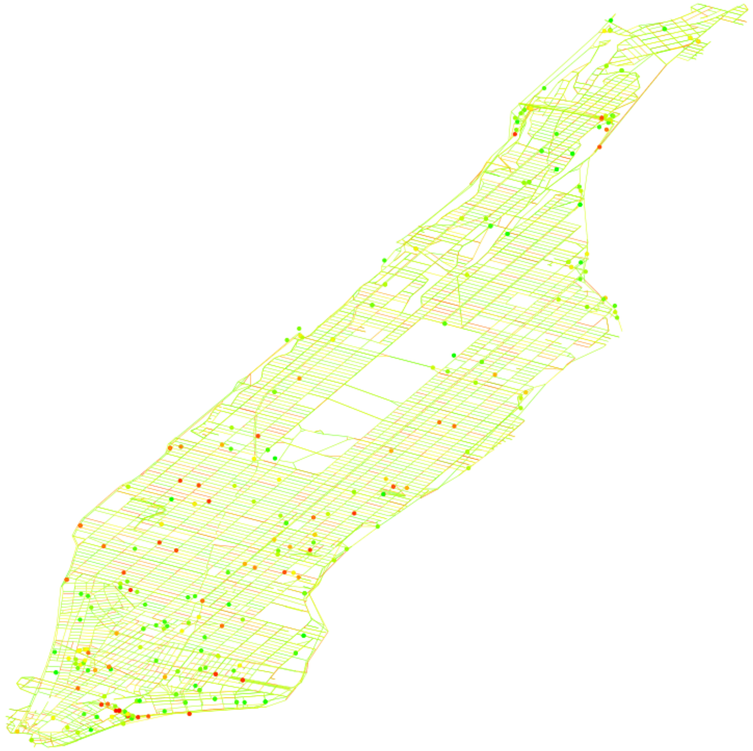
Best Results:

1. n_estimators = 50, max_depth = 6 : **292.2**
2. n_estimators = 150, max_depth = 6 : **292.9**
3. n_estimators = 100, max_depth = 6 : **293.8**

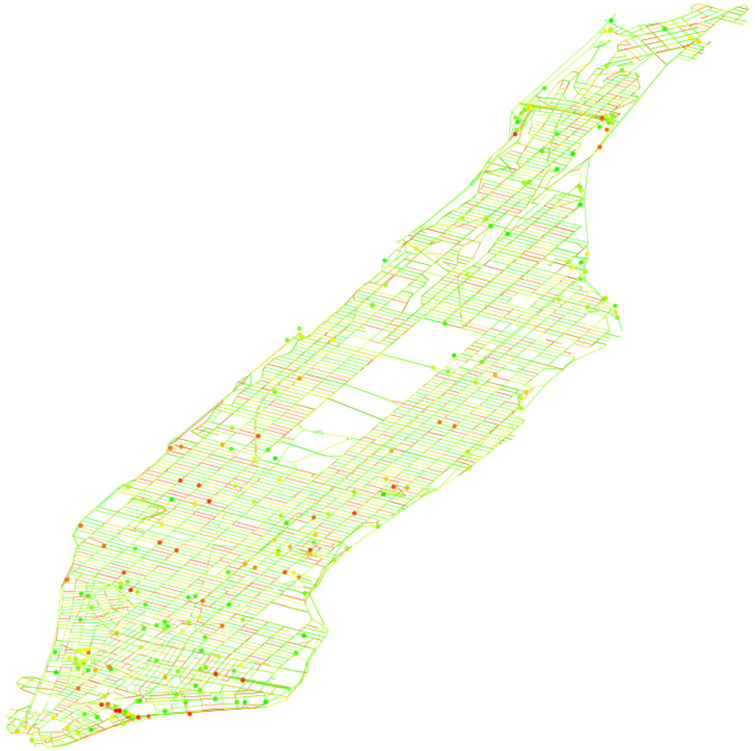
Animations of Traffic Predictions



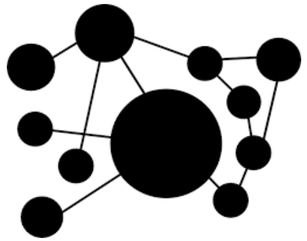
Underfitting
n_estimators = 2
max_depth = 1



Optimal
n_estimators = 50
max_depth = 6



Overfitting
n_estimators = 1
max_depth = 20



Predictive Modeling

Contents

Overview of Neural Network Algorithm

Training:

- Network parameters (weights & biases) are optimized via backpropagation

Testing:

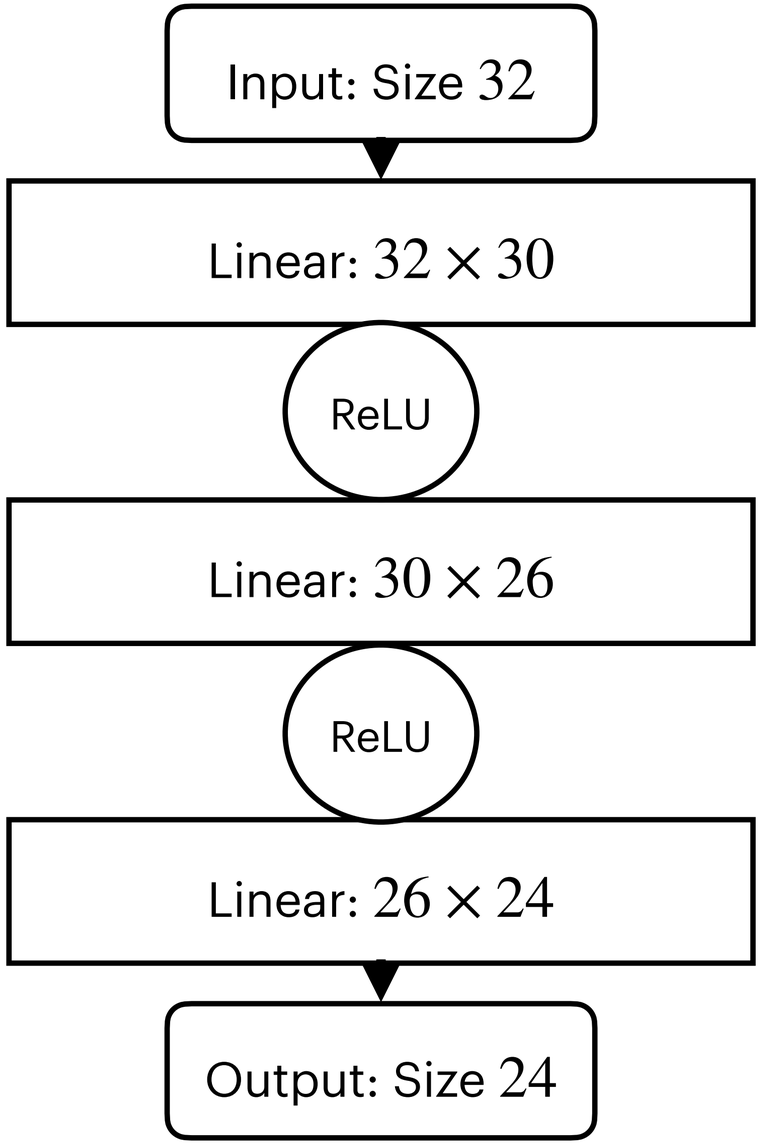
- The road segment feature is fed through the network to produce a 24-hour volume prediction

Regularization methods:

- Early stopping: stop when test error begins to rise

Hyperparameters:

- **Learning rate:** scale for weight update
- **# of hidden layers:** how deep the network is



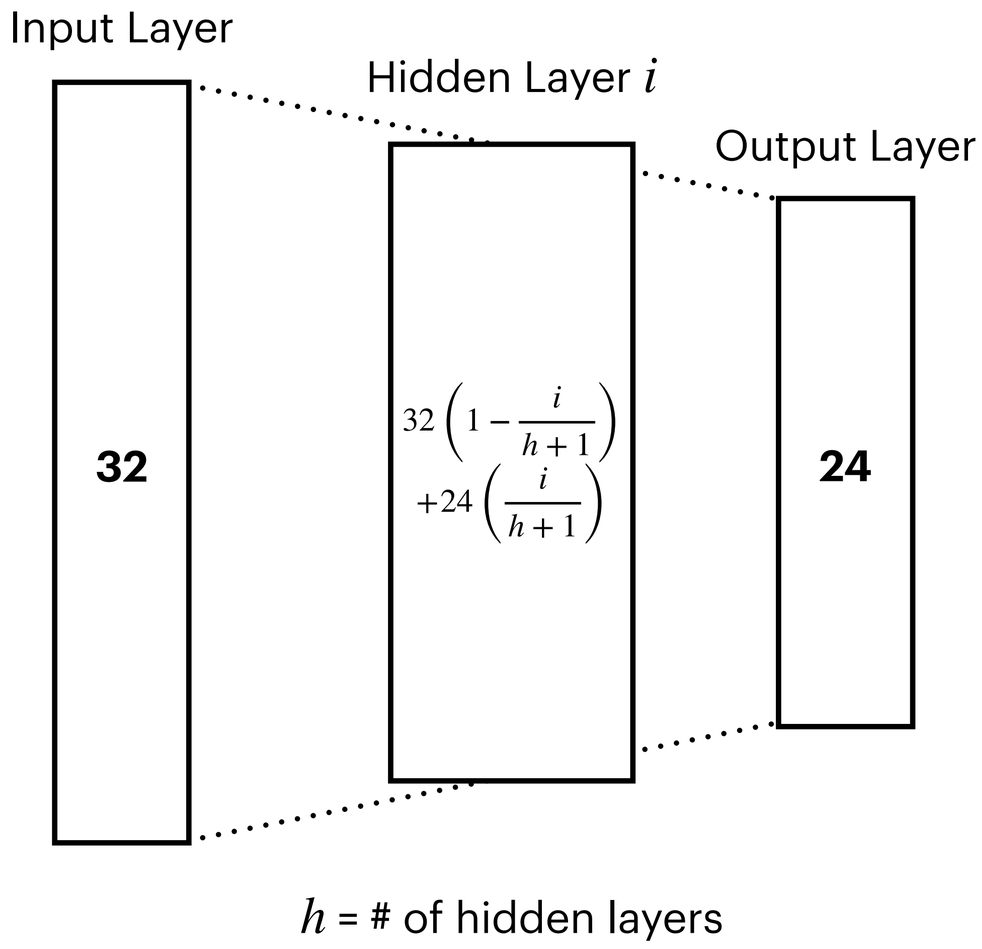
Overview of Neural Network Algorithm

Hyperparameters:

- **Learning rate:** scale for weight update
- **# of hidden layers:** how deep the network is

Implementation:

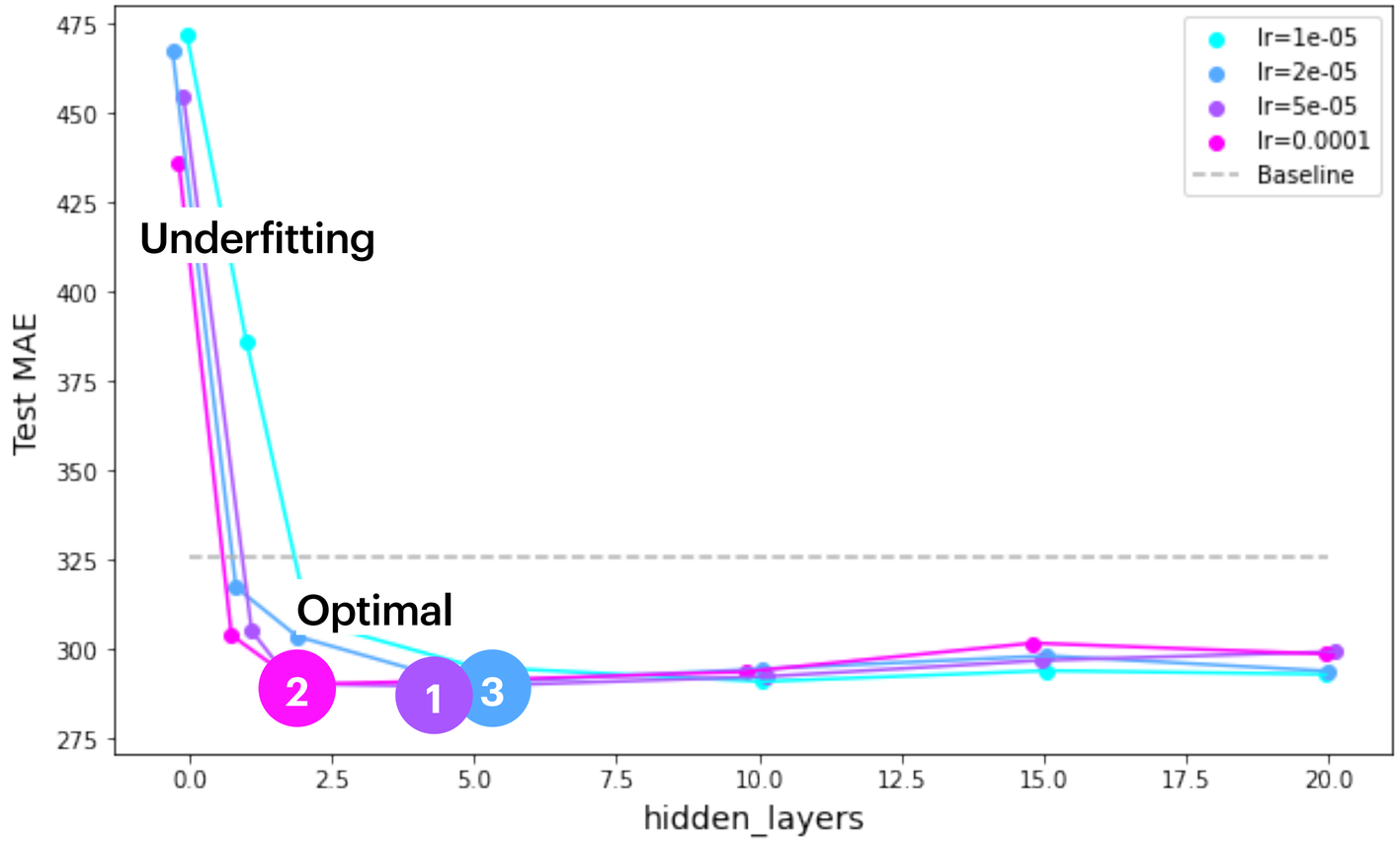
- Pytorch **nn.Module** with structure shown:
- **MAE** loss
- **Adam** optimization
- Batch size of **32**
- **Early stopping** – 5 iterations without new best test loss



Hyperparameter Tuning & Results

Hyperparameter Search:

- lr $\in \{10^{-5},\ 2\times10^{-5},\ 5\times10^{-5},\ 10^{-4}\}$
- hidden_layers $\in \{0, 1, 2, 5, 10, 15, 20\}$

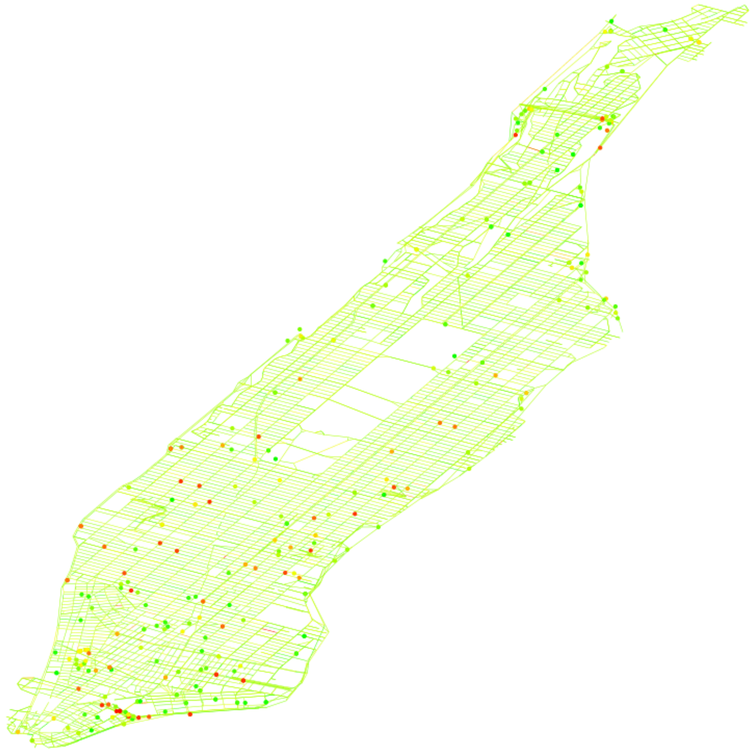


Best Results:

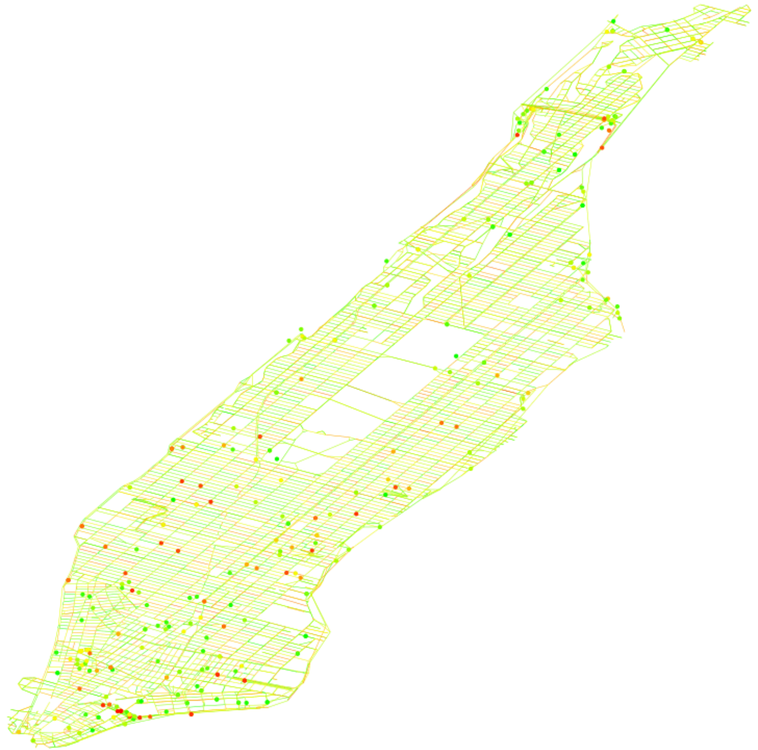
1. hidden_layers = 5, lr = 5e-5 : **289.7**
2. hidden_layers = 2, lr = 1e-4 : **290.3**
3. hidden_layers = 5, lr = 2e-5 : **290.6**

Animations of Traffic Predictions

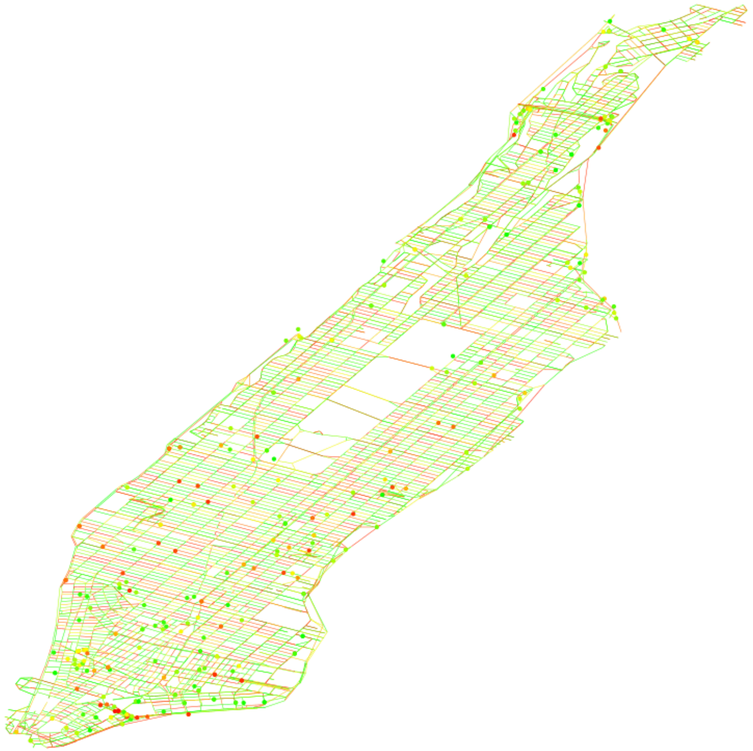
*no early stopping



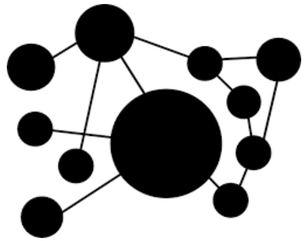
Underfitting
hidden_layers = 0
lr = 1e-5



Optimal
hidden_layers = 5
lr = 5e-5



Overfitting
hidden_layers = 10
lr = 1e-4

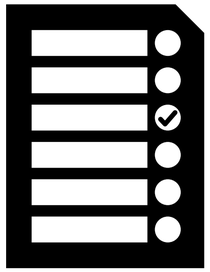


Predictive Modeling

Contents

Comparing the Best Models

	K-Nearest Neighbors	Decision Tree	Random Forest	Neural Network
Test error (mean ± std.)	309.0 ± 56.6	305.5 ± 69.0	292.2 ± 37.5	289.7 ± 48.7
Train time	0s	916ms	301ms	8.27s
Test time	295s	1.29s	1.92s	1.29s



Model Evaluation & Discussion

Contents

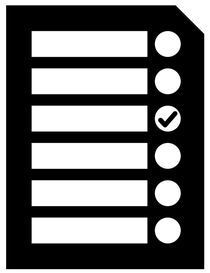
Questions We Look to Address

Model Trustworthiness: In this project, we fit models on a dataset of 310 road segments, but make predictions with these models for 9000+ unforeseen road segments. How reliable are these predictions? How do we know?

Model Performance: We received generally similar results for each algorithm. What do these results mean? Can we really quantify “performance” with one metric? How does this affect the way models learn in our project?

Project Improvements: How could we improve the outcome of the project? Should we go about the collection of data differently? With better data, should we keep the algorithms we use, or expand our consideration?

Project Merits: We covered a lot of content in this project. What are the biggest takeaways? If appropriately refined, how could the deliverables of this project serve use to the public?



Model Evaluation & Discussion

Contents

Model Trustworthiness

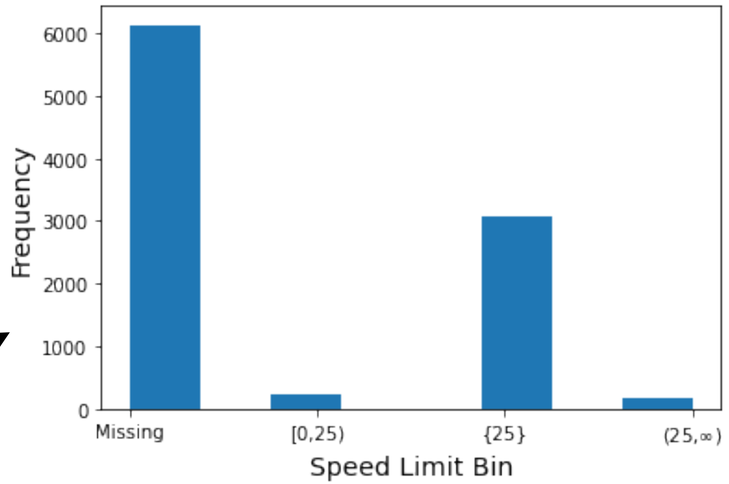
- An underlying assumption in supervised learning is that the training features and testing features are independent and **identically distributed**.
- Consider the feature vector as a 36-dimension random variable: $F := \left(X_1, X_2, X_3, \ldots, X_{36}\right)$, where X_i is a random variable of the ith dimension. If Q_F is the feature-generating distribution of road segments with stations (our training features), and P_F is the feature-generating distribution of *all* road segments in the network, then we'd like for Q_F and P_F to be as similar as possible.
- A common measurement of distribution similarity is the **KL Divergence** [X]: $D_{KL}(P\|Q) = \sum_{i=1}^{n} P[i] \times \log\left(\frac{P[i]}{Q[i]}\right)$, where P is the ground-truth distribution and Q is the estimated distribution.

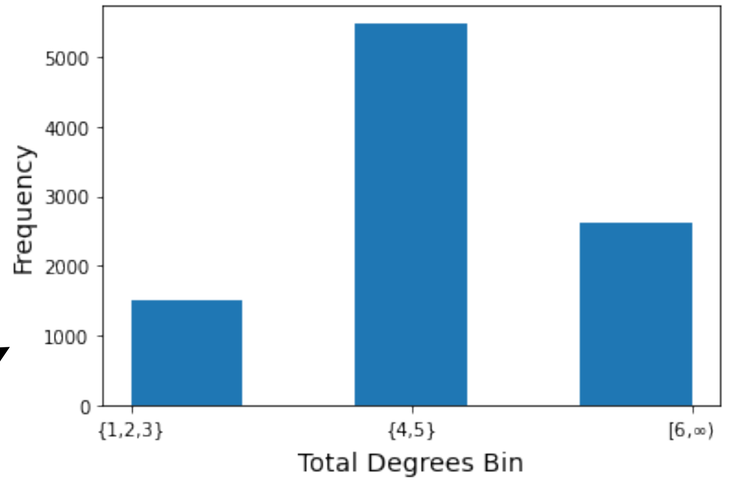
Model Trustworthiness

Partition of the feature distribution:

Index	Attribute	Bins
0	GPS Coor. Cluster	0: {0}, 1: {1}, 2: {2}, 3: {3}, 4: {4}
1	Road Length	0: (–∞, μ – σ), 1: [μ – σ, μ + σ], 2: (μ + σ, ∞)
2	Speed Limit	0: {Missing}, 1: [0,25), 2: {25}, 3: (25, ∞)
3	# of Lanes	0: {Missing}, 1: {1,2}, 2: {3,4}, 3: [5,∞)
4	Direction	0: {NE}, 1: [NW), 2: {SE}, 3: (SW)
5	Betweenness	0: (–∞, μ – σ), 1: [μ – σ, μ + σ], 2: (μ + σ, ∞)
6	Closeness	0: (–∞, μ – σ), 1: [μ – σ, μ + σ], 2: (μ + σ, ∞)
7	Total Degrees	0: {1,2,3}, 1: {4,5}, 2: [6, ∞)

Distribution Examples:





Model Trustworthiness

Partition of the feature distribution:

Index	Attribute	Bins
0	GPS Coor. Cluster	0: {0}, 1: {1}, 2: {2}, 3: {3}, 4: {4}
1	Road Length	0: (–∞, μ – σ), 1: [μ – σ, μ + σ], 2: (μ + σ, ∞)
2	Speed Limit	0: {Missing}, 1: [0,25), 2: {25}, 3: (25, ∞)
3	# of Lanes	0: {Missing}, 1: {1,2}, 2: {3,4}, 3: [5,∞)
4	Direction	0: {NE}, 1: [NW), 2: {SE}, 3: (SW)
5	Betweenness	0: (–∞, μ – σ), 1: [μ – σ, μ + σ], 2: (μ + σ, ∞)
6	Closeness	0: (–∞, μ – σ), 1: [μ – σ, μ + σ], 2: (μ + σ, ∞)
7	Total Degrees	0: {1,2,3}, 1: {4,5}, 2: [6, ∞)

Bins via cartesian product:

B_1: 0, 0, 0, 0, 0, 0, 0, 0

B_2: 0, 0, 0, 0, 0, 0, 0, 1

B_3: 0, 0, 0, 0, 0, 0, 0, 2

⋮

$B_{\sim 25k}$: 4, 2, 3, 3, 3, 2, 2, 2

Total: ~25,000

Model Trustworthiness

We will use Q_B and P_B to approximate Q_F and P_F, respectively.

Sample space of B:

- $B_1 =$ the road segment belongs to Bin 1
- $B_2 =$ the road segment belongs to Bin 2, and so on...

We can easily compute Q_B and P_B since B is discrete.

Our assumption:

$$Q_B \approx P_B \longleftrightarrow Q_F \approx P_F$$

Bins via cartesian product:

B_1: 0, 0, 0, 0, 0, 0, 0, 0

B_2: 0, 0, 0, 0, 0, 0, 0, 1

B_3: 0, 0, 0, 0, 0, 0, 0, 2

⋮

$B_{\sim 25k}$: 4, 2, 3, 3, 3, 2, 2, 2

Total: ~25,000

Model Trustworthiness

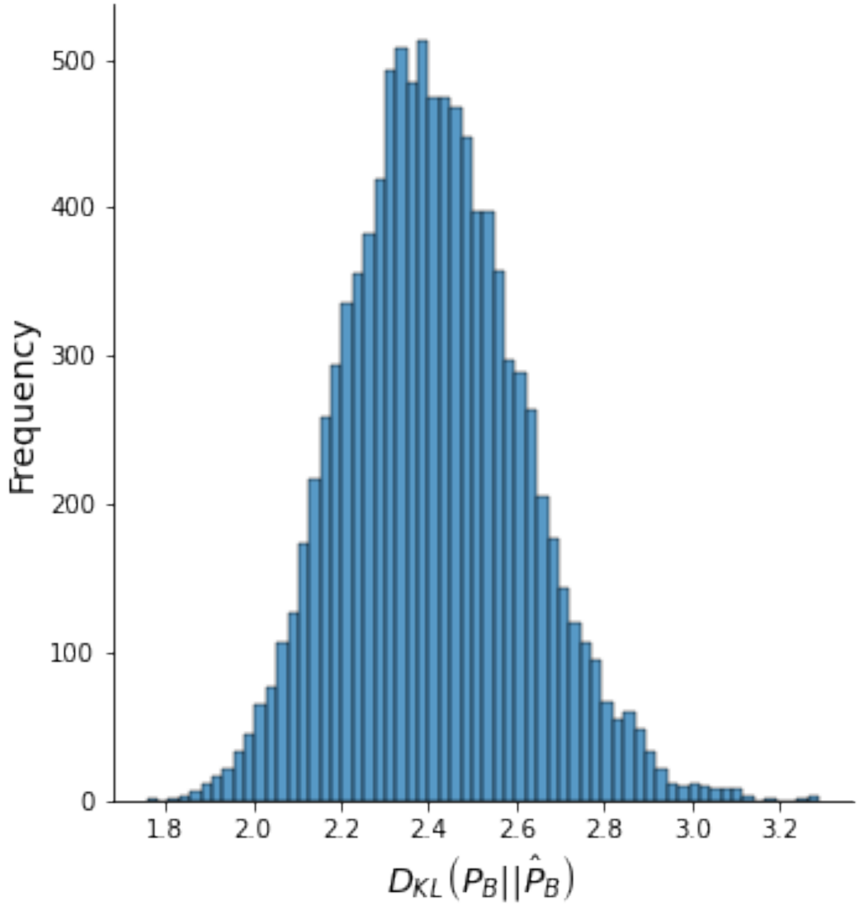
Hypothesis Test: Is there evidence to suggest that the train examples (road segments with stations) are **not** drawn from P_B, the ground-truth road segment distribution?

Null hypothesis: The train examples were sampled from P_B

Alternate hypothesis: The train examples were sampled from a different distribution.

Significance level: $\alpha = 0.05$

Approach: Draw 10,000 samples of size 310 from P_B, and for each sample, compute $D_{KL}\left(P_B \parallel \hat{P}_B\right)$, where $\hat{P}_B$ is the sample distribution over B. Plot a histogram of the results.



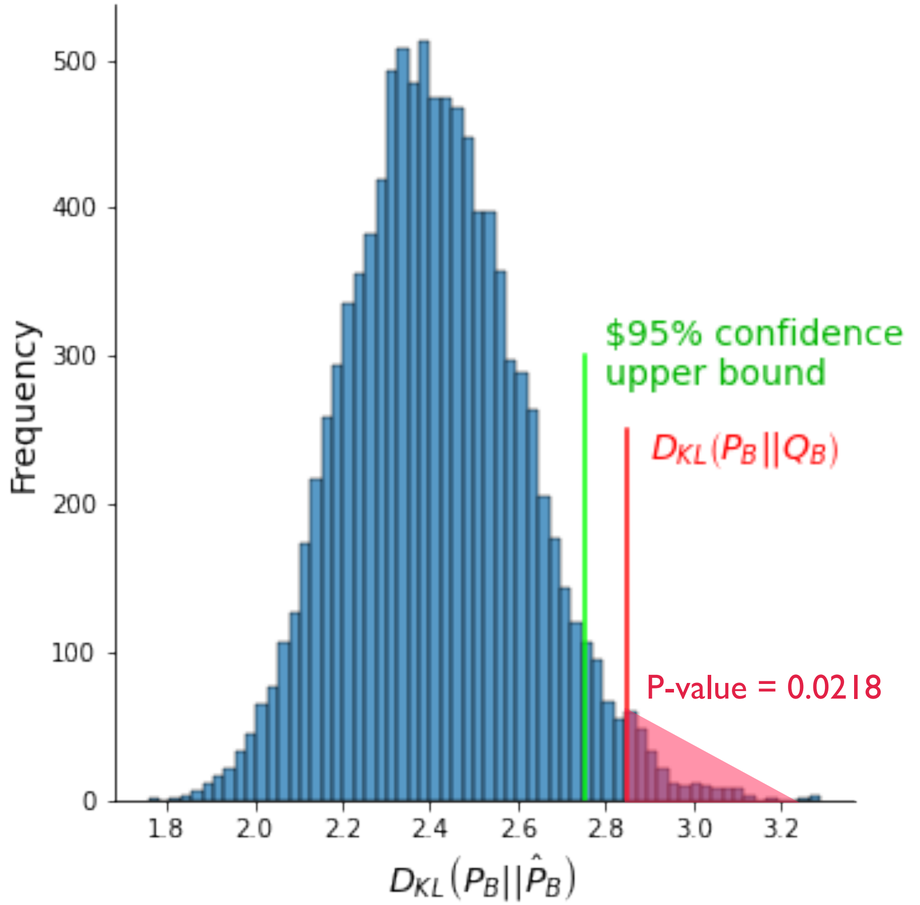
Model Trustworthiness

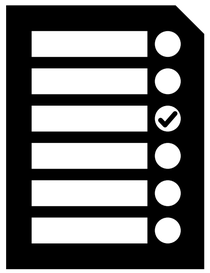
Assume that the training examples were sampled from P_B. If Q_B is the sample distribution, then with 95% $(1-\alpha)$ confidence, we would expect $D_{KL}\left(P_B \parallel Q_B\right)$ to be less than the upper bound 2.76.

Test statistic: $D_{KL}\left(P_B \parallel Q_B\right) = 2.85$

Conclusion: Since 2.85 is not less than the 95% upper bound, then we reject the null hypothesis and claim that there exists sufficient evidence to suggest that the training examples were **not drawn** from the ground-truth road segment distribution.

Bigger picture: the train & test distributions are different enough to raise concern for our model's ability to generalize.





Model Evaluation & Discussion

Contents

Model Performance

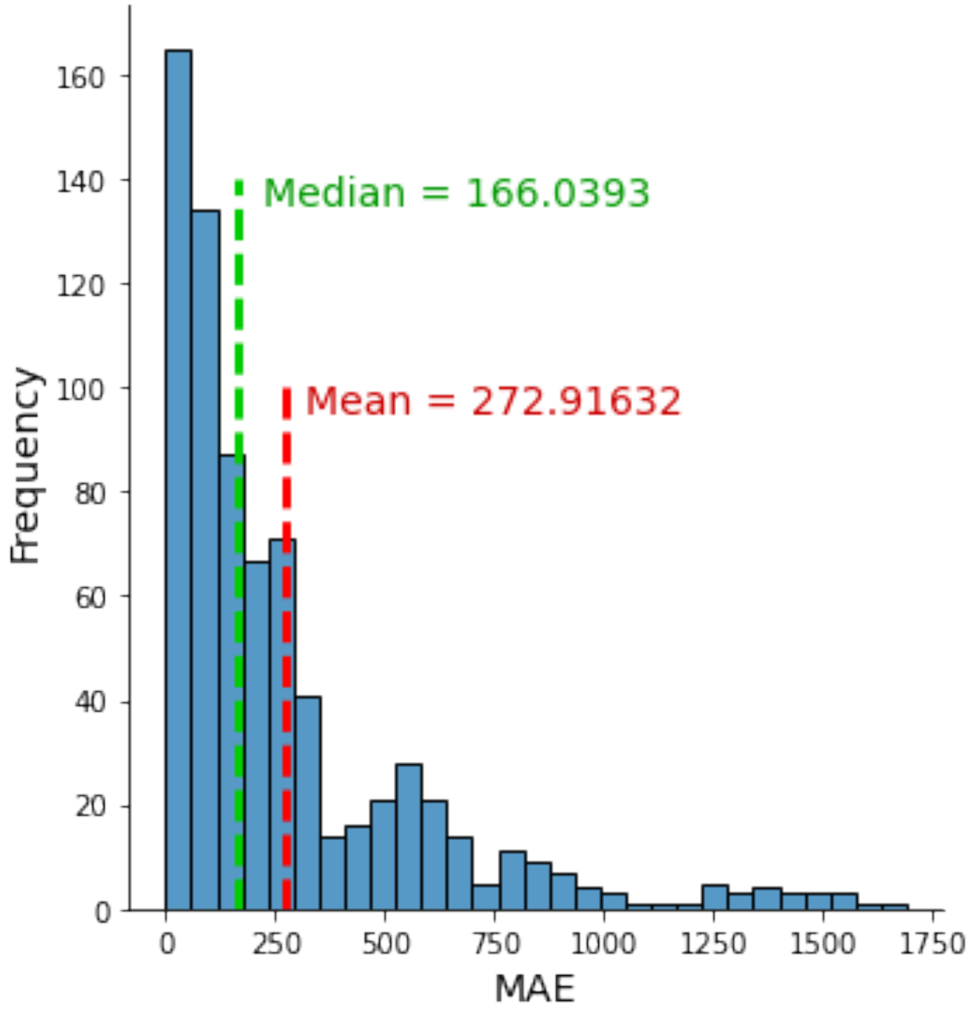
A ***Neural Network*** with 5 hidden layers and 5×10^{-5} learning rate achieves a test loss of 272.9

What does this value mean?

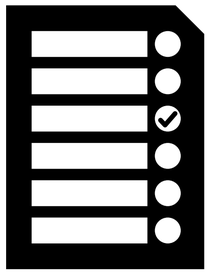
- On average, the model's ***hourly*** traffic volume prediction is off from the ground truth by 272.9
- Every ***minute***, the model's volume count is only 4.55 vehicles off.

Is *"mean MAE"* a good performance metric?

- Very sensitive to outliers
- Must keep in mind full MAE distribution
- Sometimes, a necessary evil for deep learning



	25%	50%	75%
Per hour	67.6	166.0	332.0
Per minute	1.13	2.77	5.53



Model Evaluation & Discussion

Contents

Improving Dataset Collection

My suggestions:

1. Target-side modifications

- Add more stations if possible (500? 1k?)

  ✓ Better for deep learning models

- Sample road segments from P_B

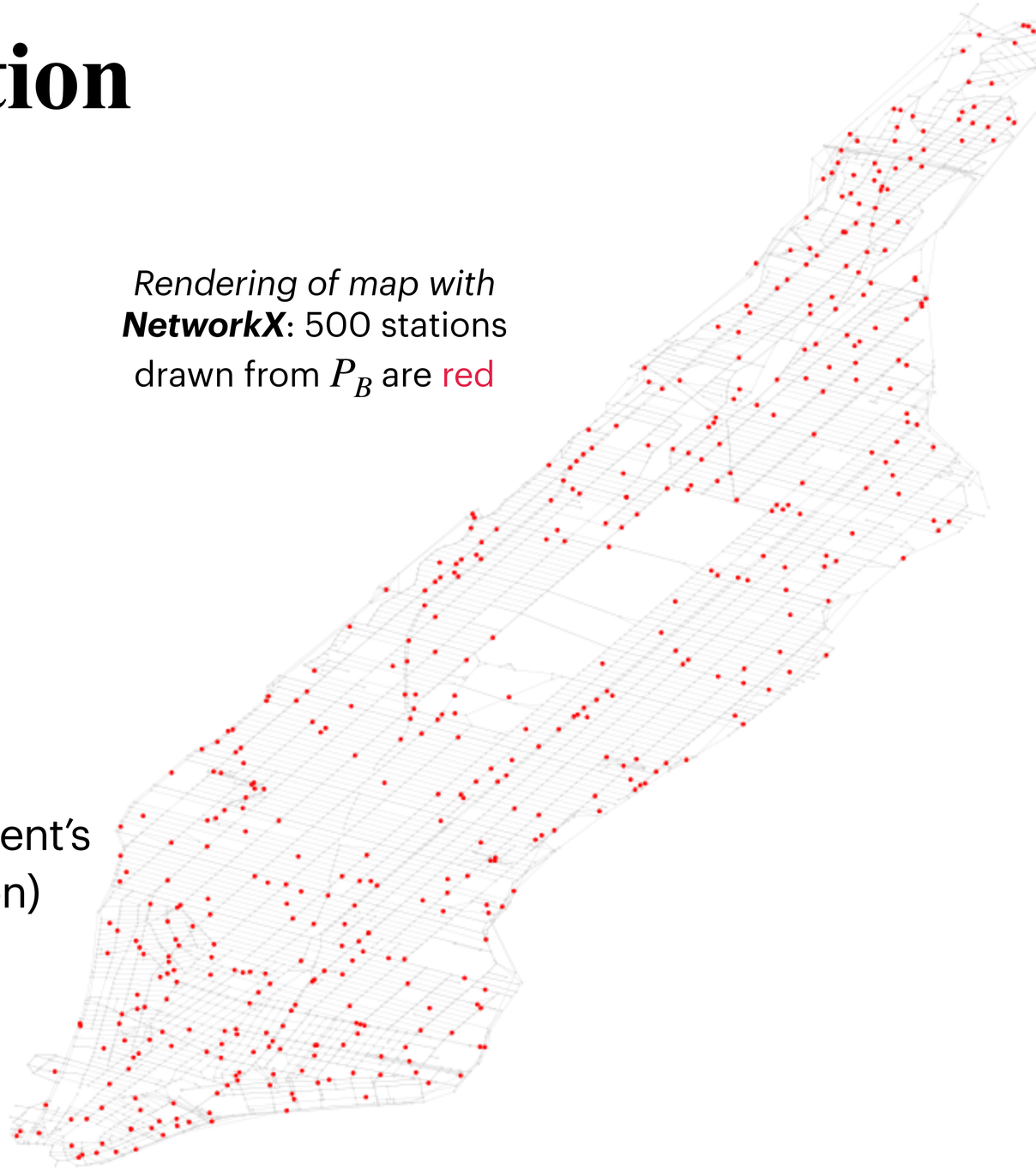
  ✓ Identical train & test distributions

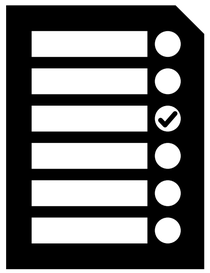
2. Feature-side modifications

- Add more information about each road segment's surroundings (e.g. residence, work, population)

  ✓ More for the model to learn on

Rendering of map with ***NetworkX***: 500 stations drawn from P_B are red





Model Evaluation & Discussion

Contents

Project Merits

- In 2017, NYSDOT purchased traffic stations for roughly $800 each [X]
  - Stations for 310 road segments would cost ~$248,000
  - Stations for **all** road segments would cost ~$7.7 million
- Counting each road segment by machine would give the best results, but is ~30x more expensive
- Thus, realistic & accurate ML traffic volume generation would **economically** serve great use to state DOTs

Arizona DOT [X]

Wisconsin DOT [X]

References

[1] https://www.dot.ny.gov/tdv

[2] https://www.openstreetmap.org/

[3] https://www.kaggle.com/code/usui113yst/basic-network-analysis-tutorial

[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[5] https://networkx.org/documentation/networkx-1.10/reference/generated/networkx.DiGraph.in_degree.html

[6] https://networkx.org/documentation/networkx-1.10/reference/generated/networkx.DiGraph.out_degree.html

[7] https://networkx.org/documentation/networkx-1.10/reference/generated/networkx.algorithms.shortest_paths.weighted.single_source_dijkstra.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html