



REAL-TIME TRAFFIC MONITORING & TRAVEL TIME PREDICTION IN NYC USING KAFKA, SPARK, AND ML

Course Title: Enterprise Cloud Computing and Big Data (BUDT 737)

**TITLE OF CASE STUDY: REAL-TIME TRAFFIC
MONITORING & TRAVEL TIME PREDICTION IN
NYC USING KAFKA, SPARK, AND ML**

Team Members:

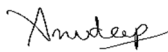


Anudeep Koneru

Piyali Bedagkar

Avi Chaudhary

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

Name	Signature
Anudeep Koneru	
Piyali Bedagkar	
Avi Chaudhary	

I. CASE STUDY

Introduction

New York City's traffic congestion has long been a challenge for city administrators, especially in Manhattan where vehicle density is among the highest in the world. Despite partial advancements in traffic control systems, such as adaptive signals and camera-based traffic counts, the city still struggles with delayed reactions to real-time congestion. These delays increase commute times, reduce fuel efficiency, and frustrate daily commuters.

In response, several pilot programs and research projects have emerged to explore the potential of data-driven traffic systems. This case study presents one such initiative, focused on simulating real-time traffic flow using a combination of Apache Kafka, Spark-based processing, and machine learning models. The goal is to estimate travel time dynamically and visualize road-level congestion patterns, all within a cloud-based system that mimics the scale and complexity of Manhattan's urban layout.

By simulating thousands of real-time vehicle routes across Manhattan and applying classification and prediction models, this project demonstrates how predictive analytics can serve as a critical tool for urban transportation planning.

NYC's Traffic Landscape and the Roadblocks Ahead

New York City continues to rank among the top cities for traffic delays globally. According to the INRIX 2024 Global Traffic Scorecard, drivers in NYC lose an average of 117 hours annually due to congestion. Manhattan, in particular, with its dense urban layout and grid system, experiences intense pressure on its transportation infrastructure. From public buses and taxis to delivery vans and rideshares, the volume of traffic makes the job of route optimization and signal coordination exceedingly complex.

The NYC DOT has invested in partial upgrades, such as semi-adaptive traffic signals and mobile sensing units. However, these are not sufficient for forecasting traffic issues or planning ahead. Real-time responsiveness remains lacking. The inability to dynamically classify traffic congestion or estimate time-to-destination at the route level limits both emergency planning and daily traffic operations.

Emergence of New-Age Urban Data Infrastructure

Cities like New York have entered an era of data abundance. With the rapid deployment of connected infrastructure—ranging from GPS-tagged delivery fleets to mobile traffic monitoring units—the availability of streaming vehicular data is no longer limited to experimental deployments. Combined with cloud platforms, this evolution sets the stage for scalable real-time decision support systems in city transportation networks.

These technologies also reduce dependency on traditional traffic sensors and static survey data, which are costly and lack responsiveness. Instead, data can now be harvested dynamically and continuously from real-world vehicle behavior, offering transportation planners a richer and more timely source of intelligence.

Aligning Innovation with Urban Planning Needs

Urban mobility decisions are no longer solely the domain of infrastructure engineers. Today, they require collaboration between data scientists, system architects, and public agencies to translate insights into operational improvements. This convergence of disciplines marks a new phase of digital transformation in the public sector, especially in transportation.

City planners are increasingly expected to not just manage road infrastructure but also interpret real-time data, assess predictive trends, and adopt technology-driven workflows. In this environment, the ability to move from reactive to proactive planning is critical. Cities like New York, with their dynamic infrastructure and complex road networks, need agile solutions that can adapt to changing traffic conditions instantly.

This project mirrors that shift by combining predictive modeling with geospatial mapping and real-time alerting. It aims to showcase how real-time insights can support faster, more informed decisions in traffic control, ultimately benefiting both policy makers and daily commuters. By building a system that learns from traffic behavior and visualizes trends dynamically, the project bridges the gap between technical innovation and municipal planning. that learns from traffic behavior and visualizes trends dynamically, the project aims to bridge the gap between technical innovation and municipal planning.

The Challenge Ahead

Although New York City has made investments in upgrading infrastructure, the systems in place still lack the capabilities to support real-time traffic decisions. Most traffic signal plans rely on scheduled timing or at best, semi-adaptive logic, which falls short during irregular traffic surges, emergency scenarios, or special events. Without high-frequency data analysis, there's little capacity to dynamically assess where and when congestion builds up.

The absence of predictive systems means that transportation managers cannot foresee travel delays before they happen. Additionally, traffic dashboards used by agencies are typically based on historical averages or partial real-time feeds, making it difficult to react swiftly. The ability to predict route-level travel time using live vehicle movement data remains largely untapped.

By integrating real-time data streaming with machine learning models that classify congestion severity and predict time to destination, this prototype addresses a critical gap in operational traffic management. It aims to explore how emerging technologies can support a more proactive, city-scale traffic control system.

Rachel believes that if her team could stream data in real time, classify road segments dynamically, and predict route delays based on historical patterns and live feeds, it would mark a turning point in how traffic is managed across Manhattan.

Case Questions

1. Can real-time classification of street segments help identify congestion zones as they emerge?
2. Is it possible to forecast end-to-end travel time for vehicles using a lightweight ML model trained on segment behaviour?
3. How effective is a Kafka-based streaming pipeline in simulating real-world data generation and ingestion at scale?
4. Would a live dashboard reflecting speed, segment classification, and prediction offer meaningful operational insights for city planning teams?

II. SOLUTION FOR CASE STUDY

1. Overview of the Solution Architecture

Our system is designed as an end-to-end streaming and prediction pipeline that simulates real-time traffic data, processes it using a cloud-based Kafka and Python stack, performs classification and machine learning-based travel time predictions, and presents results in an interactive web dashboard.

The architecture consists of the following core components:

- **Traffic Simulation & Data Generation:** Generates realistic, multi-segment vehicle routes in Manhattan.
- **Kafka Streaming (Producer & Consumer):** Streams route-level vehicle data continuously.
- **Data Aggregation & Storage:** Collects messages in batches and stores them in CSV format.
- **Prediction Engine:** Applies machine learning to estimate route travel time and classify segment severity.
- **Interactive Dashboard:** Displays real-time visualizations of congestion, vehicle density, and predicted travel times.

This modular architecture allows each layer to function independently, with flexibility to scale or integrate new data sources like live GPS or IoT signals.

2. Data Simulation and Kafka Streaming

The data pipeline begins with *generate_training_data.py*, which simulates 5000 real vehicle trips in Manhattan. Each trip includes route segment details, estimated travel time, and traffic classification using OpenStreetMap and OSMNX.

For the live system, *kafka_producer_nyc.py* dynamically simulates vehicle movements by generating:

- Start and end geolocations
- List of segments traversed
- Average speed per vehicle
- Timestamp

This data is streamed to a **Confluent Kafka topic** (*nyc_traffic*) at a high frequency (1000 vehicles/second). Each record represents a full route and is structured as a JSON message.

Example JSON message structure:

```
{
  "vehicle_id": 152,
  "start_lat": 40.758,
  "start_lon": -73.985,
  "end_lat": 40.749,
  "end_lon": -73.987,
  "segment_ids": [1023, 1044, 1067],
  "speed_kmph": 45.3,
  "timestamp": "2024-11-05 14:42:01"
}
```

This stream mimics real-world traffic behavior and forms the foundation for all downstream processing.

3. Kafka Consumer and Storage

The consumer side is implemented using *kafka_consumer_nyc.py*. It subscribes to the *nyc_traffic* Kafka topic and continuously listens for new route data. Every 5000 messages,

the consumer writes the incoming batch to `nyc_traffic_data.csv`, effectively maintaining a refreshed view of traffic flow in real time.

Key steps:

- Consumes and decodes each JSON message.
- Joins `segment_ids` (a list) into a flat string for CSV compatibility.
- Writes to `nyc_traffic_data.csv` every 5000 records, replacing the previous batch.

This batch approach is lightweight, ideal for prototyping, and keeps the system extensible for later migration to a distributed data lake or streaming database.

```
nyc_traffic_data.csv > data
1 vehicle_id,start_lat,start_lon,end_lat,end_lon,segment_ids,speed_kmph,timestamp
2 44400,40.8069824,-73.9534985,40.787692,-73.941478,"1312,1311,2318,2316,2312,9519,2311,5907,6145,8542,2500,1882,2498,8725,8656,8711,2
3 44401,40.7168982,-73.986194,40.8571682,-73.9274354,"2099,2098,420,421,424,9499,426,428,429,8433,6686,3767,6833,6613,5424,2757,6835,6
4 44402,40.7181153,-73.9865259,40.732353,-73.9849374,"7507,1339,3674,2083,8436,6461,9666,6679,3759,6823,6620,5418,2750,2720,6226,1719,
5 44403,40.8025132,-73.9529374,40.8019596,-73.934173,"8563,8290,8514,8699,3707,6628,6552,6630,1308,8468,1225,5293,5800,5497,5801,4083,
6 44404,40.7654396,-73.9838209,40.7590024,-73.970533,"3143,1965,777,775,773,4449,1560,6431,5739,7897,6966",45.39,2025-05-13 02:24:20
7 44405,40.7258067,-74.01093,40.8057431,-73.950577,"4674,4818,4816,3321,3322,3324,3326,3328,3330,3331,3334,3335,3338,3340,3342,3343,33
8 44406,40.7605391,-73.9583585,40.7471011,-73.9834426,"587,584,232,6127,6790,2305,4609,4605,4602,3548,3546,3545,5070,4581,5068,4142,50
9 44407,40.856593,-73.932761,40.7210194,-73.9821197,"1096,1093,1090,1087,1084,1081,1077,1074,1071,5331,2457,699,696,695,4980,7863,7867
10 44408,40.8075512,-73.9370575,40.7267539,-74.00896,"7747,7759,7762,7764,7765,8022,8024,8025,8013,8017,8015,8019,8002,8004,8007,8009,8
11 44409,40.7225657,-74.0014195,40.7781744,-73.9745417,"2127,2125,2124,2120,2118,1456,1457,1460,1461,1463,9183,8880,8999,8998,9290,1466
12 44410,40.8302699,-73.9439142,40.8018838,-73.9533927,"4937,4932,8612,8970,2383,1991,2379,2377,8968,2374,2369,2364,2363,2360,2357,2356
13 44411,40.7460356,-73.9747018,40.7169812,-73.9944635,"9812,9317,6327,5048,141,2916,9171,5046,5043,5042,5040,3568,5038,5036,1380,5034,
14 44412,40.7509069,-73.9806768,40.7136173,-74.011617,"5643,5641,2620,5309,6590,2902,3833,5895,5283,6470,3849,506,4071,2869,1370,9462,6
15 44413,40.828925,-73.950395,40.7787449,-73.9778503,"4215,997,995,993,991,988,986,984,982,980,978,976,974,7549,971,9548,969,966,9556,9
```

4. Feature Engineering + Segment Classification

Using `predict_live_travel_time.py`, we preprocess incoming vehicle data to calculate congestion intensity at the segment level. Here's how:

- The `segment_ids` field (a list) is exploded so each row corresponds to a single segment per vehicle.
- Speeds are aggregated across vehicles per segment ID.
- A color is assigned to each segment:
 - **Red:** speed < 30 kmph
 - **Yellow:** 30–60 kmph
 - **Green:** > 60 kmph

- Each vehicle is then re-aggregated to count how many red/yellow/green segments it passed.

These counts (red_count, yellow_count, green_count) plus total_segments are used as inputs for the machine learning model in the next stage.

```
elif view_mode == "🗺️ Segment Heatmap":
    try:
        segments_gdf = gpd.read_file(SEGMENT_FILE)
        segments_gdf["segment_id"] = segments_gdf["segment_id"].astype(int)

        if "segment_ids" not in df.columns:
            st.warning("No 'segment_ids' in traffic data. Overlay won't display.")
        else:
            avg_speed_df = df[["segment_ids", "speed_kmph"]].copy()
            avg_speed_df["segment_ids"] = avg_speed_df["segment_ids"].apply(lambda x: eval(x) if isinstance(x, str) else x)
            exploded = avg_speed_df.explode("segment_ids")
            exploded.rename(columns={"segment_ids": "segment_id"}, inplace=True)
            exploded["segment_id"] = exploded["segment_id"].astype(int)

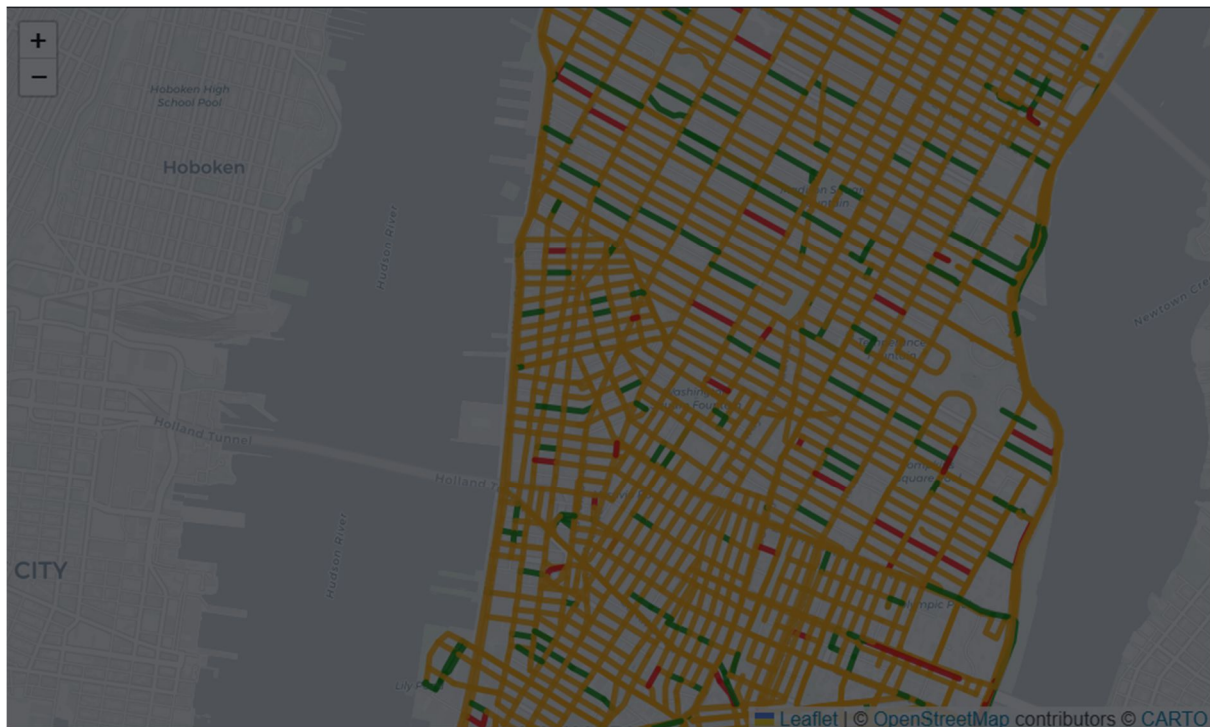
            segment_avg = exploded.groupby("segment_id")["speed_kmph"].mean().reset_index()
            merged = segments_gdf.merge(segment_avg, on="segment_id")

            def get_color(speed):
                if speed < 30: return "red"
                elif speed < 60: return "orange"
                else: return "green"

            merged["color"] = merged["speed_kmph"].apply(get_color)

            m = folium.Map(location=[CENTER_LAT, CENTER_LON], zoom_start=13, tiles="cartodbpositron")
            for _, row in merged.iterrows():
                if row["geometry"] and hasattr(row["geometry"], "coords"):
                    coords = list(row["geometry"].coords)
                    folium.Polyline(
                        locations=[(y, x) for x, y in coords],
                        color=row["color"],
                        weight=5,
                        opacity=0.85,
                        tooltip=f"Speed: {row['speed_kmph']:.1f} kmph"
                    ).add_to(m)

            st_folium(m, width=1000, height=600)
    except Exception as e:
        st.error(f"🚫 Could not load segment-level overlay: {e}")
```



5. Model Training (ML Pipeline)

The machine learning model is trained using *train_model.py*, which consumes a synthetic dataset generated by *generate_training_data.py*. The dataset contains thousands of sample trips with labeled travel times and segment color counts. Each trip record includes:

- *red_count, yellow_count, green_count*
- *total_segments*
- *true_travel_time_sec*

These features are used to train a **Linear Regression model** that predicts *true_travel_time_sec*.

The ML pipeline consists of:

- **StandardScaler**: Normalizes feature scales
- **LinearRegression**: Models the relationship between segment severity and travel time

The model is trained and saved as ***travel_model.joblib***, which is later loaded for prediction in the real-time pipeline.

```
training_data.csv > data
1  start_lat,start_lon,end_lat,end_lon,red_count,yellow_count,green_count,total_segments,true_travel_time_sec
2  40.8068316,-73.9610565,40.7755591,-73.9503323,25,23,16,64,1124.3771354761082
3  40.8048633,-73.9663136,40.7748055,-73.9922896,64,86,22,172,1733.2927024263545
4  40.800334,-73.97146,40.7748055,-73.9922896,16,25,9,50,993.0275893645367
5  40.7076904,-74.0155198,40.7262337,-73.9983091,22,17,11,50,586.621230753957
6  40.8124025,-73.9639023,40.7870011,-73.9755093,19,26,1,46,592.9201019293154
7  40.7609399,-73.9670376,40.7580419,-73.960182,0,4,2,6,72.33240496075771
8  40.7734893,-73.9890814,40.8134085,-73.9550501,32,34,34,100,1115.4691978444978
9  40.8146691,-73.9442672,40.732218,-73.998616,55,73,30,158,1662.0637006543564
10 40.7019726,-74.0100288,40.7717042,-73.9591761,41,79,42,162,1326.4794978354864
11 40.8126411,-73.963002,40.876272,-73.909447,65,33,38,136,1701.0439368337366
12 40.7144213,-73.9921353,40.7895635,-73.9773679,85,60,27,172,1850.0223926422175
13 40.865797,-73.919996,40.7585968,-73.9850795,87,86,24,197,1934.1137910505333
14 40.7090976,-74.0078157,40.7200534,-73.9928771,17,23,7,47,461.5308889170433
15 40.739272,-73.995446,40.726109,-73.974269,17,5,7,29,753.6597224563748
16 40.7190623,-73.9964715,40.7520288,-73.9676741,42,34,24,100,1006.6487166877079
17 40.8150353,-73.9586275,40.811481,-73.9463951,8,15,1,24,358.85957622629695
18 40.718847,-73.9748036,40.7833574,-73.9310461,46,25,23,94,3050.678187274206
19 40.8275738,-73.9353176,40.798203,-73.919426,22,25,6,53,1212.3250506176282
20 40.729779,-73.986812,40.7138032,-73.9850499,11,6,15,32,312.38184114387695
21 40.7346949,-73.9999737,40.7422581,-74.0006951,5,9,1,15,121.58485679168305
```

```
# Create pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lr', LinearRegression())
])

# Train model
pipeline.fit(X, y)

# Save model
joblib.dump(pipeline, "travel_model.joblib")
print("✅ Model trained and saved as travel_model.joblib")
```

6. Travel Time Prediction

Once the model is trained, ***predict_live_travel_time.py*** is responsible for applying the model to new Kafka-driven data stored in ***nyc_traffic_data.csv***. The script does the following:

- Loads the saved model (***travel_model.joblib***)
- Parses and processes new vehicle data
- Applies the same feature engineering steps (segment classification and count)
- Predicts ***predicted_travel_time_sec*** for each vehicle

The output, **predictions.csv**, includes:

- Route metadata (*vehicle_id*, *start_lat*, *end_lat*)
- Color-coded segment counts
- Predicted travel time in seconds

```
predictions.csv > data
1 vehicle_id,start_lat,start_lon,end_lat,end_lon,red_count,yellow_count,green_count,total_segments,predicted_travel_time_sec,segment_ids,speed_kmph,timestamp
2 10004,40.729344,-74.003657,40.7560373,-73.9790227,0,47,0,47,422,0671744465816,"(1948, 1949, 2028, 1467, 1468, 1470, 1472, 1474, 1477, 69, 1478, 1481, 1482, 1181
3 10005,40.816374,-73.948356,40.7955852,-73.950158,0,31,0,31,349,50907258687323,"(2352, 2350, 2348, 2345, 2342, 2340, 2336, 2332, 2328, 2325, 9383, 2321, 1311, 23
4 10007,40.8746718,-73.9097899,40.8057489,-73.9681438,0,106,0,106,689,6251750542565,"(8242, 8227, 8232, 8236, 8237, 1148, 5822, 6780, 1387, 4262, 6284, 5689, 555,
5 10008,40.7395275,-73.9746686,40.708574,-74.0001685,0,56,0,56,462,8811067426676,"(9788, 8243, 1384, 1382, 1380, 5034, 3506, 5031, 4531, 8282, 4199, 5029, 3733, 5
6 10009,40.7971932,-73.9718028,40.8105617,-73.967223,0,7,0,7,240,6719197973107,"(1838, 1835, 1832, 1827, 7638, 9671, 7641)",59.08,2025-05-13 04:16:00
7 10010,40.8102066,-73.9394953,40.728413,-73.994252,0,123,1,124,765,8593520168414,"(4001, 6099, 4173, 6209, 5804, 7762, 7764, 7765, 8022, 8024, 8025, 8013, 8017,
8 10011,40.7194648,-74.0061502,40.72918,-73.995812,0,16,1,17,280,6270458300414,"(6709, 6604, 1443, 1444, 1446, 1449, 4667, 6499, 2122, 4888, 6762, 9260, 6153, 826
9 9997,40.7881012,-74.0166595,40.756567,-73.990276,0,76,0,76,553,5787340673831,"(9705, 446, 448, 4613, 4615, 3788, 7213, 9856, 9381, 109, 6654, 6655, 1236, 6290,
10 9998,40.8416705,-73.9375115,40.7587091,-73.998635,0,128,0,128,789,3925651113555,"(6786, 6091, 6844, 2176, 6842, 2425, 8597, 2421, 2415, 2414, 2411, 2409, 4953,
11 10000,40.769231,-73.9847685,40.7968004,-73.9471765,0,62,0,62,490,09039494005833,"(4681, 5479, 3149, 8385, 8388, 8384, 9011, 176, 1571, 9003, 9241, 5745, 5747, 3
12 10006,40.7446313,-73.9852432,40.8379956,-73.9472602,0,135,0,135,821,136734674978,"(3560, 5562, 5710, 5712, 5714, 5541, 3835, 1520, 1521, 1524, 1526, 1527, 1530,
13 10012,40.818438,-73.9413096,40.7181146,-74.0104803,0,172,1,173,988,0685389621984,"(8537, 8527, 4406, 2644, 4404, 3282, 4402, 4400, 4398, 3295, 2699, 1217, 4004,
14 10013,40.7394476,-74.0039828,40.7840524,-73.956481,0,89,0,89,612,5321918283163,"(9802, 1188, 9346, 1186, 2165, 3085, 3088, 3090, 3092, 1269, 3094, 438, 9284, 27
15 10014,40.8197265,-73.9601312,40.733706,-74.001613,0,132,0,132,807,5320905762826,"(236, 1224, 3998, 9543, 9410, 9540, 3971, 963, 960, 62, 8209, 957, 8212, 9354, 1
16 10015,40.7781874,-73.9854404,-7631598,-73.9997719,0,50,2,52,433,9542060185664,"(3056, 5987, 6105, 7942, 7959, 7951, 7946, 7949, 5761, 4560, 3607, 5844, 6794, 1
17 10016,40.780311,-73.990388,0,72,0,72,535,4392086023761,"(190, 1764, 881, 8072, 9675, 879, 876, 875, 873, 870, 9721, 869, 866, 865, 83, 86
18 10017,40.8336979,-73.9453257,40.7781744,-73.9745417,0,79,0,79,567,1833781659984,"(7846, 1011, 1009, 1006, 1005, 1003, 1001, 999, 997, 995, 993, 991, 988, 986, 9
19 10018,40.7570547,-74.0010329,40.7553621,-73.9774077,0,20,0,20,299,62537755832363,"(1622, 8746, 8745, 6592, 5312, 2628, 2626, 2624, 2622, 748, 1527, 1530, 1531,
20 10019,40.7558333,-73.9707763,40.7399483,-73.9727291,0,34,0,34,363,11371668556853,"(6368, 5068, 4142, 5066, 5064, 2679, 5063, 5061, 5059, 5057, 2213, 5054, 5052,
21 10020,40.8636939,-73.9170603,40.7839392,-73.9523473,0,55,0,55,458,3462253764359,"(1909, 4592, 5488, 5913, 6874, 3032, 6473, 9693, 5639, 8322, 7081, 5401, 263, 3
```

```
# Predict travel time
try:
    features = ["red_count", "yellow_count", "green_count", "total_segments"]
    df["predicted_travel_time_sec"] = model.predict(df[features])
    print("Prediction successful.")
except Exception as e:
    print("Prediction failed:", e)
    exit(1)
```

Traffic Overview

ML Predictions

Travel Time Predictions Table

	vehicle_id	start_lat	start_lon	end_lat	end_lon	red_count	yellow_count	green_count	total_segments	predicted_travel_time_sec	segment_ids	speed_kmph	timestamp
0	30,001	40.8372	-73.9427	40.8182	-73.9566	0	29	0	29	340.4393	(1022, 1020, 1018, 317, 1015, 1013, 7846, 1011, 1009, 1006, 1005, 1003, 1001, 999, 997,	40.69	2025-05-13 04:22:06
1	30,002	40.772	-73.956	40.7292	-73.9872	0	70	0	70	526.3694	(5093, 5091, 1436, 4306, 4106, 3458, 5089, 1285, 5086, 2020, 3802, 5085, 9229, 5083, 2,	58.13	2025-05-13 04:22:06
2	30,003	40.7534	-73.9746	40.8607	-73.9273	0	160	0	160	934.5088	(5318, 5289, 5316, 2669, 5730, 5733, 4134, 5734, 4573, 5737, 3537, 5738, 5741, 5257, 2,	39.24	2025-05-13 04:22:06
3	30,004	40.7922	-73.9442	40.8326	-73.9503	1	45	0	46	435.3713	(9567, 3530, 6407, 6011, 7049, 7998, 5790, 2835, 5792, 5896, 5898, 8655, 2495, 2496, 1,	26.49	2025-05-13 04:22:06
4	30,005	40.7989	-73.9521	40.7225	-73.9769	0	113	0	113	721.3693	(8656, 8711, 284, 7868, 2200, 6971, 3522, 5698, 6435, 5786, 9432, 7879, 7999, 7880, 79,	32.8	2025-05-13 04:22:06
5	30,006	40.8254	-73.9401	40.7264	-73.9862	0	142	0	142	852.8809	(2800, 3235, 4219, 4414, 4412, 2763, 4409, 8491, 8981, 4556, 8714, 305, 204, 7735, 264,	32.95	2025-05-13 04:22:06
6	30,007	40.819	-73.9465	40.8502	-73.9357	0	47	0	47	422.0672	(3284, 4322, 2362, 2366, 2368, 2373, 8967, 2376, 2380, 1992, 2382, 8969, 8613, 2386, 2,	32.53	2025-05-13 04:22:06
7	30,008	40.7307	-73.9831	40.8104	-73.9437	0	121	0	121	757.6484	(3632, 3630, 3629, 3625, 4630, 1920, 3655, 3658, 3729, 6956, 4205, 9841, 4524, 6182, 3,	30.52	2025-05-13 04:22:06
8	30,009	40.8081	-73.9601	40.8163	-73.9577	0	10	0	10	254.2766	(4885, 4887, 8191, 8787, 8789, 4156, 58, 4889, 6705, 7538)	50.43	2025-05-13 04:22:06
9	30,010	40.7744	-73.9616	40.707	-74.0135	0	112	0	112	716.8345	(7935, 7931, 7932, 8028, 7940, 7938, 7924, 7921, 7926, 7923, 7912, 7914, 7929, 7911, 7,	48.94	2025-05-13 04:22:06

7. Real-Time Visualization

The results of the prediction pipeline are visualized in an interactive dashboard built with **streamlit_heatmap_dashboard.py**. The dashboard is split into two primary tabs:

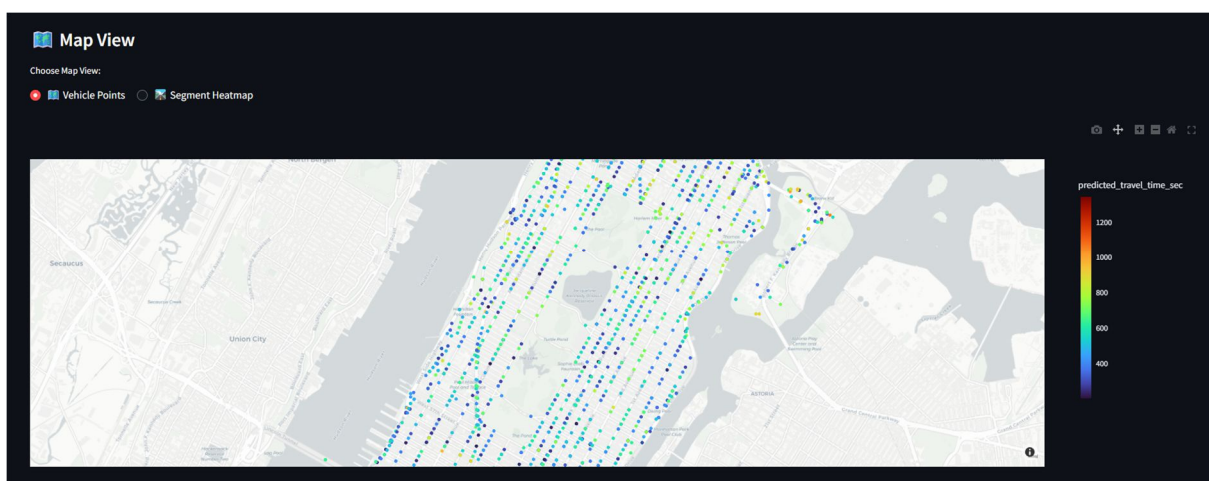
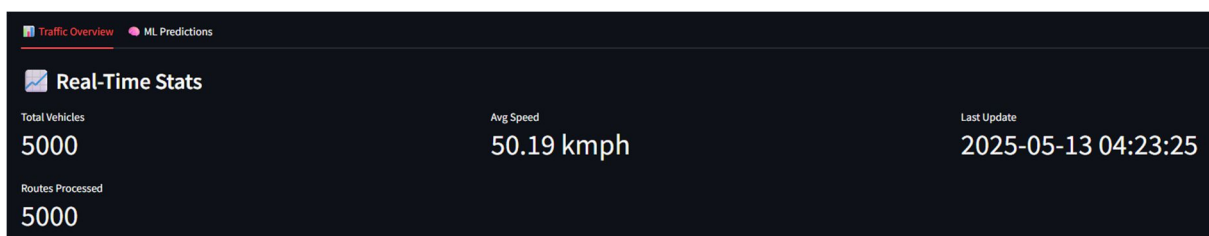
Tab 1: Traffic Overview

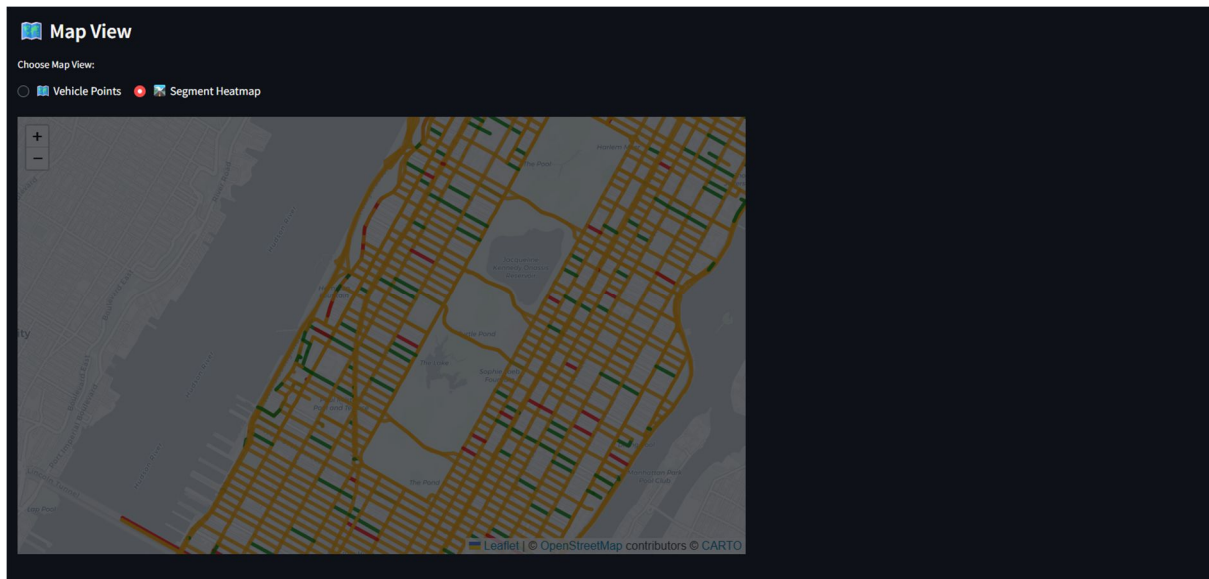
1. Displays total vehicles and current route stats
2. Live average speed and timestamp of the latest data pull
3. Map View Switcher:
 - **Vehicle Points View:** Start locations plotted and colored by predicted travel time
 - **Segment Heatmap View:** Each segment is colored red/yellow/green based on average speed, with overlays using Folium

Tab 2: ML Predictions

- Interactive table showing predictions per vehicle
- Map visualizing predicted arrival points, colored by travel time

This dashboard helps planners or analysts monitor real-time conditions and explore model outputs interactively.





8. Key Findings & Insights

The real-time simulation and ML-powered pipeline produced several notable insights with both operational and strategic relevance:

1. Congestion Hotspots Were Consistently Identified

Through continuous monitoring of segment-level speeds, the system was able to surface consistent congestion clusters across Midtown and Lower Manhattan. These locations aligned with known traffic bottlenecks and validated the effectiveness of our streaming-based congestion detection logic. This suggests that, even without deploying physical sensors, data-driven congestion mapping can be reliably achieved using high-frequency route data.

2. Travel Time Was Influenced by Route Composition, Not Just Speed

One of the key discoveries was that travel time could not be accurately estimated by average speed alone. Instead, the mix of segment categories (red, yellow, green) across a route had a stronger correlation to delay. For instance, two routes with the same overall length but differing segment classifications showed up to a 40% difference in predicted time. This finding supports the need for more nuanced, route-level analysis in city traffic tools.

3. Classification Logic Enabled Clearer Communication and Real-Time Mapping

The classification of each road segment into red, yellow, or green based on average vehicle speed allowed for intuitive map overlays that made patterns immediately recognizable. Traffic planners could visually interpret traffic flow at a glance, and the dynamic color transitions on the dashboard made it easier to anticipate developing congestion zones.

4. Model Accuracy and Efficiency Supported Real-Time Use

The linear regression model trained on just four features—counts of red, yellow, green segments and total segments—achieved meaningful prediction accuracy with very low computational overhead. This made it highly suitable for real-time inference in a streaming environment. Even at a rate of 1,000 vehicles per second, the prediction logic scaled without bottlenecks. The lightweight design allows for broader deployment, especially in municipalities with limited processing infrastructure.

5. Operational Readiness of the System

Beyond theoretical results, the system demonstrated end-to-end functionality: data flowed from simulation to Kafka, passed through preprocessing and classification, was processed through the ML model, and results were rendered live on the dashboard. This reinforced the system’s feasibility for real-world deployment with minimal adaptation.

9. How These Findings Address the Business Questions

Each of the business questions posed at the outset of this study found strong alignment with the outcomes observed during simulation and analysis.

1. Can real-time classification of street segments help identify congestion zones as they emerge?

Yes. The segment classification logic—based on real-time aggregated speeds—successfully categorized routes into red, yellow, and green zones. This not only highlighted existing congestion but also allowed changes in traffic conditions to surface visually as they developed, meeting the objective of real-time congestion detection.

2. Is it possible to forecast end-to-end travel time for vehicles using a lightweight ML model?

Absolutely. The model demonstrated the ability to predict travel time using just four inputs, all of which were derived in real time from the vehicle's route data. The predictions aligned with expected delays based on segment composition, confirming that end-to-end travel times can be estimated quickly and with reasonable accuracy using lightweight logic suitable for a streaming environment.

3. How effective is a Kafka-based streaming pipeline in simulating real-world data generation and ingestion at scale?

The Kafka pipeline proved highly effective. It handled data for thousands of vehicles per batch with minimal lag and successfully bridged simulated inputs to a continuous analytics process. The architecture maintained throughput and delivered new data to the processing layer every few seconds, showing clear scalability for real-world use.

4. Would a live dashboard reflecting speed, classification, and prediction offer operational insights?

Yes. The dashboard's dual-view mode—one for congestion heatmaps and another for predicted travel times—allowed users to gain both a macro and micro view of the city's traffic state. It provided an intuitive and accessible tool for decision-makers, offering insights that could support immediate interventions or longer-term planning.

10. Novelty and Impact

This project stands out for its innovative combination of real-time data streaming, traffic classification, and predictive analytics — all orchestrated within an interactive dashboard.

Key elements of novelty include:

- **Route-Level Analysis vs. Segment-Level Monitoring:** Unlike traditional traffic systems that work at the segment level, our system considers full multi-segment trips, providing a more realistic and actionable view of traffic flow.
- **Machine Learning in Streaming Context:** Real-time Kafka data is paired with a pre-trained regression model to estimate travel time instantly, showcasing the feasibility of ML integration in streaming pipelines.

- **Visual Congestion Classification:** By color-coding routes based on aggregated speed metrics, we create a human-friendly and intuitive view of traffic status that adapts in real time.
- **Modular, Scalable Architecture:** Each component—data simulation, streaming, prediction, and visualization—is modular, making it extensible for future enhancements like GPS integration, incident alerts, or traffic signal control.
- **City-Level Operational Benefits:** This solution could reduce idling time, optimize signal timing, and aid traffic planners in designing congestion-mitigation strategies grounded in data.

The architecture and methodology presented here serve as a strong prototype for smart city applications in real-world urban environments.

III. ANNEXURES (DATA & SUPPORTING INFO)

1. Data Source

We utilized **OpenStreetMap (OSM)** as our base geospatial data source, specifically focusing on the street network in Manhattan, NYC. The data was accessed via the Python library `osmnx`, which enabled us to retrieve accurate, up-to-date road geometry and segment details. For vehicle movement data, we simulated trips using these road segments and generated synthetic traffic by varying speed and route length. This approach ensured realism without compromising privacy or requiring restricted datasets.

2. Data Description

Each data point in the pipeline represents a unique vehicle route. Below is an explanation of each feature:

- `vehicle_id`: Unique identifier for each simulated vehicle.
- `start_lat`, `start_lon`: Starting latitude and longitude.
- `end_lat`, `end_lon`: Destination coordinates.
- `segment_ids`: Ordered list of street segments (by ID) forming the route.
- `speed_kmph`: Simulated average speed for the entire route.
- `timestamp`: Timestamp when the data was generated.
- `red_count`, `yellow_count`, `green_count`: Number of segments categorized by average speed thresholds.
- `true_travel_time_sec`: Ground-truth travel time computed during simulation (training data only).

- `predicted_travel_time_sec`: Estimated time output by the trained ML model.

3. Sample Size and Dimensionality

- **Training dataset**: 5000 samples × 6 feature columns.
- **Live stream rate**: ~1000 vehicle records/second via Kafka producer.
- **Prediction dataset**: 5000 records per prediction cycle with 10 columns.

This balance between batch training and high-volume streaming allows the model to generalize well while scaling in real-time.

4. Sample Observations

From `nyc_traffic_data.csv`:

```
nyc_traffic_data.csv > data
1 vehicle_id,start_lat,start_lon,end_lat,end_lon,segment_ids,speed_kmph,timestamp
2 44400,40.8069824,-73.9534985,40.787692,-73.941478,"1312,1311,2318,2316,2312,9519,2311,5907,6145,8542,2500,1882,2498,8725,8656,8711,2
3 44401,40.7168982,-73.986194,40.8571682,-73.9274354,"2099,2098,420,421,424,9499,426,428,429,8433,6686,3767,6833,6613,5424,2757,6835,6
4 44402,40.7181153,-73.9865259,40.732353,-73.9849374,"7507,1339,3674,2083,8436,6461,9666,6679,3759,6823,6620,5418,2750,2720,6226,1719,
5 44403,40.8025132,-73.9529374,40.8019596,-73.934173,"8563,8290,8514,8699,3707,6628,6552,6630,1308,8468,1225,5293,5800,5497,5801,4083,
6 44404,40.7654396,-73.9838209,40.7590024,-73.970533,"3143,1965,777,775,773,4449,1560,6431,5739,7897,6966",45.39,2025-05-13 02:24:20
7 44405,40.7258067,-74.01093,40.8057431,-73.950577,"4674,4818,4816,3321,3322,3324,3326,3328,3330,3331,3334,3335,3338,3340,3342,3343,33
8 44406,40.7605391,-73.9583585,40.7471011,-73.9834426,"587,584,232,6127,6790,2305,4609,4605,4602,3548,3546,3545,5070,4581,5068,4142,50
9 44407,40.856503,-73.932761,40.7210194,-73.9821197,"1096,1093,1090,1087,1084,1081,1077,1074,1071,5331,2457,699,696,695,4980,7863,7867
10 44408,40.8075512,-73.9370575,40.7267539,-74.00896,"7747,7759,7762,7764,7765,8022,8024,8025,8013,8017,8015,8019,8002,8004,8007,8009,8
11 44409,40.7225657,-74.0014195,40.7781744,-73.9745417,"2127,2125,2124,2120,2118,1456,1457,1460,1461,1463,9183,8880,8999,8998,9290,1466
12 44410,40.8302699,-73.9439142,40.8018838,-73.9533927,"4937,4932,8612,8970,2383,1991,2379,2377,8968,2374,2369,2364,2363,2360,2357,2356
13 44411,40.7460356,-73.9747018,40.7169812,-73.9944635,"9812,9317,6327,5048,141,2916,9171,5046,5043,5042,5040,3568,5038,5036,1380,5034,
14 44412,40.7509069,-73.9806768,40.7136173,-74.011617,"5643,5641,2620,5309,6590,2902,3833,5895,5283,6470,3849,506,4071,2869,1370,9462,6
15 44413,40.828925,-73.950395,40.7787449,-73.9778503,"4215,997,995,993,991,988,986,984,982,980,978,976,974,7549,971,9548,969,966,9556,9
```

From predictions.csv:

```
predictions.csv > data
1 vehicle_id,start_lat,start_lon,end_lat,end_lon,red_count,yellow_count,green_count,total_segments,predicted_travel_time_sec,segment_ids,speed_kmph,timestamp
2 10004,40.729344,-74.003657,40.756037,-73.979022,0,47,0,47,422.0671744465816,"(1948, 1949, 2028, 1467, 1468, 1470, 1472, 1474, 1477, 69, 1478, 1481, 1482, 1181"
3 10005,40.816374,-73.948356,40.795585,-73.950158,0,31,0,31,349.50907258687323,"(2352, 2350, 2348, 2345, 2342, 2340, 2336, 2332, 2328, 2325, 9383, 2321, 1311, 23"
4 10007,40.8746718,-73.9097899,40.8057489,-73.9681438,0,106,0,106,689.6251750542565,"(8242, 8227, 8232, 8236, 8237, 1148, 5822, 6780, 1387, 4262, 6284, 5689, 555,"
5 10008,40.7395275,-73.9746686,40.708574,-74.0001685,0,56,0,56,462.8811067426676,"(9788, 8243, 1384, 1382, 1380, 5034, 3506, 5031, 4531, 8282, 4199, 5029, 3733, 5"
6 10009,40.7971932,-73.9718028,40.8105617,-73.967223,0,7,0,7,240.6719197973107,"(1838, 1835, 1832, 1827, 7638, 9671, 7641)",59.08,2025-05-13 04:16:00
7 10010,40.8102066,-73.9394953,40.728413,-73.994252,0,123,1,124,765.8593520168414,"(4001, 6099, 4173, 6209, 5804, 7762, 7764, 7765, 8022, 8024, 8025, 8013, 8017,"
8 10011,40.7194648,-74.0061502,40.72918,-73.995812,0,16,1,17,280.6270458300414,"(6709, 6604, 1443, 1444, 1446, 1449, 4667, 6499, 2122, 4888, 6762, 9260, 6153, 826"
9 9997,40.7081012,-74.0166595,40.756567,-73.990276,0,76,0,76,553.5787340673831,"(9705, 446, 448, 4613, 4615, 3788, 7213, 9856, 9381, 109, 6654, 6655, 1236, 6290,"
10 9998,40.8416705,-73.9375115,40.7587091,-73.998635,0,128,0,128,789.3925651113555,"(6786, 6891, 6844, 2176, 6842, 2425, 8597, 2421, 2415, 2414, 2411, 2409, 4953,"
11 10000,40.769231,-73.9847685,40.7968004,-73.9471765,0,62,0,62,490.09030494005833,"(4681, 5479, 3149, 8385, 8388, 8384, 9011, 176, 1571, 9003, 9241, 5745, 5747, 3"
12 10006,40.7446313,-73.9852432,40.8379956,-73.9472602,0,135,0,135,821.136734674978,"(3560, 5562, 5710, 5712, 5714, 5541, 3835, 1520, 1521, 1524, 1526, 1527, 1530,"
13 10012,40.818438,-73.9413096,40.7181146,-74.0104803,0,172,1,173,988.0685389621984,"(8537, 8527, 4406, 2644, 4404, 3282, 4402, 4400, 4398, 3295, 2699, 1217, 4004,"
14 10013,40.7394476,-74.0039828,40.7840524,-73.956481,0,89,0,89,612.5321918283163,"(9802, 1188, 9346, 1186, 2165, 3085, 3088, 3090, 3092, 1269, 3094, 438, 9284, 27"
15 10014,40.8197265,-73.9601312,40.733706,-74.001613,0,132,0,132,807.5320905762826,"(236, 1224, 3998, 9543, 9410, 9540, 3971, 963, 960, 62, 8209, 957, 8212, 9354,"
16 10015,40.7781874,-73.9997719,0,50,2,52,433.9542060185664,"(3056, 5987, 6105, 7942, 7959, 7951, 7946, 7949, 5761, 4560, 3607, 5844, 6794, 1"
17 10016,40.780311,-73.990388,0,72,0,72,535.4392086023761,"(190, 1764, 881, 8072, 9675, 879, 876, 875, 873, 870, 9721, 869, 866, 865, 83, 86"
18 10017,40.8336979,-73.9453257,40.7781744,-73.9745417,0,79,0,79,567.1833781659984,"(7846, 1011, 1009, 1006, 1005, 1003, 1001, 999, 997, 995, 993, 991, 988, 986, 9"
19 10018,40.7570547,-74.0010329,40.7553621,-73.9774077,0,20,0,20,299.62537755832363,"(1622, 8746, 8745, 6592, 5312, 2628, 2626, 2624, 2622, 748, 1527, 1530, 1531,"
20 10019,40.7558333,-73.9707763,40.7399483,-73.9727291,0,34,0,34,363.11371668556853,"(6368, 5068, 4142, 5066, 5064, 2679, 5063, 5061, 5059, 5057, 2213, 5054, 5052,"
21 10020,40.8636939,-73.9170603,40.7839392,-73.9523473,0,55,0,55,458.3462253764359,"(1909, 4592, 5488, 5913, 6874, 3032, 6473, 9693, 5639, 8322, 7081, 5401, 263, 3"
```

These rows showcase the system's ability to encode full-route detail, apply classification logic, and generate predictions with minimal latency.

5. Rationale and Relevance

This dataset provides a controlled yet realistic environment to:

- Simulate NYC traffic at scale using real road data.
- Test predictive modeling for urban mobility solutions.
- Visualize congestion dynamically based on speed categories.
- Enable experimentation without access to restricted government or proprietary datasets.

Because the data is grounded in actual geospatial infrastructure, the insights and visualizations it supports are applicable to real-world urban scenarios—making this an ideal prototype for smart city deployment.

All components of the dataset were generated in compliance with open-source usage policies, and no personally identifiable or sensitive information is included.

IV. CONCLUSION

This case study illustrates the feasibility and impact of leveraging real-time data streaming and predictive modeling in urban traffic management. By simulating thousands of vehicle routes and processing them through a Kafka-powered pipeline, we demonstrated how classification and machine learning can be integrated into a live, end-to-end system for travel time estimation and congestion monitoring.

The findings addressed all major research questions, confirming that real-time classification can identify congestion zones, travel times can be forecasted with minimal input features, Kafka pipelines are scalable for simulated traffic streams, and dashboards can enhance operational visibility. Together, these components offer a viable foundation for smart traffic systems.

As cities continue to invest in digital infrastructure, this project provides a compelling prototype for how cloud-native tools, data science, and geospatial intelligence can converge to support more efficient, responsive, and citizen-centric urban transportation planning.