Resource Allocation Optimization and Predictive Analytics for Crime and Accidents based on 911 NYPD Incident Data.

BIA 678 A
Professor Venu Guntupalli

Team H
By:
Preeti Chougule (20027873)
Smeet Nalawade (20033924)
Vrushali Khatane (20027219)

Table of Contents

Introduction

The New York Police Department (NYPD) faces growing challenges in ensuring timely responses to emergency calls, with response times reaching historic highs. Efficient resource allocation and response time prediction are crucial to address these issues and enhance public safety. By analyzing patterns in 911 call data, such as peak hours, high-demand areas, and incident trends, this project aims to develop data-driven strategies to optimize resource deployment and reduce response delays. Leveraging predictive models and clustering techniques, this analysis provides actionable insights to improve operational efficiency and ensure quicker responses to emergencies.

Problem Statement and Objectives

Problem Statement NYPD has been experiencing increasing **response times** to emergency calls, impacting the efficiency of emergency services and public safety. With millions of incidents reported annually, understanding response delays and optimizing resource allocation are critical challenges that need to be addressed using a data-driven approach.

Objectives

1. **Analyze Emergency Response Times:**
    - Measure and evaluate **dispatch**, **arrival**, and **total response times**.
    - Identify trends in response delays based on **time**, **location**, and **incident type**.
2. **Predict Response Times:**
    - Use machine learning models (Linear Regression, Random Forest, Gradient Boost) to predict based on **total response times**.
3. **Identify High-Risk Zones:**
    - Apply **clustering algorithms** (K-Means) to locate areas with high incident density for efficient **resource allocation**.
4. **Provide Actionable Insights:**
    - Highlight key factors influencing delays.
    - Offer recommendations to improve response efficiency and optimize resource deployment.

Dataset Overview

Dataset Link: https://catalog.data.gov/dataset/nypd-calls-for-service

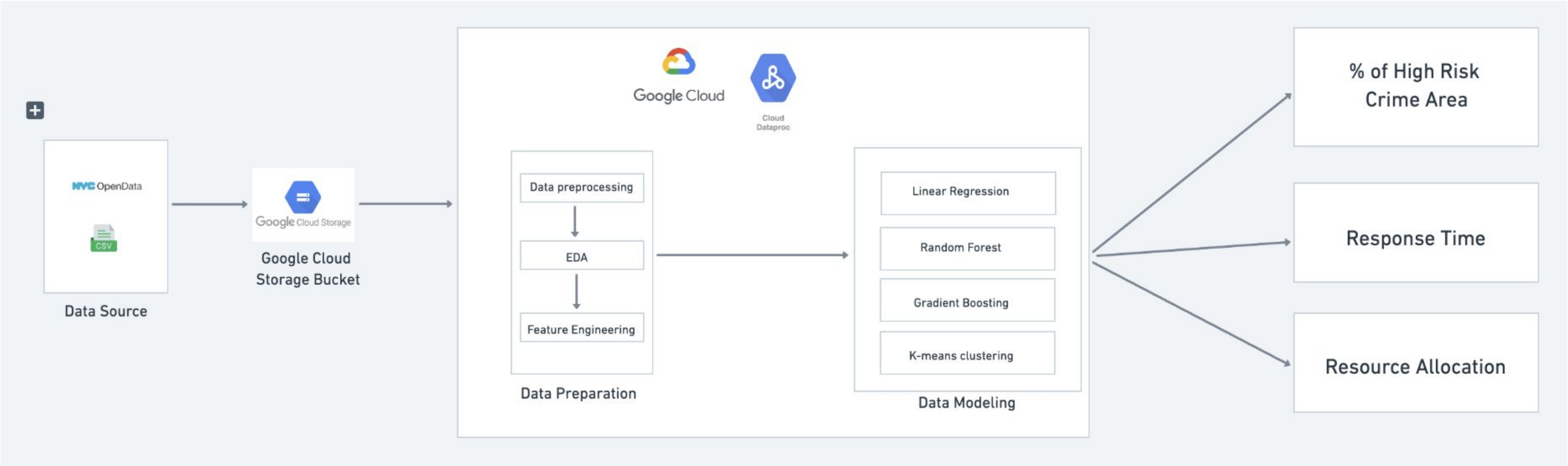
Time Range: Jan 2024 - Oct 2024

Number of rows: 5,430,525

Column features: 18

Feature	Description
CAD_EVNT_ID	Unique identifier for the 911 event.
CREATE_DATE	Date when the event was created in the system.
INCIDENT_DATE	Date of the reported incident.
INCIDENT_TIME	Time when the incident was reported.
NYPD_PCT_CD	NYPD precinct code where the incident occurred.
BORO_NM / PATRL_BORO_NM	Borough name (e.g., Bronx, Brooklyn, Manhattan).
GEO_CD_X / GEO_CD_Y	Geographic coordinates (X, Y) for the incident location.
RADIO_CODE	Code used for radio communications regarding the event.
TYP_DESC	Description of the type of incident (e.g., dispute, alarm).
CIP_JOBS	Classification of job priority (e.g., Critical, Serious).
ADD_TS / DISP_TS	Timestamps for when the call was added and dispatched.
ARRIVD_TS	Timestamp for when units arrived on the scene.
CLOSNG_TS	Timestamp for when the incident was closed.
Latitude / Longitude	Geographic latitude and longitude of the incident.

Flow for Project



Data Cleaning

1. Checked for Duplicate Records:
- Verified that the dataset contained no duplicate rows.
- Ensured data integrity for accurate analysis.
- Handled Missing Values
2. Identified and filtered out rows with missing values in critical fields:
- ARRIVD_TS (Arrival Time): Ensured only valid response times are analyzed.
- CLOSNG_TS (Closing Time): Removed incomplete incident data.
- Replaced missing values in NYPD_PCT_CD (Precinct Code) with "UNKNOWN" to retain useful records.

```
Number of rows with NULL values in ARRIVD_TS: 1138027
Number of rows with NULL values in CLOSNG_TS: 33
Number of rows with NULL values in NYPD_PCT_CD: 1
```

```
Total rows after dropping NULL values in ARRIVD_TS and CLOSNG_TS: 4292472
+--------------+--------------+
|ARRIVD_TS_NULL|CLOSNG_TS_NULL|
+--------------+--------------+
|         false|         false|
|         false|         false|
|         false|         false|
|         false|         false|
|         false|         false|
+--------------+--------------+
only showing top 5 rows
```

3. Dropped Irrelevant Columns:

- Removed columns like CAD_EVNT_ID, PATRL_BORO_NM, and CREATE_DATE that did not contribute to the analysis.

4. Converted Timestamps to Standard Format:

- Transformed columns (ADD_TS, DISP_TS, ARRIVD_TS, CLOSNG_TS) into timestamp format to enable accurate time-based calculations.

```
df.printSchema()
```

```
root
 |-- CAD_EVNT_ID: integer (nullable = true)
 |-- CREATE_DATE: string (nullable = true)
 |-- INCIDENT_DATE: string (nullable = true)
 |-- INCIDENT_TIME: timestamp (nullable = true)
 |-- NYPD_PCT_CD: integer (nullable = true)
 |-- BORO_NM: string (nullable = true)
 |-- PATRL_BORO_NM: string (nullable = true)
 |-- GEO_CD_X: integer (nullable = true)
 |-- GEO_CD_Y: integer (nullable = true)
 |-- RADIO_CODE: string (nullable = true)
 |-- TYP_DESC: string (nullable = true)
 |-- CIP_JOBS: string (nullable = true)
 |-- ADD_TS: string (nullable = true)
 |-- DISP_TS: string (nullable = true)
 |-- ARRIVD_TS: string (nullable = true)
 |-- CLOSNG_TS: string (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
```

```
df_cleaned = df_cleaned.withColumn("ADD_TS", F.to_timestamp("ADD_TS", "MM/dd/yyyy hh:mm:ss a")) \
                       .withColumn("DISP_TS", F.to_timestamp("DISP_TS", "MM/dd/yyyy hh:mm:ss a")) \
                       .withColumn("ARRIVD_TS", F.to_timestamp("ARRIVD_TS", "MM/dd/yyyy hh:mm:ss a")) \
                        .withColumn("CLOSNG_TS", F.to_timestamp("CLOSNG_TS", "MM/dd/yyyy hh:mm:ss a"))

print("Schema after converting to timestamp:")
df_cleaned.printSchema()

df_cleaned.select("ADD_TS", "DISP_TS", "ARRIVD_TS", "CLOSNG_TS").show(5, truncate=False)
```

```
Schema after converting to timestamp:
root
 |-- INCIDENT_DATE: string (nullable = true)
 |-- INCIDENT_TIME: timestamp (nullable = true)
 |-- NYPD_PCT_CD: integer (nullable = true)
 |-- BORO_NM: string (nullable = true)
 |-- GEO_CD_X: integer (nullable = true)
 |-- GEO_CD_Y: integer (nullable = true)
 |-- RADIO_CODE: string (nullable = true)
 |-- TYP_DESC: string (nullable = true)
 |-- CIP_JOBS: string (nullable = true)
 |-- ADD_TS: timestamp (nullable = true)
 |-- DISP_TS: timestamp (nullable = true)
 |-- ARRIVD_TS: timestamp (nullable = true)
 |-- CLOSNG_TS: timestamp (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
```

ADD_TS	DISP_TS	ARRIVD_TS	CLOSNG_TS
2024-01-01 00:01:21	2024-01-01 00:02:19	2024-01-01 01:19:58	2024-01-01 01:20:02
2024-01-01 00:06:11	2024-01-01 00:07:19	2024-01-01 00:19:27	2024-01-01 01:03:22
2024-01-01 00:04:51	2024-01-01 00:09:21	2024-01-01 00:15:11	2024-01-01 00:56:56
2024-01-01 00:04:57	2024-01-01 00:12:08	2024-01-01 00:29:16	2024-01-01 00:29:53
2024-01-01 00:00:07	2024-01-01 00:00:07	2024-01-01 00:00:07	2024-01-01 00:30:23

Feature Engineering

Outliers filtered using the **95th percentile**.

Created New Features to Measure Response Times:

- **dispatch_time:** Time taken between the incident being added (ADD_TS) and dispatched (DISP_TS).
  - Formula: (DISP_TS - ADD_TS) / 60 (in minutes).
- **arrival_time:** Time taken for units to arrive after dispatch.
  - Formula: (ARRIVD_TS - DISP_TS) / 60 (in minutes).
- **total_response_time:** Total time taken from incident addition to unit arrival.
  - Formula: (ARRIVD_TS - ADD_TS) / 60 (in minutes).

```
df_cleaned = df_cleaned.withColumn("dispatch_time", F.round((F.
 ↪unix_timestamp("DISP_TS") - F.unix_timestamp("ADD_TS")) / 60, 2)) \
                     .withColumn("arrival_time", F.round((F.
 ↪unix_timestamp("ARRIVD_TS") - F.unix_timestamp("DISP_TS")) / 60, 2)) \
                     .withColumn("total_response_time", F.round((F.
 ↪unix_timestamp("ARRIVD_TS") - F.unix_timestamp("ADD_TS")) / 60, 2))

df_cleaned.select("ADD_TS", "DISP_TS", "ARRIVD_TS", "dispatch_time",␣
 ↪"arrival_time", "total_response_time").show(10)
```

ADD_TS	DISP_TS	ARRIVD_TS	dispatch_time	arrival_time	total_response_time
2024-01-01 00:01:21	2024-01-01 00:02:19	2024-01-01 01:19:58	0.97	77.65	78.62
2024-01-01 00:06:11	2024-01-01 00:07:19	2024-01-01 00:19:27	1.13	12.13	13.27
2024-01-01 00:04:51	2024-01-01 00:09:21	2024-01-01 00:15:11	4.5	5.83	10.33
2024-01-01 00:04:57	2024-01-01 00:12:08	2024-01-01 00:29:16	7.18	17.13	24.32
2024-01-01 00:00:07	2024-01-01 00:00:07	2024-01-01 00:00:07	0.0	0.0	0.0
2024-01-01 00:00:14	2024-01-01 00:08:24	2024-01-01 00:36:32	8.17	28.13	36.3
2024-01-01 00:00:25	2024-01-01 00:00:25	2024-01-01 00:00:25	0.0	0.0	0.0
2024-01-01 00:00:35	2024-01-01 00:00:35	2024-01-01 00:00:35	0.0	0.0	0.0
2024-01-01 00:05:03	2024-01-01 00:15:06	2024-01-01 00:42:21	10.05	27.25	37.3
2024-01-01 00:00:51	2024-01-01 00:22:17	2024-01-01 00:30:42	21.43	8.42	29.85

only showing top 10 rows

Feature Engineering

Standardized the Data for Modeling:

- Scaled features (dispatch_time and arrival_time) using **StandardScaler** to ensure consistent scaling for machine learning models.

Rounded Spatial Coordinates:

- **Latitude and Longitude** were rounded to 3 decimal places to group incidents in specific areas for clustering analysis.

```
Scaled Features:
+----------------------------------------+-------------------+
|features                                |total_response_time|
+----------------------------------------+-------------------+
|[-0.5321306579097865,2.7423513497455883]|78.62              |
|[-0.4942177402296219,-0.21748906672196322]|13.27            |
|[0.3043230884088464,-0.5020891067669201]|10.33              |
|[0.9393644595516045,0.008383980932764353]|24.32             |
|[1.1739506376976234,0.5053046857731651]|36.3               |
+----------------------------------------+-------------------+
only showing top 5 rows
```

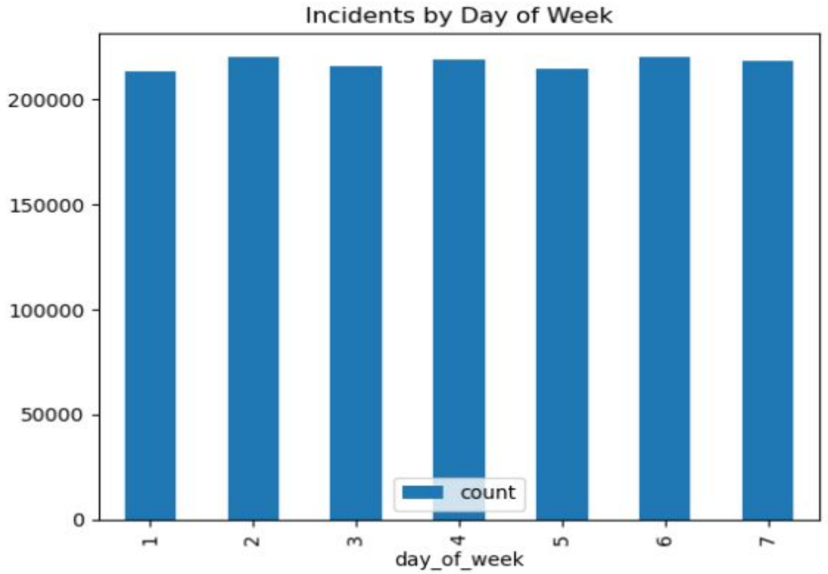
```
df_cleaned_eda = df_cleaned_eda.withColumn("latitude_rounded", F.
 ↪round("Latitude", 3)) \
                              .withColumn("longitude_rounded", F.round("Longitude", 3))

top_locations = df_cleaned_eda.groupBy("latitude_rounded", "longitude_rounded").
 ↪count().orderBy("count", ascending=False).limit(40000).toPandas()
```

Exploratory Data Analysis - Temporal Analysis

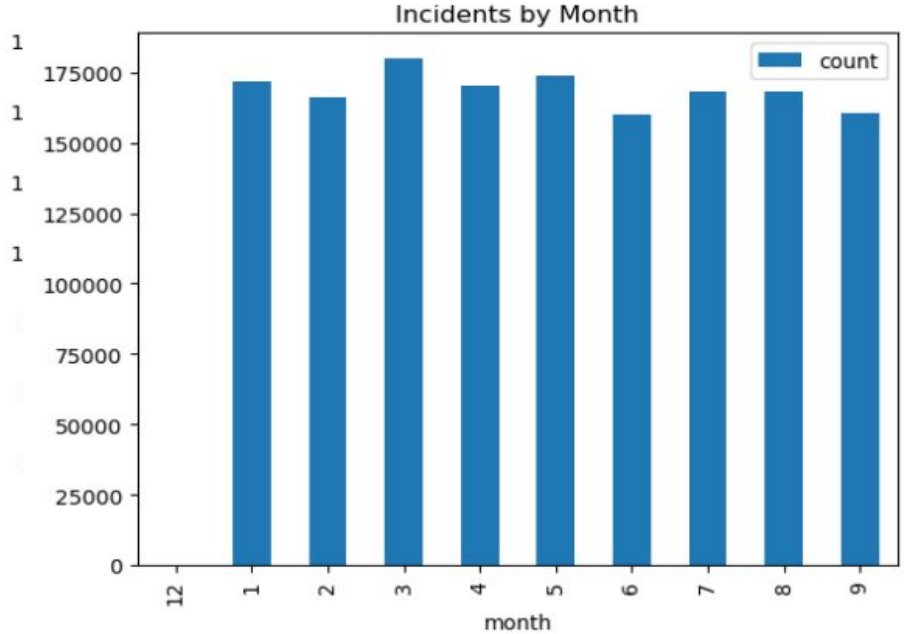
Incidents by Day of Week Analysis:

- Incident volume remains **relatively consistent** across the week, with a slight increase on **weekends**.
- Higher calls on weekends may be related to recreational activities and reduced traffic enforcement.



Incident by Month

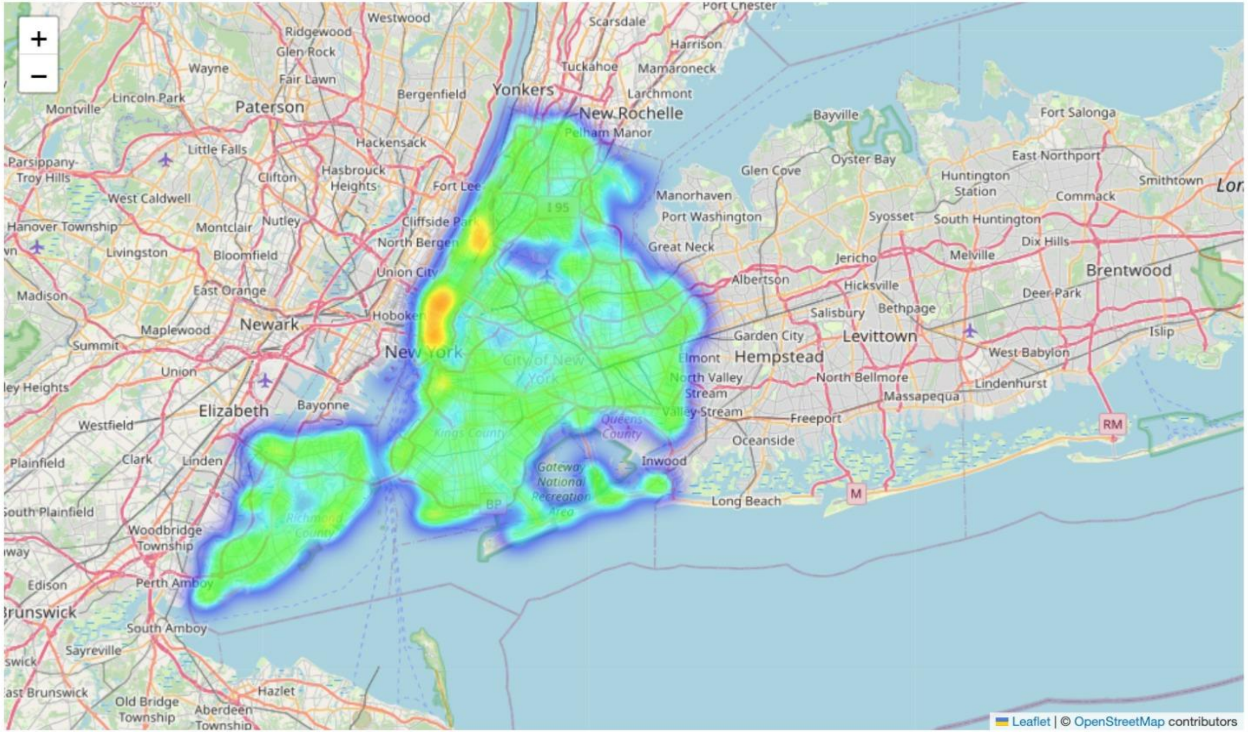
- **March** and **April** show **higher incident counts**, peaking close to **180,000**.
- Incident counts slightly decrease in **June** and **September** but remain above **160,000**.
- **March and April** may align with increased outdoor activities as spring begins, leading to higher emergency calls.
- **Summer months (June–August)** maintain relatively high volumes, possibly due to outdoor events, travel, and increased public activity.



Exploratory Data Analysis - Spatial Analysis

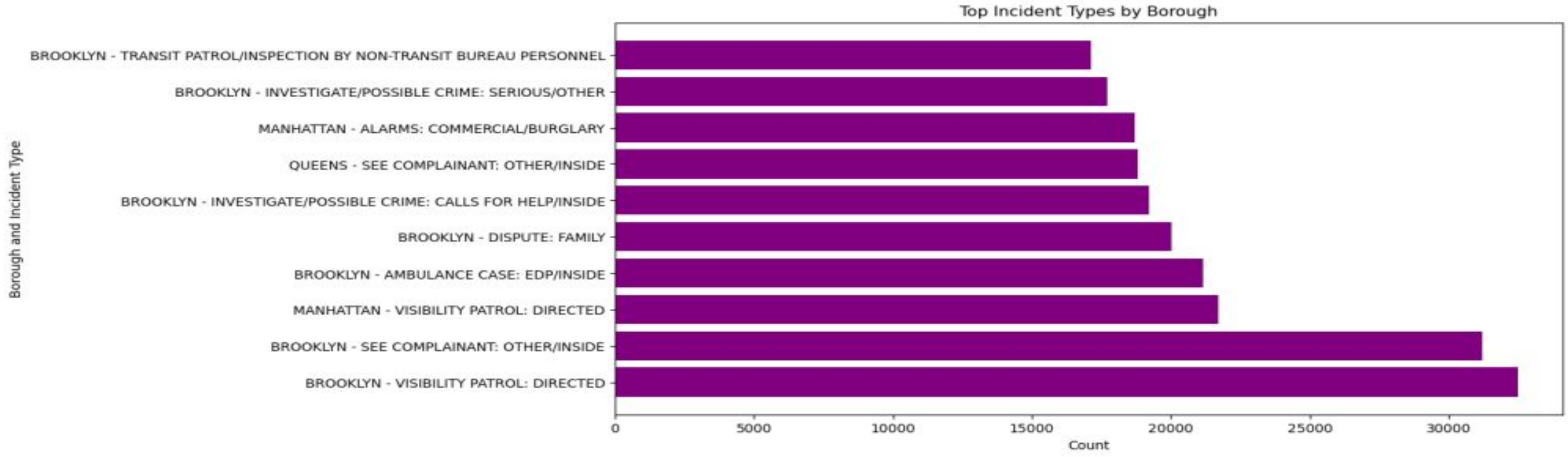
- Hotspots were observed in **central Brooklyn**, **lower Manhattan**, and **northwest Queens**.
- **Insight:** These regions consistently experience a high volume of emergency calls and require optimized resource allocation.

Out[27]:



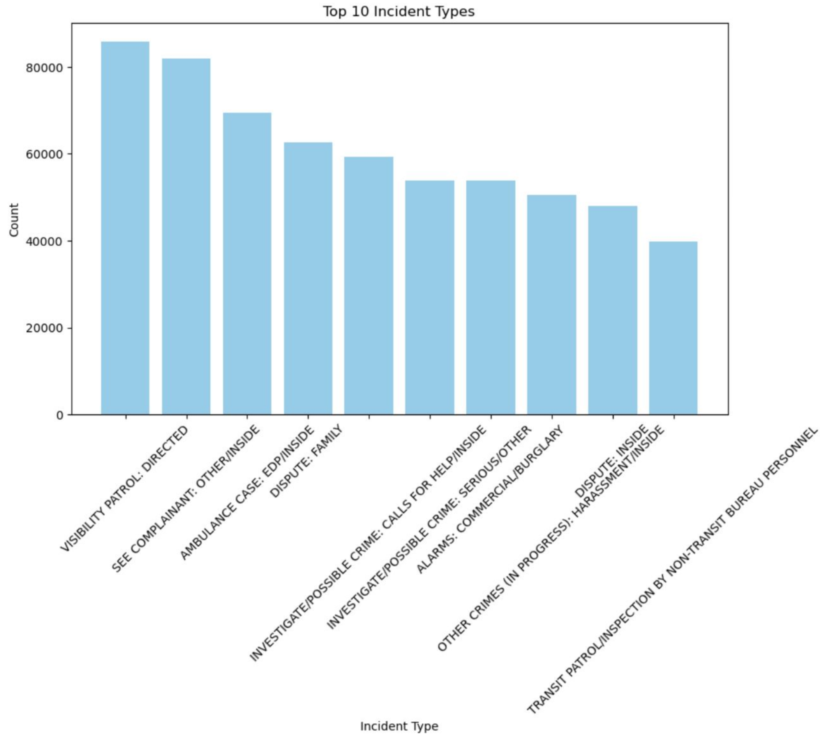
Spatial Analysis

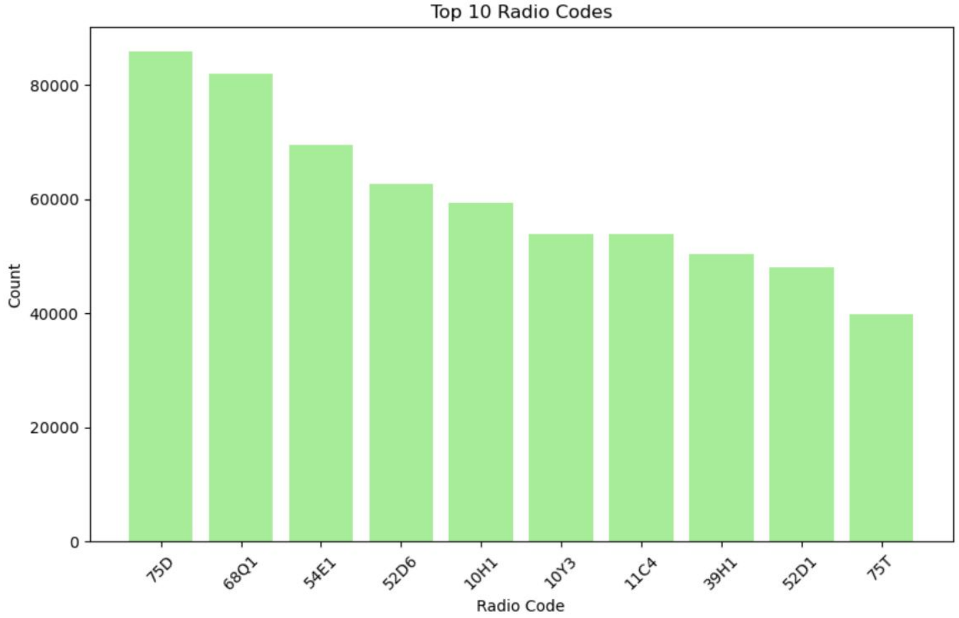
- **Brooklyn** reported the **highest number of incidents**, followed by **Manhattan**.
- **Staten Island** had the **lowest incident volume**.
- Higher population density and activity in Brooklyn contribute to increased emergency calls.



Incident Classification

- The NYPD handles a significant volume of **routine patrols**, **complaints**, and **medical emergencies**, highlighting areas where resources are most needed.
- Radio codes provide further insights into call frequency, helping prioritize and allocate resources effectively.





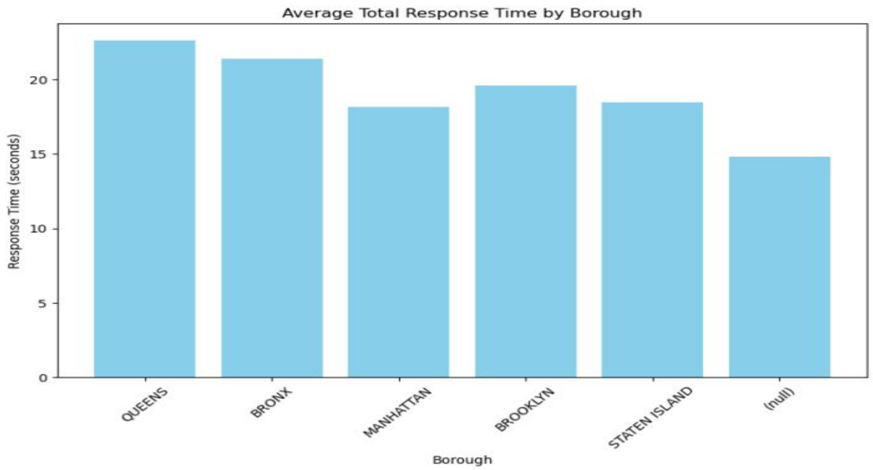
Response Time Analysis

- Response times tend to be **lower at late night and early morning** (e.g., between **2 AM and 6 AM**) with average times around **15–17 minutes**.
- Response times peak during **evening hours** (e.g., **5 PM–8 PM**) with average times exceeding **20 minutes**, likely due to high call volumes and traffic congestion.
- Evening hours pose operational challenges, indicating a need for better resource allocation during peak times.

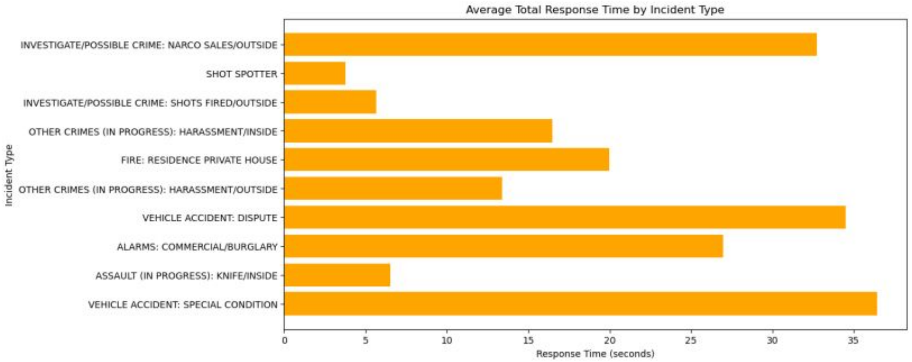
hour	avg(dispatch_time)	avg(arrival_time)	avg(total_response_time)
22	3.1272606028272865	13.53621373542937	16.66363704831049
23	4.341739894208651	18.10833291043063	22.450261354985987
0	2.7815602304781795	14.271719284559152	17.05347188043951
1	2.467993675038265	12.777316433127371	15.245506922132085
2	2.3968036431300566	12.546193712664378	14.94316325531275
3	2.4486705376843902	12.623134130764058	15.07200092904415
4	2.4914846488502476	13.001222871349647	15.492924549409468
5	2.507468121395905	13.641176868723159	16.148846684895787
6	2.6962617963524047	14.082330018061256	16.77873069976859
7	4.150677509717355	23.767868667348697	27.918648237823497
8	2.567756004268294	18.62475393932958	21.192663220479318
9	2.860873968184314	18.695965533266726	21.55695238616037
10	3.125105299245533	18.78806711165481	21.91331332979636
11	3.2958690285840233	19.24268983702451	22.538693511740846
12	3.4051720113375272	19.536395959542755	22.94173150669866
13	3.3526879459464434	18.077859876364244	21.43069893603734
14	3.3539938284241235	15.20779753722193	18.561966129457435
15	4.8294797123040105	20.968853380750698	25.798443622151325
16	3.327067386970393	17.853575964168037	21.180765302356836
17	3.28389922441303	17.46224476605933	20.746284533888502

only showing top 20 rows

Response Time by Borough



- **Queens** has the **highest total response time** (~22.6 minutes), followed closely by **Bronx** (~21.3 minutes).
- **Manhattan** has the **lowest total response time** (~18.1 minutes), indicating faster emergency responses in this borough.
- **Staten Island** (~18.5 minutes) and **Brooklyn** (~19.6 minutes) fall in the mid-range for response times.
- **Insight:** Higher response times in Queens and Bronx may be due to larger geographic areas, traffic conditions, or resource distribution challenges.



Classification Models

Linear Regression: Used to **predict incident volumes**, enabling the estimation of resources required in different areas. This helps in identifying peak demand periods and optimizing resource allocation.

Random Forest: Applied for **classifying incident types**, facilitating the decision-making process for allocating the **appropriate resources** based on the nature of incidents.

Gradient Boosting: Highlights the model's ability to **accurately predict the type and frequency of incidents**. This is critical for ensuring timely deployment of the **right resources** to the **right place** at the **right time**, improving operational efficiency.

K-Means Clustering: Used to **identify clusters of high-risk zones**, allowing for strategic resource placement and optimization. By pinpointing geographical hotspots, it ensures effective coverage and reduces incident response times.

Performance Metrics - Linear Regression

```
Linear Regression Predictions:

+------------+-------------------+-------------------+
|features    |total_response_time|prediction         |
+------------+-------------------+-------------------+
|[0.02,0.02]|0.03                |0.0402676259235627б|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
|[0.02,0.02]|0.03                |0.04026762592356276|
+------------+-------------------+-------------------+
only showing top 10 rows

Root Mean Squared Error (RMSE): 0.004880453984611745
[Stage 84:>
R2 Score: 0.9999999572697764
```

RMSE (Root Mean Squared Error): Measures the model's accuracy by quantifying the difference between the predicted values and the actual values.

R-squared: Indicates the percentage of the response variable variation that is explained by the linear model.

- Used to **predict total response time**
- **R^2 Score: ~0.99**, indicating that the model explains 99% of the variance in the response time.
- **RMSE (Root Mean Square Error): ~0.0048** (as per the notebook), showing very low prediction errors.

Random Forest

features	total_response_time	prediction
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398
[0.02,0.02]	0.03	0.7888153334353398

only showing top 10 rows

```
Root Mean Squared Error (RMSE): 3.678746451618793
R2 Score: 0.9757219146809109
```

Random Forest captures complex, **non-linear relationships** between features like **dispatch time**, **arrival time**, and **total response time**.

Feature importance highlights **dispatch time** and **arrival time** as dominant contributors to the prediction.

R^2 Score: ~0.976, showing that the model explains 97.6% of the variance in total response time.

RMSE (Root Mean Square Error): ~3.64, which is slightly higher compared to Gradient Boosting and Linear Regression.

Gradient Boosting

features	total_response_time	prediction
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646
[0.02,0.02]	0.03	0.032046177496824646

only showing top 10 rows

```
Root Mean Squared Error (RMSE): 1.5631399430441817
R2 Score: 0.9956166066697358
```

Gradient Boosting was employed to improve prediction accuracy over simple regression and classification models. It was evaluated using the above metrics, particularly focusing on how incremental improvements (boosting) can minimize prediction errors.

R^2 Score: ~0.995, the **highest among all models**, indicating that the model explains **99.5% of the variance** in total response time.

RMSE (Root Mean Square Error): ~1.61, showcasing **minimal prediction errors** compared to Random Forest (~3.64) and Linear Regression (~0.0048).

Gradient Boosting delivers the **highest accuracy** and **lowest error** among all models, making it the best choice for predicting total response time.

K-Means Clustering

- Applied **K-Means Clustering** to identify **high-incident zones** using latitude and longitude data.
- This helps map incident-prone areas, facilitating **optimized resource allocation** to regions with high incident density.
- Used **latitude**, **longitude**, and **total response time** to create features for clustering.
- Evaluation was done using the **Davies-Bouldin Index (DBI) and Silhouette Score** to measure clustering quality.

Prepared Data for K-means Clustering:

features	Latitude	Longitude
[40.743037,-73.916826]	40.743037	-73.916826
[40.776057,-73.934906]	40.776057	-73.934906
[40.86433,-73.867393]	40.86433	-73.867393
[40.862274,-73.929562]	40.862274	-73.929562
[40.764566,-73.971757]	40.764566	-73.971757

only showing top 5 rows

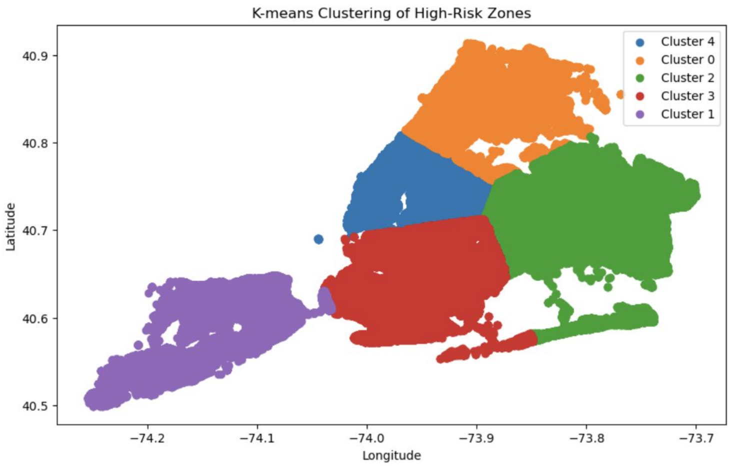
Data with Cluster Assignments:

Latitude	Longitude	cluster
40.743037	-73.916826	4
40.776057	-73.934906	4
40.86433	-73.867393	0
40.862274	-73.929562	0
40.764566	-73.971757	4
40.706102	-73.793242	2
40.740547	-74.008547	4
40.706528	-73.791997	2
40.770813	-73.811147	2
40.70215	-73.790564	2

only showing top 10 rows

K-Means Clustering

```
Cluster Centers (High-Risk Zones):
Cluster 0: [ 40.83743223 -73.90282894]
Cluster 1: [ 40.60199467 -74.12495999]
Cluster 2: [ 40.70022946 -73.81260808]
Cluster 3: [ 40.65558713 -73.95103755]
Cluster 4: [ 40.74828972 -73.96746351]
```



High-Response-Time Zones:

- Cluster **4** (highlighted yellow) shows **higher total response times**, indicating areas that need prioritized attention.

Low-Response-Time Zones:

- Clusters **0** and **1** have **lower response times**, reflecting efficient resource allocation.

Model	Silhouette Score	Davies-Bouldin Index
K- Means Cluster	0.5913598921815876	0.7477194917382969

Model Comparison

Model	RSME	R-squared
Random Forest	3.678746451618793	0.9757219146809109
Gradient Boost	1.5631399430441817	0.9956166066697358

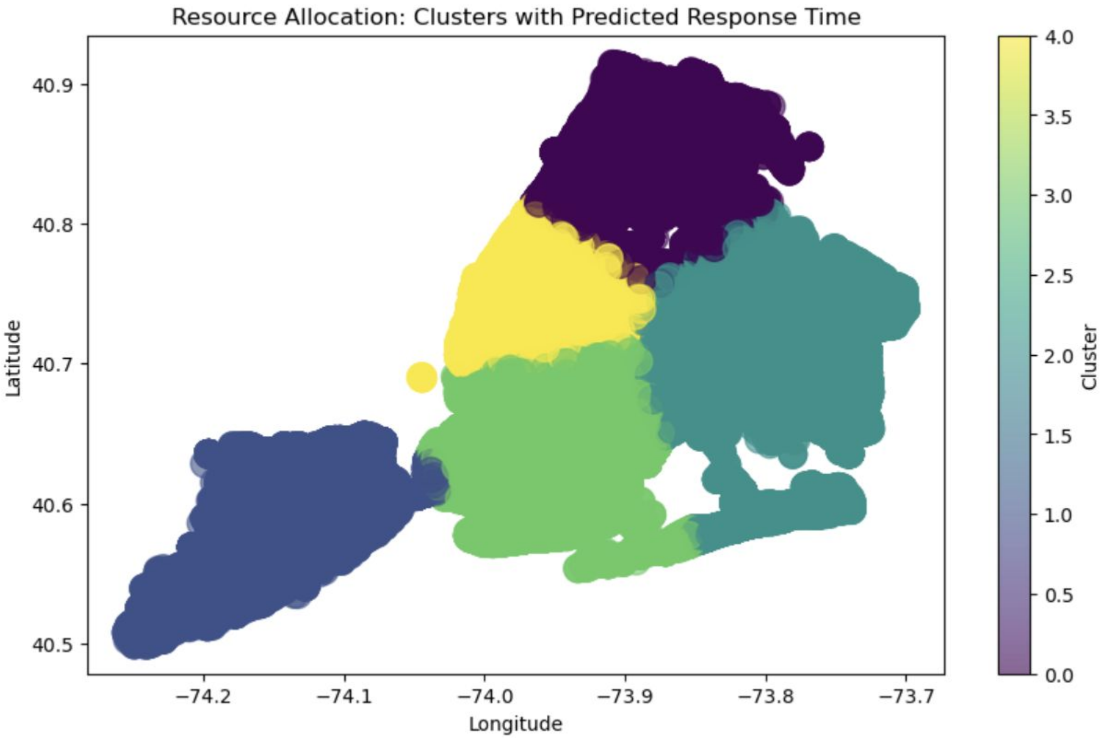
Model	Silhouette Score	Davies-Bouldin Index
K- Means Cluster	0.5913598921815876	0.7477194917382969

Latitude	Longitude	total_response_time	prediction	features	cluster
40.743037	-73.916826	78.62	21.236297410076237	[40.743037,-73.91...	4
40.776057	-73.934906	13.27	21.236297410076237	[40.776057,-73.93...	4
40.86433	-73.867393	10.33	21.236297410076237	[40.86433,-73.867...	0
40.862274	-73.929562	24.32	21.236297410076237	[40.862274,-73.92...	0
40.764566	-73.971757	36.3	21.236297410076237	[40.764566,-73.97...	4

only showing top 5 rows

Latitude	Longitude	total_response_time	prediction
40.743037	-73.916826	78.62	21.236297410076237
40.776057	-73.934906	13.27	21.236297410076237
40.86433	-73.867393	10.33	21.236297410076237
40.862274	-73.929562	24.32	21.236297410076237
40.764566	-73.971757	36.3	21.236297410076237

only showing top 5 rows



Predicted Response Times Across Clusters:

Response times vary significantly across different clusters, with some regions showing higher total response times.

Cluster 4 (highlighted yellow) has incidents with **higher predicted response times** (~78.62 minutes).

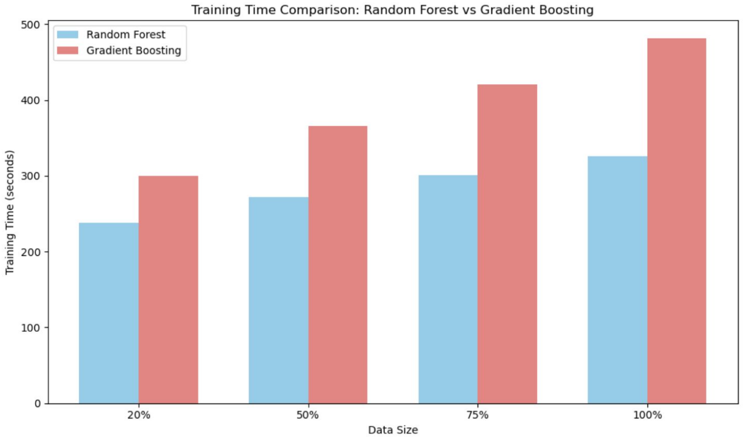
Cluster 0 (purple regions) shows **lower response times**, such as **10.33–24.32 minutes**.

High response times in clusters like **4** suggest areas where emergency response resources may need to be prioritized to reduce delays.

Scale In

Training Time Increases with Data Size:

- As the data size increases from **20% to 100%**, both models show a **linear increase** in training time.
- Random Forest (RF): Increases from **237s** to **325s**.
- Gradient Boosting (GBT): Increases from **299s** to **481s**.



Data Size	RF Training Time	RF RMSE	RF R^2	GBT Training Time	GBT RMSE	GBT R^2
20%	237.972965	3.643566	0.976288	299.883065	1.616928	0.995330
50%	271.645378	3.570368	0.977129	365.595535	1.642350	0.995161
75%	300.540333	3.567099	0.977091	420.709476	1.632526	0.995202
100%	325.801209	3.649114	0.976130	481.002217	1.616351	0.995317

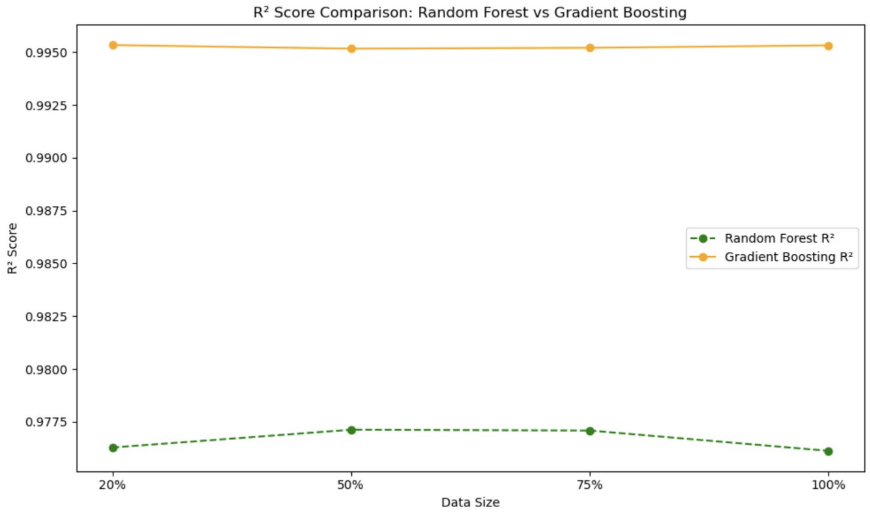
Scale In

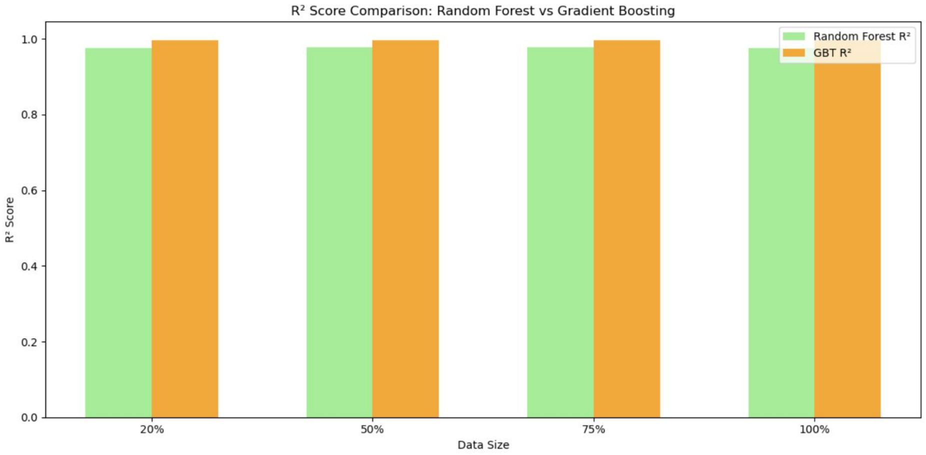
1. Gradient Boosting Consistently Outperforms Random Forest:

- Across all data sizes (20%, 50%, 75%, and 100%), **Gradient Boosting (GBT)** achieves higher **R^2 scores (~0.995)** compared to Random Forest.
- This indicates that GBT explains a larger proportion of variance in the response variable, making it the more accurate model.

2. Random Forest Shows Slight Fluctuations:

- Random Forest R^2 remains stable around **0.976–0.977** but shows minor variations as the data size increases.
- For instance:
  - **50% Data Size:** R^2 improves slightly.
  - **100% Data Size:** R^2 drops slightly, indicating a potential limitation in capturing complex patterns at larger scales.



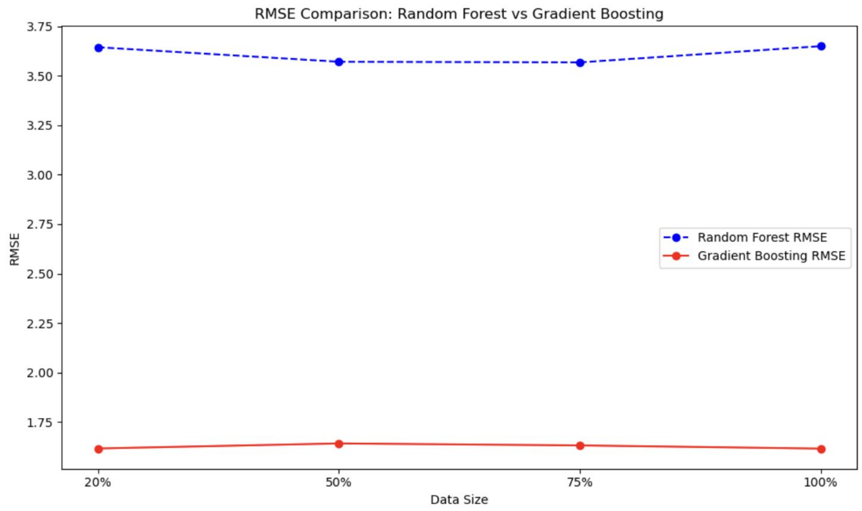


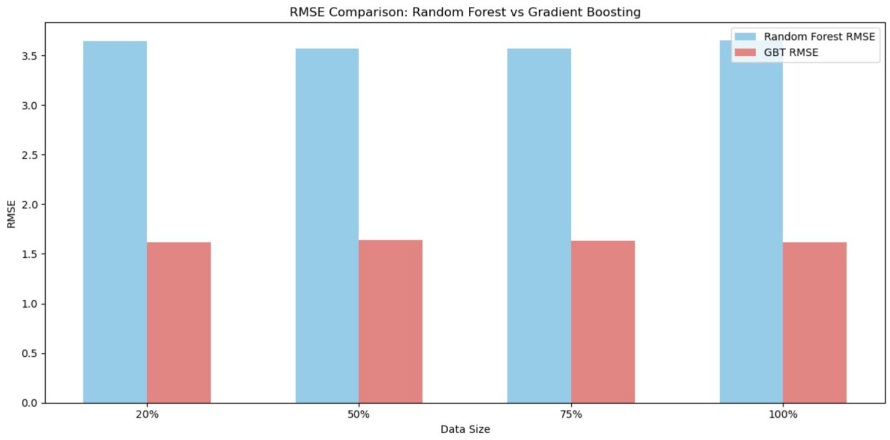
Gradient Boosting Shows Lower RMSE:

- **Gradient Boosting Trees (GBT)** consistently achieves a **lower RMSE (~1.6)** compared to Random Forest (**~3.5**).
- This demonstrates that GBT provides more **accurate predictions** with lower errors across all data sizes.

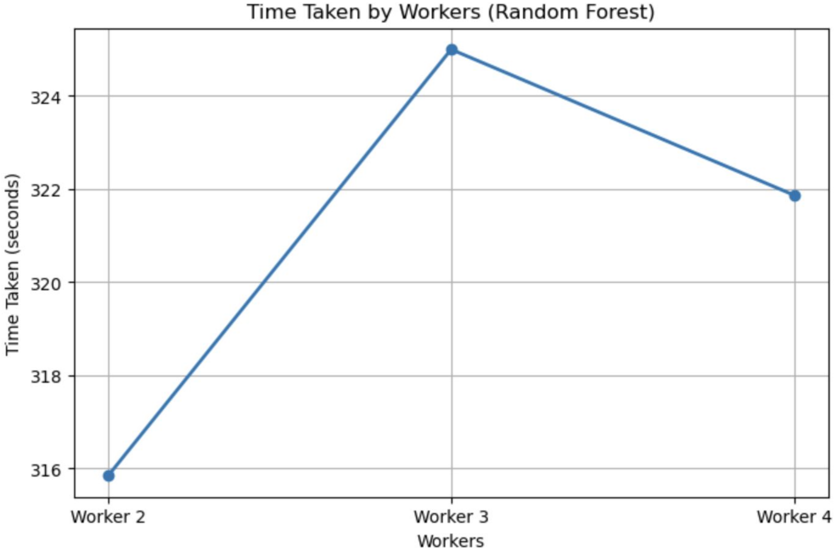
2. Random Forest RMSE is Higher and Stable:

- Random Forest RMSE hovers around **3.5**, showing minimal variation as data size increases.
- While stable, the higher RMSE indicates that RF struggles to match the accuracy of GBT.

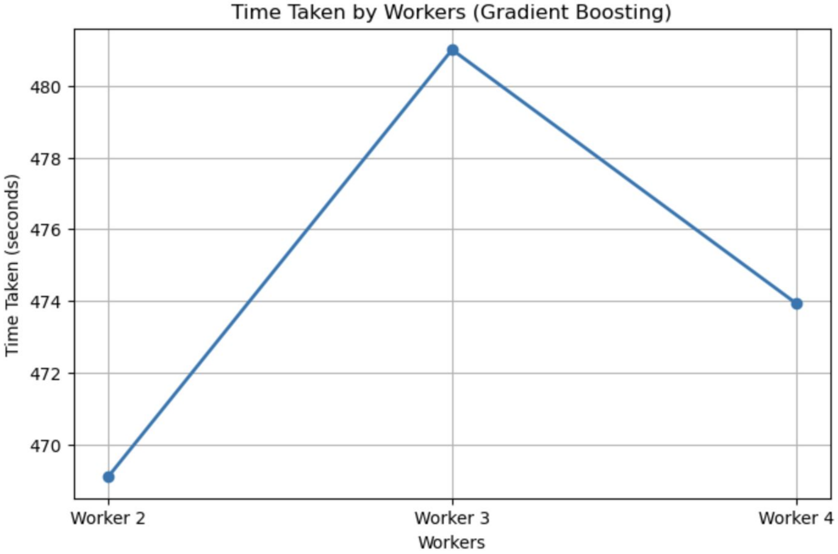




Scale out



Time Taken by Workers (Random Forest)
Time Taken (seconds)
316
318
320
322
324
Worker 2
Worker 3
Worker 4
Workers



Time Taken by Workers (Gradient Boosting)
Time Taken (seconds)
470
472
474
476
478
480
Worker 2
Worker 3
Worker 4
Workers

Scale out

- Worker 2 shows the lowest execution time due to **lower communication overhead** and optimal resource utilization.
- Adding more workers (3 and 4) introduces **task coordination delays** and **data shuffling overhead**, slowing execution.
- Uneven task distribution across workers can cause some nodes to finish early while others lag.
- Scaling out beyond Worker 2 does not linearly improve performance, highlighting the need for **load balancing** and cluster-level optimization.

Worker Nodes	RF RMSE	RF R^2	RF Time (s)	GB RMSE	GB R^2	GB Time (s)
2	3.6491	0.9761	315.848	1.616	0.9953	469.107
3	3.6491	0.9761	325.000	1.616	0.9900	481.000
4	3.6491	0.9761	327.450	1.616	0.9953	473.940

Thank You!