Project Final Technical Report Outline

Cross Zeigler, Suchada Hapikul, Ewin Hong, Neethu Battula, Sagar Deepakgiri Goswami

Note: This is a skeletal form for the final technical report meant to outline ideas for the document, **not the fully fleshed out final technical report itself**

Note: The reused material is highlighted in orange. It may be necessary to update this material to match with the **current version** of the project. It may also be necessary to **delete** or **move** certain material into different sections depending on its current usage.

Cover

Title

Analyzing and Predicting Traffic Speeds for New York City

# Abstract

<Content from previous deliverables.>

New York City (NYC) is notorious for having heavy traffic at nearly all times of day, heavily impeding motor traffic in the city even at the best of times. It often leads to congestion that can delay travel and cause serious issues ranging from inflated costs of living to high gasoline prices. Therefore, it is very important for any changes in infrastructure to lead to more pedestrian-friendly roads and help people access public transportation easier, leading to less cars on the road over time. This project aims to study one or more datasets of traffic patterns of NYC and to create a model that predicts when traffic occurs the most hourly, daily, weekly, monthly, and annually. The project studies this phenomenon by utilizing datasets from NYC open data to analyze traffic in the city, discovering commute patterns in transportation to resolve congestion amongst other traffic issues within NYC. Various analytical methods like K-Means Clustering, Time-Series Analysis, Multivariate Regression Models, and Decision Trees were considered to explore the data, and generate crucial insights that might fulfill the aforementioned goals.

When performing the K-Means Clustering, it was observed that some of the sensors/link points behaved similarly in terms of daily average traffic speed. Two such clustering models were created that individually examined the average traffic speed, as well as the delta values for change in the average traffic speeds. On the other hand, the time-series analysis rendered different kinds of insights which focused on the variation/patterns in the average traffic speed grouped by hours of the day, days of the week, and months of the year. The data for 2018, 2019, and 2021 behaved very similarly, and mostly overlapped with each other when plotted on graphs for the Time-Series Exploration. The 2020 data showed similar traits, except it exhibited slightly higher traffic speeds compared to other years.(Reused from [AIT614 Team 1 Final Technical Report](https://docs.google.com/document/d/1iqD_PWN11449W1ypqMIoxRbqcHXi55wlHK09NFPkTSA/edit))

# Introduction

<Content from previous deliverables explained in finest level of granularity and detail.>

New York City (NYC) is notorious for having heavy traffic at nearly all times of day, heavily impeding motor traffic in the city even at the best of times. This often leads to congestion which impacts travel and causes delays, wastes gas, and can leave people feeling dissatisfied with personal and public transportation. A prominent question for NYC, then, is how to improve the traffic infrastructure within NYC, leading to changes and improvements in transportation, pollution levels, and more. The main goal of this project was to study the datasets of traffic patterns of NYC and to create a model that predicted when traffic occurs the most hourly, daily, weekly, monthly, and annually. The thought was that this could help to consolidate which times of the year are “problem” times and allow people to adjust their commute accordingly before major infrastructure changes are made. Another thought that we had for this project is that by doing this we may even help change traffic patterns based on people’s ideal personal schedule, allowing someone to leave right on time to make traffic as smooth as possible for them and others. It became more important than ever this year to conserve gas not just for the environment but for personal use too as the cost of living, including gasoline prices, rises.

The main motivation for completing these goals is to give insights into the NYC traffic system to give suggestions to improve it. Improving everyone’s personal schedule is a small stepping stone to accomplish this goal, but that is useless and a short-term solution without improving the actual infrastructure of NYC. These changes could lead to more pedestrian-friendly roads and easier access to public transport, reducing the number of cars on the road over time. To accomplish this, the experimenters attempted to look over the specific areas of congestion along with the times there is little to high congestion. The thought was that by looking over the areas of high congestion they could find a way to reroute traffic to make those areas less busy. Looking over times of congestion helped us to see how traffic changed over time and allowed us to theorize why these changes occurred due to holidays, construction, or other events. The end goal was to advocate for public policy to refine infrastructure.

Several audiences could benefit from this analysis. The primary audiences are the stakeholders in the urban transportation industry and transportation agencies. For both, if people have an easier time moving around, they may be more inclined to utilize public transportation, which could drive revenue up. The latter in particular may like this, as they have a lot more to gain directly from people using their services more.

Two other audiences are emergency services and NYC residents. Ambulances and firetrucks need to get to their destinations quickly to maximize the amount of lives they save and the chances of saving said lives. With NYC as it is, even if people try to maneuver out of their way, they still have to deal with traffic regularly that could impede their progress. NYC residents would not just appreciate this ease of access emergency services have to save their lives, but also in day-to-day life. Someone may struggle to get to work on time or meet up with friends if they constantly have to battle heavy traffic. Changing infrastructure could help improve satisfaction with living in NYC, improve work ethic and punctuality, and lead to increased happiness with one’s social life by ease of access alone.

Attempting to improve NYC’s traffic situation is not new. People have studied this issue extensively in hopes of understanding *why* traffic is so bad in the city and how to improve it over time. Nibareke and Laassiri (2020) used various machine learning models to model traffic flow over time to predict traffic effectively. While they specifically dealt with air traffic their model allowed people to see how accurately one could predict delays and traffic in transportation with the correct model. Vasudevan (2016) presented a technical approach that combined Apache Spark’s open-source data analytics and machine learning techniques to predict traffic flow patterns using simulated connected vehicle messages. The study reported that connected vehicle data can be processed rapidly using Big Data analytics to generate precise predictions of traffic flow regimes. Other researchers reviewed had similar results. (Reused from the [Project Progress Report](https://docs.google.com/document/u/0/d/1XMK9vkf_mwko-5k4U43iwo9B1tKerVg4tF_OiH-BFIM/edit))

# Goals

<Project Goals>

<Long Term Future Goals>

The primary goal of this project is to create a machine learning model to predict the traffic within NYC based on the NYC traffic statistics collected from the New York City Department of Transportation website. Looking over times and the specific areas of congestion can assist audiences to know how traffic changed over time. This project aims to provide valuable insights into the NYC traffic system to improve transportation infrastructure for increased satisfaction, safety and decreased travel time. Providing audiences with the times and areas with the greatest amount of congestion may allow people to leave their locations at the optimal times to make traffic as smooth as possible for them. Thus, they can improve their own commuting schedule and potentially increase their own safety. Improving personal schedules is a small stepping stone to accomplish reforming traffic infrastructure within NYC for the long term.

The long term goal of this project is to improve the infrastructure of NYC. While the short-term goal can help individuals and perhaps even large collectives of people, it may only support so many people for so long. During the short-term goal’s tenure, NYC’s departments for transportation and infrastructure could make plans to fix and reform the public transportation and infrastructure of the city. That way, commuting schedules amongst other problems would not fall completely on individuals and instead cater to them for consistently reliable transportation. This project intends to improve commuting schedules in the short-term by improving personal schedules, but the ultimate goal is to affect policy for the improvement of NYC’s infrastructure. (Reused from the [Project Proposal](https://docs.google.com/document/u/0/d/1h3MHoQYYvw7Q7lPh0WYnJcs2fMpxShezMNUoayZOXDw/edit))

* Create machine learning models based on previous NYC traffic data
  + Sourced from New York City Department of Transportation
  + Determine how traffic works within NYC
* Provide insight into traffic system for improved transportation
  + Times & areas with greatest congestion
  + Improve commuting schedules in the short-term
  + Overhaul transportation infrastructure for the long term

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))

# Requirements

The project has several functional requirements. The requirements for analysis include several tools from programming languages to visualization tools. This project uses R Studio for extracting, processing, cleaning, and exporting data. R is easy to use and provides a tool with a good amount of complexity combined with a low entry bar to help with our initial analysis. Thus, R is utilized for both data exploration and deeper data analysis as well due to it affording us the proper tools to do so. Moreover, R also provides clear graphs and outputs that help to create a data story for projects. In addition to this, the earth library was used to run a more complex MARS analysis on the data.

The second main analysis tool we used is Python. The project used Python for data cleaning and analysis with specific libraries such as Pandas. Python combines many R’s strengths from good visualizations for data stories, a great breadth of data analysis tools, and a low bar for entry, especially if one is familiar with Java or C. The language has many libraries that assist with these tasks, including the aforementioned Pandas which helps with extracting, processing, and cleaning data. Alongside R, Python was used to store the data initially in order to do rudimentary exploration so that one could familiarize themself with what the data may mean. Python’s libraries such as prophet gave access to more complex analysis such as the time series analysis, pushing the project further along.

The main platforms used for this project are Databricks, Apache Spark, GitHub and Microsoft Teams. Databricks provided many of the languages and other tools necessary to perform the analysis, and served as a better storage tool for the data in the long-term relative to other platforms. Databricks DBFS specifically was used as a NoSQL database to handle the large volume of data from the original dataset. Databricks also served as a decent collaboration tool for multiple people to modify the code together in real time, circumventing any issues of having to meet in-person for the project.

Apache Spark and its corresponding libraries (i.e. Spark MLlib) were used for most of the more complex data analysis, such as principal component analysis (PCA) and k-means clustering. Spark also provided access to SQL and R on Databricks for visualization and the use of more complex libraries relative to other platforms. These languages gave quick insights into the data with simple queries that could be visualized easily.

The primary platform for communication was Microsoft Teams. Teams worked as a central hub for most meetings and information exchange. Even if a person were to miss a meeting, it was easy and quick to update them on the current status of the project by posting a summary of what was accomplished on Teams right after the meeting. Teams also held a number of the files for the project, including coding files, documentation on the code, and reports on the project. This way, people had unhindered access to the files they may need for analysis or directions.

Google Docs was used in tandem with Teams, as it provided an easy way to share and create files for reports as well. In addition to that, Docs provided a running “meeting” file that recorded what went on in each meeting. A Google Drive was eventually installed that held every report necessary for the project and was accessible to every group member.

GitHub was the last main collaborative tool, and was used alongside Databricks for ease of code access and editing. Everyone had access to the original repository that hosted the code and could push for changes from a cloned repository. GitHub served as an excellent platform to track the coding progress of the project on, and could be used to control which version of the project came to fruition better than Databricks could in many cases. It also served as the home of the documentation on how to run the project.

(Reused from the [Project Proposal](https://docs.google.com/document/u/0/d/1h3MHoQYYvw7Q7lPh0WYnJcs2fMpxShezMNUoayZOXDw/edit))

* Python / R
  + Extracting, processing, cleaning, exporting data
  + Pandas, Prophet, etc.
  + Initial storage and exporting for data
  + earth library (MARS) - R
* Databricks / Spark
  + Data Exploration using Spark SQL
  + Spark MLLib - k-means, PCA
  + Databricks DBFS
* GitHub (Version Control), Microsoft Teams (Communication), Google Drive (Shared File Storage)

Preliminary exploration was done in python but due to computational limitations Spark/Databricks provided a better data exploration process using Spark SQL.

Later on, we used the Facebook Prophet python library to perform the time series analysis.

In addition to Facebook Prophet, we used Spark MLLib kmeans and principal component analysis

We used tableau as a map visualization for some geospatial analysis

Additional, the team used various collaboration tools for code sharing and file sharing (github/onedrive/google drive) and teams communication

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))

# The Datasets

* Selection
* Description
* RDB schema or NoSQL Data Model

<data source>

<data format>

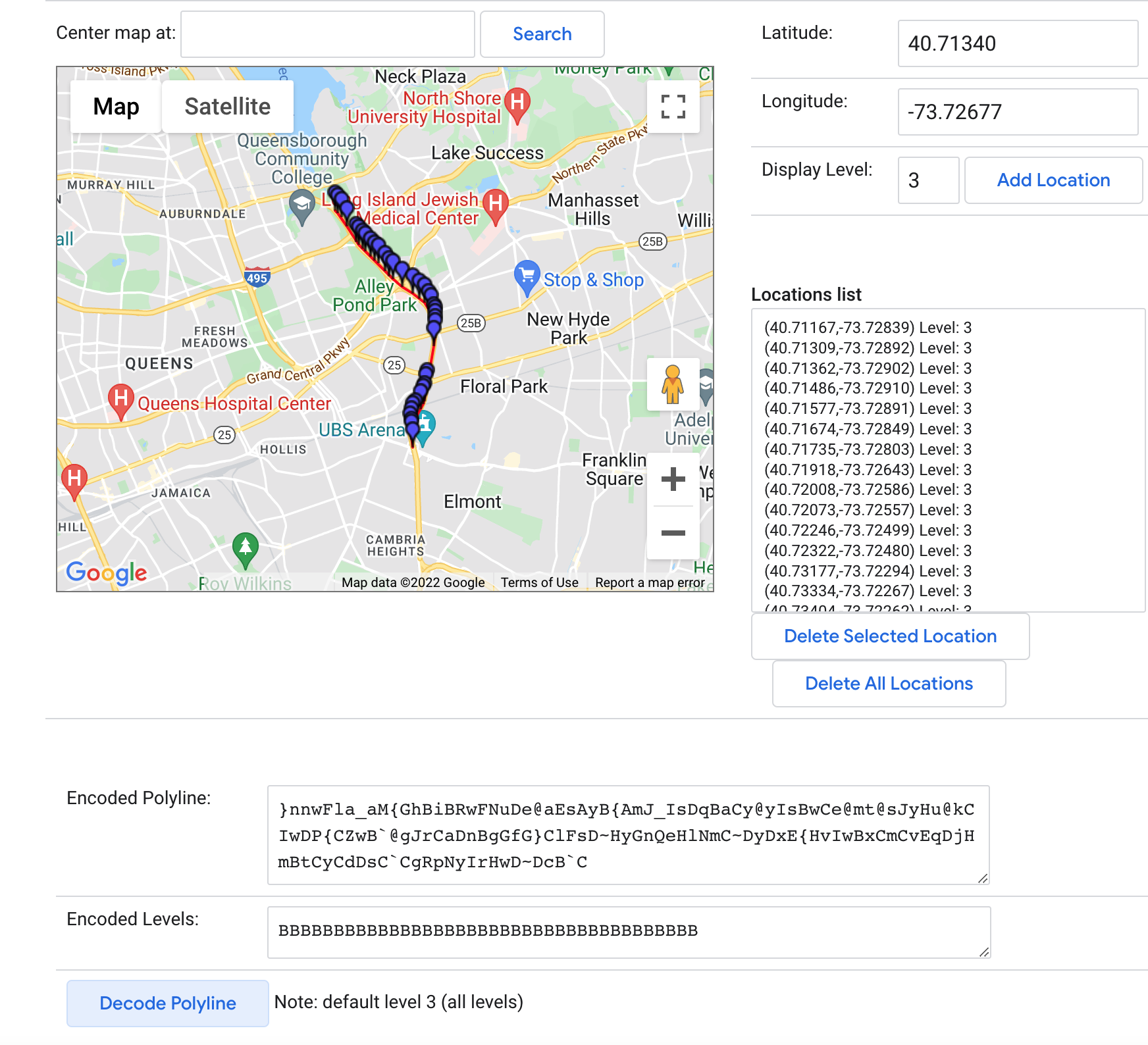
The dataset is collected from the New York City real time traffic speed found at: <https://data.cityofnewyork.us/Transportation/DOT-Traffic-Speeds-NBE/i4gi-tjb9> (The NYC Department of Transportation website). The dataset is gathered in. As of March 9th, 2022, the dataset has 58.8 million records starting on April 17, 2017. The dataset is continuously updated with real time data being provided by the traffic sensors. There are 13 features listed found as table 1 in Appendix. With these 13 features, 8 features are useful for our research. These eight features are id, speed, travel time, data as of, link points, owner, borough, and link name.

Speed in this dataset is the average speed between all of the link points. Travel time is time spent through the link points. The link points are a group of latitude and longitude points. The link point can be used to calculate the distance of the starting and ending points of a single link. Additionally, distance could be calculated by using speed and travel time as means of validating the calculated distance by link points. These link points can be evaluated over time and their evolution. Then with the link points, the Boroughs can be added for grouping for additional analysis. Each link point has at least 2 points and could go to 10 points. Due to the dataset, there is some evidence there might be more than 10, but are cut off due to the original dataset’s database type constraint of 256 characters. (Reused from the [Project Proposal](https://docs.google.com/document/u/0/d/1h3MHoQYYvw7Q7lPh0WYnJcs2fMpxShezMNUoayZOXDw/edit))

**Appendix**

**Table 1 - Dataset Features, Description, and Data Type**

| Name of Feature | Description | Data Type |
| --- | --- | --- |
| ID | Unique Identifier for Sensor within dataset | Integer |
| Speed | Average Speed traveled between the link points origin and destination | Double |
| TravelTime | Time Travel in seconds | Integer |
| Status | Artifact (not useful) | Integer |
| Data\_As\_Of | Date and Time of Day for Sensor Data | Datetime |
| Link\_Id | TRANSCOM Link ID | Integer |
| Link\_Points | Group of Latitude and Longitude points of Sensor data | List of 2 double (latitude and longitude) points |
| Encoded\_Poly\_Line | Link\_Point representation of Google compatible poly line | String |
| ENCODED\_POLY\_LINE\_LVLS | Encoded representation of Poly Level | String |
| Owner | Owner of Sensor | String |
| TRANSCOM\_ID | Artifact (not useful) | String |
| BOROUGH | Name of Borough Sensor exists | String |
| Link\_Name | Description of Sensor location | String |



(Reused from the [Project Proposal](https://docs.google.com/document/u/0/d/1h3MHoQYYvw7Q7lPh0WYnJcs2fMpxShezMNUoayZOXDw/edit))

# The System

* Architecture
  + Draw an overview diagram of the system architecture or framework
  + Describe each module in detail
* Data Pre-Processing
  + Describe the steps and methods for data pre-processing in detail
* Data Analysis Methods
  + Describe the major data analytics algorithm(s).
    - Existing Method(s)/algorithm(s)
      * Briefly describe the existing algorithm(s) you used and explain why
        + Data visualization
        + Data analytics
      * Clearly explain how to use these algorithms for your data analysis
    - New Method(s)/algorithm(s) developed by your own
      * Clearly describe the algorithm(s) in detail
      * Clearly explain why and how to use these algorithms for your data analysis
  + Briefly describe other algorithms you used in the system
* Software and hardware development platforms
* Importing the Data
  + (CSV from the website)
  + Converting to parquet
* Data Preparation
  + Data Cleaning
    - Filtering Data between 2018 and 2021
    - Removing data for Link Points that have comparatively fewer records

1. Data Preparation
   1. Data Ingestion and Cleaning  
      24 GB worth of CSV data is fetched from the NYC Department of Transportation open data portal. The file is imported as a data frame inside a python module running on the Databricks portal. This project executed the wrangling steps in order to receive appropriate data in the form of a tidy format. Therefore, all null values are removed.
   2. Data Exploration  
      Basic data analysis is carried out to understand the scope of advanced analytics on the data. Locations with lower average traffic speeds will be identified to concentrate on the most affected regions. Additionally, basic visualizations are performed to understand the data distribution and range better.
2. Data Analytics
   1. Linear Regression Models to predict Congestion  
      Machine learning models are trained to perform linear regression and predict the frequency, interval, severity, and location of the congestion for a given time. Multiple Regression models are trained to predict the congestion a few hours, days, weeks, months, and years ahead. These models can provide pivotal insights to understand the nature and seasonality of occurrence for congestion. Further, GWR (geographically weighted regression) is performed to train models and make predictions based on latitude and longitude information.
   2. Random Forest / Feature Extraction(Not done)  
      To study the effect of congestion at one place on neighboring localities, the Random Forest technique can be used to identify locations that are potential bottlenecks or chokepoints. A dependency map can be created based on the most important features.
   3. Clustering locations and sensors  
      Clustering performed on the data points to group the traffic speed sensors with similar proportionate delta values for change in average traffic speeds. Such clusters can provide insights regarding the directional flow of traffic for a given time and location. Features like Average Traffic Speed and Congestion Duration are used for clustering.
3. Predictions Dashboard (Didn’t do, change of plans)
   1. Back-end Scheduled Script  
      A back-end script will run on a schedule to fetch the data from the department of transportation website, run machine learning models based on the new data, and generate new predictions. The coefficients for the new model will be logged and stored on a file. Even Near-Real time Predictions can be made using Spark, instead of scheduled batch processing.
   2. Front-end Predictions Dashboard (Didn’t do, change of plans)  
      A dashboard will be created that predicts congestion based on the latest model and data. Various predictions will be made to predict congestion in a few hours, days, weeks, and months in the future.
4. Case Studies (Didn’t do, change of plans)
   1. Case study on localities with the most congestion  
      Locations with the highest congestion will be identified in the Linear Regression stage. Literature Review will be carried out for these locations to validate the findings and understand the independent factors that cause congestion.
   2. Case studies on localities with the highest dependency for congestion in neighboring localities  
      Locations that have the highest dependency for congestion in neighboring localities are identified in the Random Forest stage to understand how a particular location causes congestion in surrounding locations. Again, a literature review is performed to validate the findings and outcome of using this method and determine the factors that cause such situations.

**Proposed Development Platforms**

Hardware/OS By Person:

* Cross: Mac 8GB RAM, 3.1 GHz, Dual-Core Intel Core i-5
* Ewin: Mac 6core CPU at 3.2GHZ Intel i7 64GB of ram
* Fon: Mac M1 chip with 8-Core CPU and 7- Core GPU
* Neethu: Mac 16GB RAM, 2GHZ,Quad-Core Intel Core i5
* Sagar: Windows Machine: 16 GB RAM, AMD Ryzen 4800H 8-cores 16-threads, Nvidia RTX 3060 Mobile GPU

Platforms & Software:

* Databricks (Apache Spark)
* Python, version 2.7 or above
* R, version 3.0 or above

In the finalized project, the systems framework consisted of several modules/tools used to complete the project. The data was first gathered from the source and turned into a CSV file around 25 GB in size. To make the data easier to process, this CSV file was converted into parquet files that anyone experimenting with the project could feed into the algorithms to run to avoid storage and performance problems the CSV may have provided. Most of this was done on a local machine due to the power said machines provided relative to other platforms.

After this occurred, the data was stored on the Databricks DBFS. Here, the data was cleaned processed. Most of the code to clean the data was done in Python, with a focus on removing data that was extraneous and useless. To measure how useful the data was, the average speed was calculated (i.e. 38 mph), and no speed more than two standard deviations (15 mph for one standard deviation) going above this speed were kept. The range of kept speeds became 0 to 68. There was some debate on whether to keep all speeds that were 0 mph based on this criteria as well, but ultimately nothing was found using a similar criteria to eliminate data points with a speed of 0 specifically. After this, the link IDs were grouped together and the number of times each ID appeared was counted. If any IDS had less than 45,000 records, they were removed due to not contributing too much to the analysis and providing little to no interesting insights. Additionally, the year 2017 was omitted because the data returned results not reflective of the actual data, suggesting an error in how the data was originally stored.

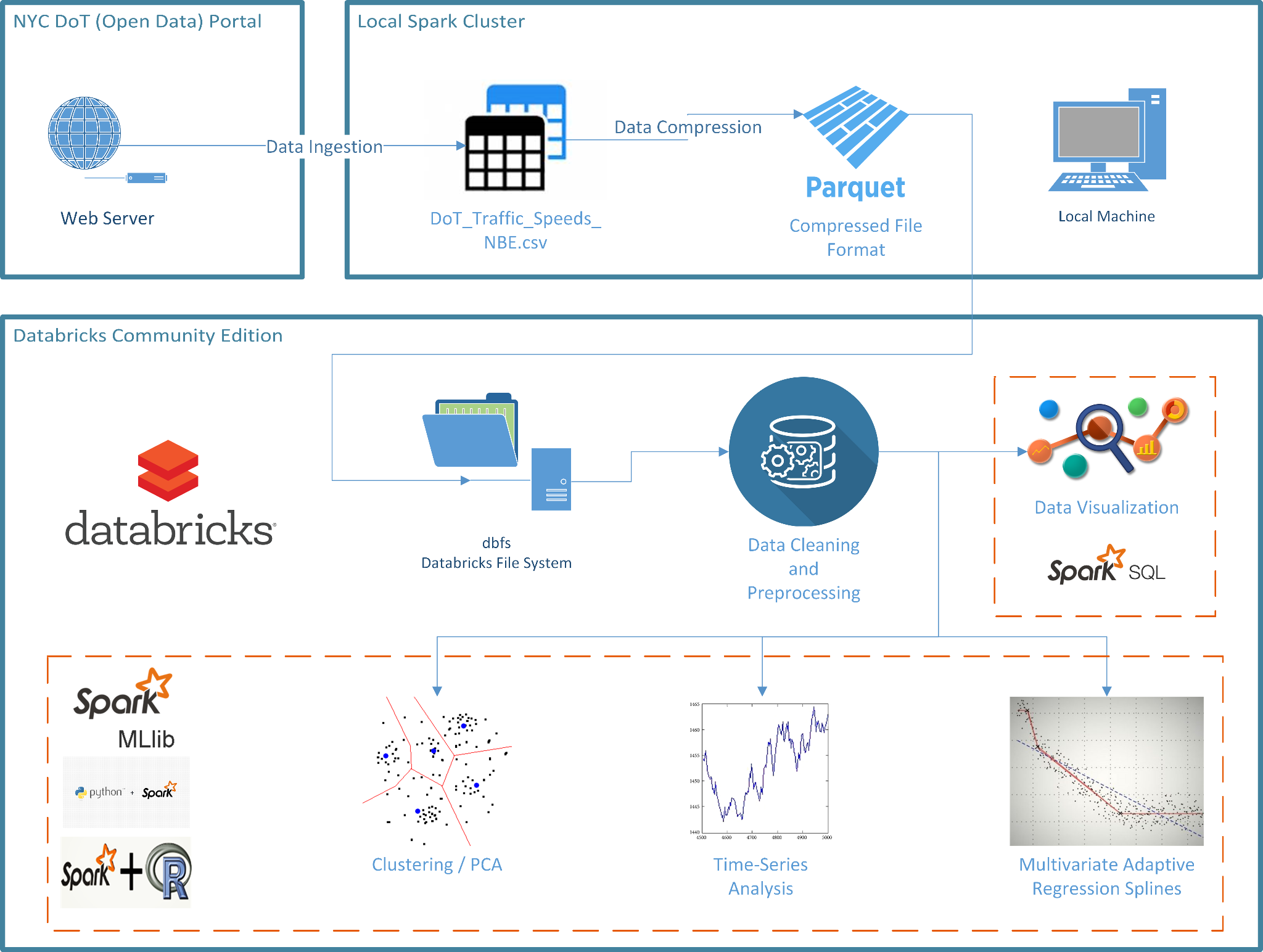
The first major analysis utilized was K-means clustering to determine which link IDs had similar speeds and potentially why. PySpark was used for this, and the Spark MLlib library was utilized specifically for clustering. The 24 features from the cleaned dataset were fed to the k-means and PCA algorithm to determine the potential number of clusters. After this, a clustering evaluation was performed with Spark MLlib to determine the optimal number of clusters from 2 to 10, with the algorithm providing an answer of 3. After this, PCA was performed on the standardized features of the data. The three clusters were mainly divided by their speeds and the ranges of said speeds and locations, and each cluster's link IDs were graphed on a box and whisker chart to determine their ranges of speed. The cluster having the lowest speed typically were located near bridges and had the lowest range of values for each link ID while the cluster with the highest speeds had the largest range of values for each link ID.

The next major analysis performed was a time series analysis to determine traffic trends for the past several years and to predict trends for the next two (up until 2024). To do this, Python’s FBProphet library was installed and fed two values: the datetime and the speed. While link IDs were grouped together in this instance, most of the visualizations featured were for an overall, general trend on a daily to yearly basis. The FBProphet method then used the speeds from 2018 to 2022 to find a trend line for viewers to follow, establishing how speed changed over time and even returning delta values for this change in speed over specific durations. Afterwards, the algorithm predicted what the speed trend would be for the rest of 2022 to 2024 based on the trend established from 2018 to 2022. This data provided an approximate view of congestion, and when grouped by link IDs, could give an idea of which areas were most congested (low speeds) and least congested (high speeds) overall and at certain times of the year.

There was an attempt to perform two other types of analysis, those being decision trees and Multi-variate Adaptive Regression Spline (MARS). Both were performed in PySpark with MLlib as well, but several issues prevented any fruitful analysis from either. Even on a sample of the data (100,000 records vs. tens of millions of records), the algorithm took too much time, sometimes hours even. One reason was that Fitting splines on top of linear functions increased the time of performance. Another issue for MARS in particular was a lack of libraries to properly visualize the data, as modern libraries did not have the capabilities most of the time and older libraries often lacked this functionality altogether. Finally, the time series analysis provided better results for a comparatively shorter period of time and lower resources overall, pushing time series analysis as the better tool.

Beyond these algorithms, very few were used for the data analysis. SQL was used in some parts for an initial analysis and to display and even clean the data, but most of that utility came from R and Python in Databricks.

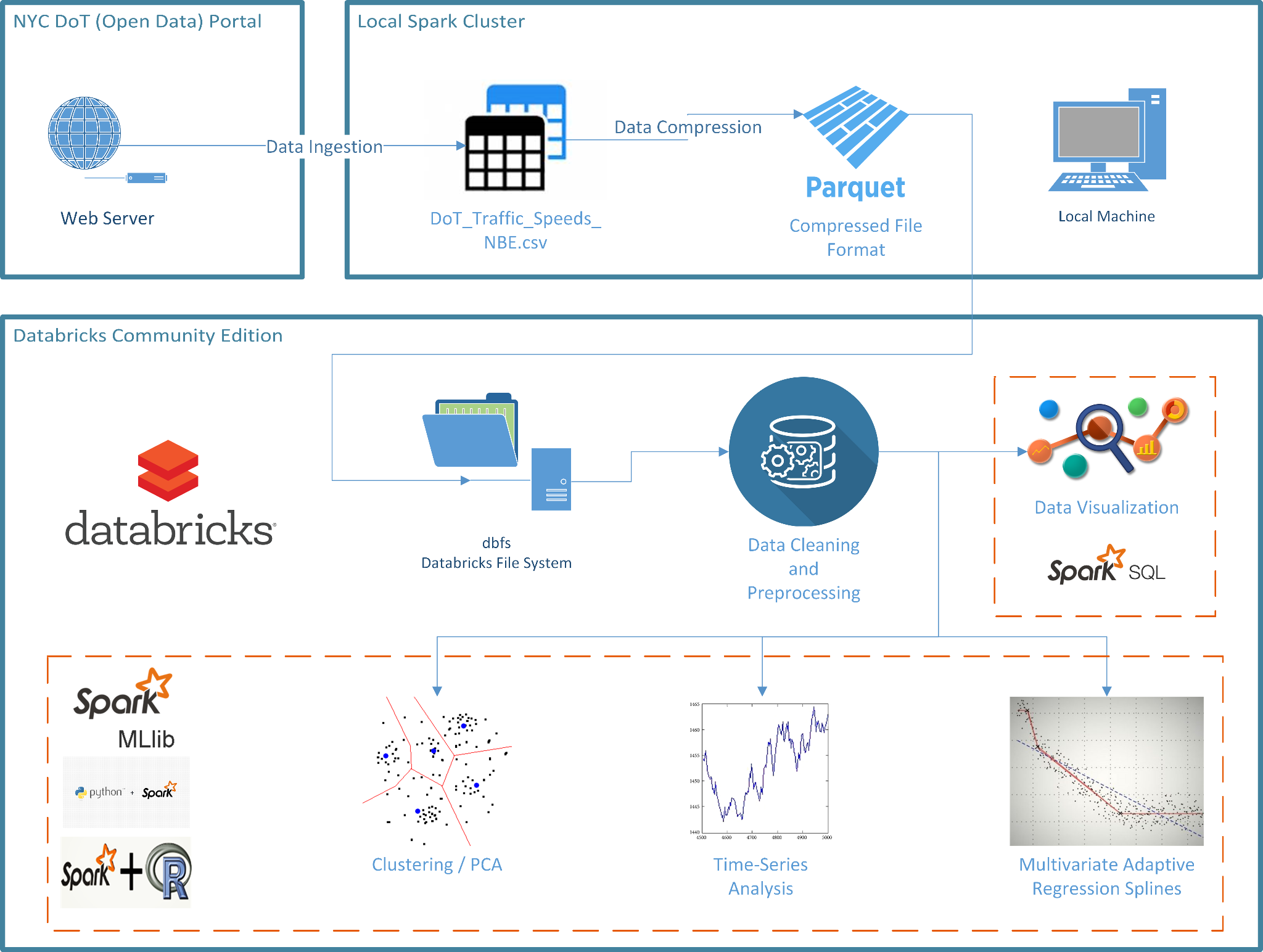
For hardware, there was a mixture of computers and OSs running the algorithms. Several people utilized Macs with at least 8 GB of RAM, with one even having 64 GB of RAM. One user had a Windows machine that had 16 GB of RAM. Most of the OSs running on each machine were up to date as well to ensure quality performance and to be compatible with the most modern software available. For software, most versions of the programs used were the most current ones or near the most modern ones, with most users utilizing Python version 2.7 or above and R version 3.0 and above.



(Reused from the [Project Proposal](https://docs.google.com/document/u/0/d/1h3MHoQYYvw7Q7lPh0WYnJcs2fMpxShezMNUoayZOXDw/edit))

* Databricks
  + Collaborative platform to run models on
* Apache Spark
  + Used within collaborative Databricks to make predictive models
* Spark MLlib
  + Machine learning library with Apache Spark used for:
    - Clustering
    - Regression

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))



# Experimental Results & Analysis

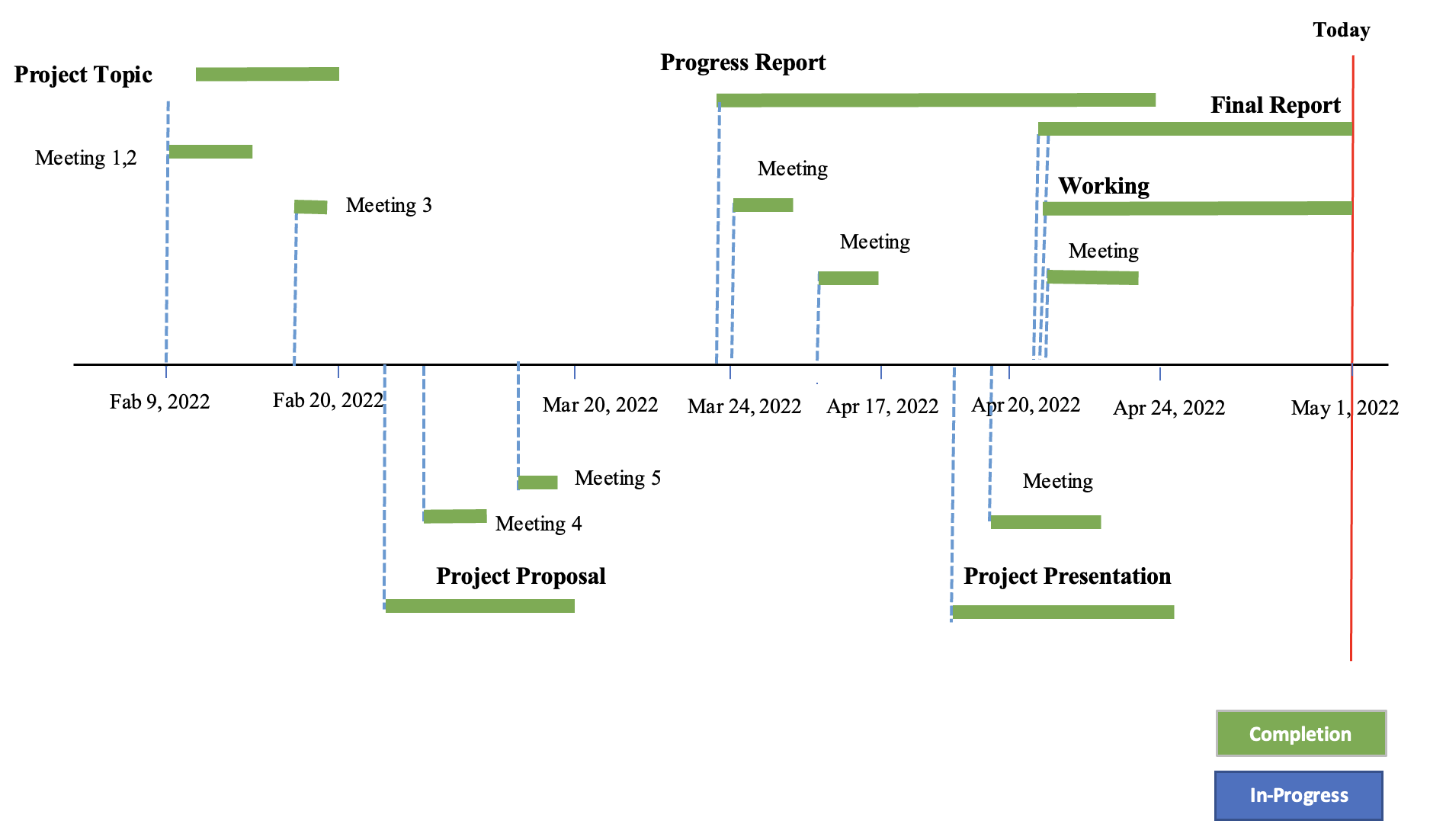
* Explore and present analysis of the dataset using relevant tools
* Prepare relevant analysis and visualizations for selected data items
* Interpret the results

**Analysis Overview/Intro**

* Techniques we started with:
  + Started with clustering and Linear Regression
  + Linear Regression too simple for project complexity, tried MARS Analysis
  + Other Attempted Techniques:
    - Decision Trees, Association Rule Mining and Time-Series
* Final Outcome:
  + Clustering and Time-Series performed exceptionally well for the data we have
  + Not proper libraries available for Decision Trees and MARS
  + MARS was not scalable to the data, took too much of time to process

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))

# Summarized Final Project Timeline



The project took place over the course of three months, starting in early February with the formation of the initial group up until the start of May with the final deliverable’s submission. As observers can see in the timeline above, every individual task is marked with a pin that has a tail with the assignment’s status: either being “in-progress” (blue) or “completed” (green). Additionally, there was a third category named “duration” colored gray, referring to tasks that hadn’t been started yet.

Each pin’s tail extends for the rough duration of the project its respective task occurred under, e.g. the progress report was planned to start around March 24th, 2022, and was worked on up until its due date, April 24th. At the time of writing this, the final report and working system is being finalized, and should be complete by the due date, making them green upon turn in.

# Conclusions

* Draw conclusions for the overall project
* Lessons learned

**Analysis/Conclusion**

* Clustering analysis provided insight into high congestion/low speed areas
  + 14 link points had a lower than average speed
    - Higher vehicle usage and low traffic flow due to bridges/toll
    - Being near/in the Manhattan borough
* During the year 2020, the average speed was faster
  + Likely due to COVID leading to less travel in general

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))

**Lessons Learned**

This project has gained understanding significantly about **how to work collaboratively** with Databricks as well as the technology in the BigData world.

There are the important things for executing group project which are

**Coordinating** and **dividing** responsibilities. such as dividing based on skill set.

**Managing a schedule** and working towards realistic goals

**Knowing what analysis** will be useful to our goals

**Identifying traffic patterns** with a few attributes. To exemplify this, speed is a good example to identify traffic patterns. While low speed usually correlated with high congestion, high speed meant less traffic.

**Using complementary sources** to find their causes. For example, collecting data from related articles and news reports that discussed traffic situations during the COVID-19 pandemic.

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))

**Future Considerations**

* Multiple datasets combined could present a better picture to traffic conditions like speed or travel time.
* Geospatial analysis for nearby congestion to recommend a path with a start and destination in mind
* Additional consideration of features
  + Using an average lag or lead for k-means
  + # of lanes
  + Foot (pedestrian) traffic
  + Flow of traffic
    - One way or Bidirectional
    - Cardinal Direction

Traffic has multiple variables that contribute to the flow of traffic and landscape factors like bridges and tolls. These are physical components that are measurable and predictable. But there are unforeseen circumstances like accidents and road maintenance that this study did not consider due to time constraints and lack of experience. Combining traffic conditions to accidents and road maintenance may help gain a better insight into the NYC traffic issue.

Performing geospatial analysis of the nearby area can influence the flow of traffic at the start or along the route. Consider an analogy of a pipe where water can flow through a pipe quickly if flow is not impeded.

Using geospatial data as additional features into a recommendation model with a starting point and a destination

Some consideration could be used as additional features like number of lanes, if a road has foot traffic, if a road is a one way or bidirectional, and cardinal direction.

(Reused from the [Final Presentation](https://docs.google.com/presentation/u/0/d/1DH5gvKwUV_76aAYBfdH4uq0TuDkombtygrEFJxJX1Pw/edit))

# References

* Provide appropriate citations and references
* Be sure to include a citation and link(s) for the dataset(s)
* see http://infoguides.gmu.edu/citingdata

Works Cited

Chen, H., Rakha, H. A., & Sadek, S. (2011). Real-time freeway traffic state prediction: A particle filter approach. *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. https://doi.org/10.1109/itsc.2011.6082873

City of New York Department of Transportation. Retrieved on March 9, 2022 from:

https://data.cityofnewyork.us/Transportation/DOT-Traffic-Speeds-NBE/i4gi-tjb9

Hofleitner, A., Herring, R., & Bayen, A. (2012). Arterial travel time forecast with streaming data: A hybrid approach of flow modeling and machine learning. *Transportation Research Part B: Methodological*, *46*(9), 1097–1122. https://doi.org/10.1016/j.trb.2012.03.006

Kong, F., Li, J., Jiang, B., Zhang, T., & Song, H. (2018). Big data‐driven machine learning‐enabled traffic flow prediction. *Transactions on Emerging Telecommunications Technologies*, *30*(9). https://doi.org/10.1002/ett.3482

Nibareke, T., & Laassiri, J. (2020). Using Big Data-machine learning models for diabetes prediction and flight delays analytics. Journal of Big Data, 7(1). https://doi.org/10.1186/s40537-020-00355-0

Vasudevan, M. (2016). *Big data analytics: predicting traffic flow regimes from simulated connected vehicle messages using data analytics and machine learning.* US Department of Transportation. Retrieved March 14, 2022, from https://rosap.ntl.bts.gov/view/dot/32616

Yisheng, L., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2014). Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 1–9. https://doi.org/10.1109/tits.2014.2345663

(Reused from the [Project Proposal](https://docs.google.com/document/u/0/d/1h3MHoQYYvw7Q7lPh0WYnJcs2fMpxShezMNUoayZOXDw/edit))

**Other Content as needed**

Appendix (?, located in the dataset section)