**TRAFFIC SIMULATION AND PREDICTION**

**INTRODUCTION:**

Traffic congestion is a significant challenge in large cities, leading to delays, increased fuel consumption, air pollution, and frustration among residents. Traditional traffic management methods are insufficient, necessitating innovative solutions to improve urban mobility. In the modern era marked by rapid urbanization and a growing reliance on automobiles, traffic congestion has become a critical issue, hindering economic productivity, and diminishing the overall quality of life in major cities.

To address this problem, we've developed a project that utilizes datasets to build a machine learning model for estimating traffic congestion. Specifically, we employ time series forecasting as our chosen technique. By continuously monitoring both popular and alternative routes, our advanced traffic simulation systems offer valuable insights into traffic patterns, identifying bottlenecks and high-congestion areas. Armed with this information, traffic controllers can implement adaptive measures to regulate traffic flow. Additionally, navigation apps can play a crucial role by providing drivers with alternative routes to avoid congested areas, thereby alleviating strain on key roadways.

This comprehensive approach aims to enhance the efficiency of traffic management, reduce congestion-related issues, and contribute to a smoother and more sustainable urban transportation system.

**Dataset1:**<https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Historical-Congestion-Esti/kf7e-cur8>

The "Chicago Traffic Tracker - Congestion Estimates by Region" dataset offers crucial insights into traffic congestion in Chicago, Illinois. Maintained by the City of Chicago and accessible on the data portal, this dataset spans from January 2018 to the present day. It provides historical congestion estimates for 29 traffic regions in the city. The congestion data is derived from real-time monitoring and analysis of GPS traces received from Chicago Transit Authority (CTA) buses, specifically focusing on arterial streets rather than freeways.

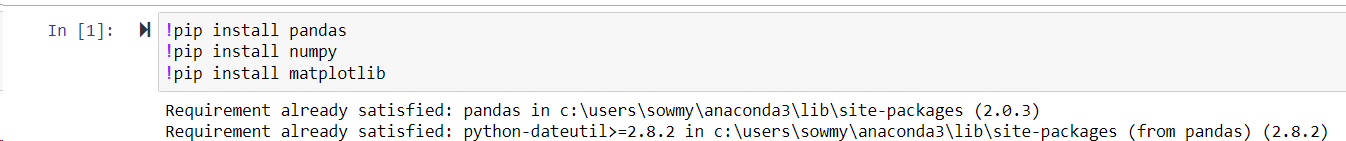
**Dataset2**:<https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Congestion-Estimates-by-Re/t2qc-9pjd>

The dataset encompasses traffic congestion estimates for 29 regions in Chicago. One significant column in this dataset is "LAST\_UPDATE," providing information on traffic congestion updated every 10 minutes. Additionally, the dataset features a "CURRENT\_SPEED" column, offering a reflection of the condition level within each region based on real-time speed data.

**METHODOLOGY:**

**1.Data Acquisition:**

The data was taken from the Chicago open data portal, which archives traffic records spanning five-year intervals. We acquired the dataset through the export option on the portal, available in CSV format. The data extraction process involved direct access to the API. Using the panda’s library in Python, we successfully imported the dataset for analysis.





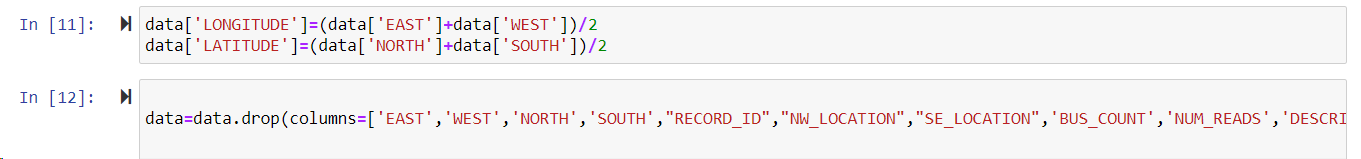


A screenshot of a computer

Description automatically generated

**2.Data Cleaning and Transformation:**

The downloaded dataset is in its raw form, so we are cleaning the data by removing unnecessary columns from the dataset and finding the required columns from the related columns from the dataset. We are storing the cleaned data in a separate file and counting the total no of data reads for each region in Chicago city.

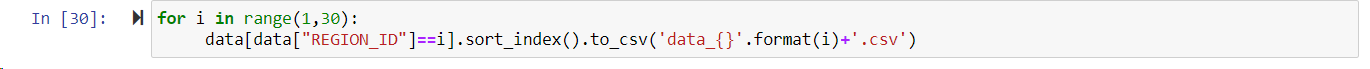


After the cleaning process, we are storing the cleaned data in a separate file and counting the total no of data reads for each region in Chicago city for further data transformation and analysis.

Here, we have 29 regions in a data set, and we are extracting each individual region samples into 29 individual CSV files.

A screenshot of a computer

Description automatically generated



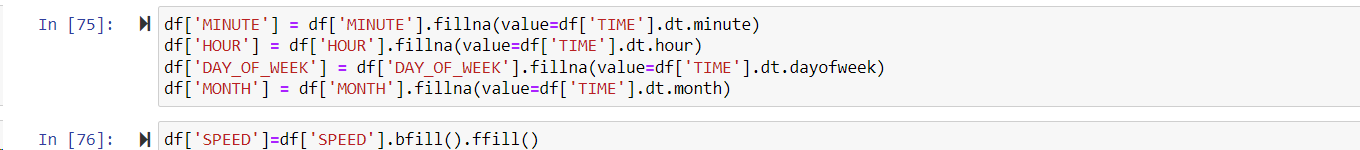
It came to our attention that the data lacks a sorted order and isn't consistently updated every 10 minutes. Furthermore, technical glitches resulted in periods where data upload was disrupted. To streamline the dataset, we extracted only the time information from the TIME column and removed irrelevant columns. As an additional step, we computed the latitude and longitude for each region and incorporated this geographical data into the dataset. This enhancement allows us to visually represent each region on a map using the folium library.

It helps simplify the process of training a model with data containing irregular time intervals by setting the intervals to a consistent value of 10 minutes.

A screenshot of a computer

Description automatically generated

Using backfill and forward fill, we are filling in continuous missing data in the speed column by sorting the data by time. The minute, hour, dayofweek and month columns will be dependent on time columns so, we will be calculating these values from time column using fillna.



**Exploratory Data Analysis:**

To observe the frequency of region 1’S speed, we plotted a graph using the matplotlib function.

A screenshot of a computer

Description automatically generated



To gain more insight into the region 1 speed, we have narrowed down the graph period to May 2023- September 2023 to get more insight.

A screenshot of a computer

Description automatically generated



Based on the above figure, we can see how the data is on weekends and weekdays. On weekends, congestion levels are high, whereas on weekdays, they are low.

A screenshot of a computer

Description automatically generated

For the above figure, we used Python's stats function to analyze the weekly data with the trend and the season.

**3.Model Training and Evaluation**

For model evaluation, based on the horizontal trend analysis, we explored various statistical models, including Simple Exponential Smoothing, Double Exponential Smoothing, and Triple Exponential Smoothing. By thorough analysis and experimentation (trial and error), we determined that Triple Exponential Smoothing is particularly adept at capturing the nuanced patterns within the traffic flow data.

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

The above graph illustrates that the model's estimations are based on the provided data. The blue plot represents the actual data, while the orange plot represents the model's predicted data. It is evident from the visualization that the model performs accurately at peak points and maintains a consistent approximation in other scenarios.

Due to the limitation in visualizing detailed insights into the model's behavior, we are generating this specific graph for a selected range of samples, spanning from 80000 to 80500.

A screenshot of a graph

Description automatically generated

A screenshot of a computer

Description automatically generated

Having completed the training and testing phases with data from June to August 2023, we proceeded to utilize the model for predictions on future data. Based on our observations on data, we obtain 1008 samples per week for a single region So, we applied the model to forecast the subsequent 2500 samples, covering the period from 22 August to 7 September.

A screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

Upon observation, we note that the model's predictions for traffic congestion over the three weeks, depicted in orange, closely resemble patterns seen in previous months.

**4.Data Visualization**

We are reading the cleaned dataset that contains 29 regions data.

A screenshot of a computer

Description automatically generated

We are importing the required folium library and pointing 29 regions from the Chicago city into an interactive maps.

A map with a location pin on it

Description automatically generated

We employed ipywidgets to create interactive widgets, allowing us to select both the source and destination from a dropdown menu and pointing those two regions on a map.

A screenshot of a computer

Description automatically generated

We employed ipywidgets to create interactive widgets, allowing users to select both the source and destination from a dropdown menu.

A screenshot of a computer

Description automatically generated

A map with blue pins on it

Description automatically generated

In this context, we utilized **OSMnx** and **networkx** to determine the optimal path for the user to reach their destination in the minimum amount of time. OSMnx, which stands for OpenStreetMap, serves the purpose of visualizing and analyzing real-world street models. On the other hand, networkx is employed for handling complex structures, such as maps and graphs.

A screenshot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

The depicted figure illustrates the shortest path between the specified source and destination points. The green pointer designates the source location, while the red pointer indicates the destination. We computed three alternative routes, considering traffic congestion, and employed time as the sorting parameter for comparison.

A map with a blue line

Description automatically generated

The below code is used to find the alternative shortest path between the specified source and destination points.

A screenshot of a computer code

Description automatically generated

The below depicted figure illustrates the alternative shortest paths between the specified source and destination points.

A map with a red line

Description automatically generated

A map of a city

Description automatically generated

**CHALLENGES FACED:**

* Managing the large volume of data and the associated processing demands.
* Implementation of outlier detection mechanisms to identify and handle anomalous data points effectively.
* During the model construction using Exponential Smoothing, challenges were encountered, particularly in achieving satisfactory accuracy with simple and double exponential smoothing techniques.
* While determining alternative routes, it was observed that some routes overlapped with the original one, although this occurrence was infrequent.
* Data intervals are irregular, deviating from the specified 10-minute increments, occurring approximately every 9, 10, or 11 minutes. This irregularity needs to be addressed.
* Identification and handling of missing dates by employing forward and backward fill methods.
* Considering the limitation of the Google Routes API, which offers the first 40,000 API calls for free before incurring charges, it is imperative to explore alternative solutions.

**FUTURE WORK:**

**Incorporating External Factors (Weather Impact):** weather data into the model to gain insights into how weather conditions influence traffic congestion.

**Scalability and Generalization**: Assess the effectiveness of the model across various cities and regions to gauge its scalability and generalization capabilities.

**User Feedback and Engagement:** Gather user feedback to assess the impact of predictions on commuters' experiences.

**Seasonal Variation Analysis:** Study seasonal variations in traffic congestion to tailor the model accordingly. Additionally, investigate the impact of holidays and special events on traffic flow to enhance the accuracy of predictions.

**Collaborative Data Sharing:** Engage in collaboration with other cities and transportation agencies to facilitate the sharing of traffic data and insights, fostering cross-city traffic management.