UNIVERSITY OF MÁLAGA

MASTER DEGREE IN BIG DATA AND
ARTIFICIAL INTELLIGENCE

MASTER THESIS

CHICAGO TRAFFIC ANALYSIS

NURIA MARÍA HABA DÍAZ
MÁLAGA, 2024

MASTER DEGREE IN BIG DATA AND ARTIFICIAL INTELLIGENCE

CHICAGO TRAFFIC ANALYSIS

Author: Nuria María Haba Díaz

Supervisor: Antonio J. Nebro and Cristóbal Barba

Department: Computer Science

Keywords: Urban Traffic, Chicago Traffic analysis, Predictive Modelling, Machine Learning, Apache Spark, Regression Analysis, Traffic Patterns, Big Data, Data Analytics, Geo-Spatial Analysis, Root Mean Squared Error (RMSE), Open data, Data processing, Big Data, Spark Streaming, Traffic data, Data visualization

Abstract

This project focuses on leveraging Apache Spark and Machine Learning techniques to analyze and predict traffic patterns in urban segments. The study utilizes a comprehensive dataset spanning from 2018, including information about segment characteristics of historical traffic data.

The initial phase involves exploratory data analysis to gain insights into the temporal and spatial aspects of traffic behavior. Visualizations, statistical summaries, and geospatial analyses are employed to understand traffic variations over time, identifying trends, and assessing the impact of external factors such as time of day, day of the week, and month.

Subsequently, a predictive model is developed using Spark's Machine Learning library, MLlib. The model employs linear regression to estimate travel times for specific segments, considering various features such as hour, day of the week, and month. The model's performance is evaluated, and the Root Mean Squared Error (RMSE) is employed as a metric.

To enhance the predictive capabilities, the project explores the possibility of predicting percentiles of travel times, rather than focusing solely on the mean. Different percentiles, including 25th, 50th, and 75th, are analyzed independently, leading to the development of tailored regression models for each.

Acronyms

API	Application Programming Interface
ETL	Extract, Transform, Load
ML	Machine Learning
MLib	Machine Learning Library (in Apache Spark Context)
RMSE	Root Mean Squared Error

Contents

List of Figures

Part I

Introduction

Chapter 1

Introduction

Content

Synopsis

In the ever-evolving urban landscape, efficient traffic management is a critical facet of ensuring smooth transportation systems. With the advent of big data and data analytics, there is an unprecedented opportunity to gain insights into urban traffic patterns and predict travel times accurately. This project delves into the application domain of urban traffic data analysis, leveraging Machine Learning techniques to model and predict estimated travel times for different road segments

1.1 Objective

The main objective of this study is to develop a predictive model for estimating travel times on urban road segments using historical traffic data. The aim is to enhance understanding of temporal patterns and variability in travel times, enabling

improved traffic planning and management in urban environments. Additionally, the practical application of the model is explored to enhance the visualization of traffic congestion on a map.

This objective provides an overview of the direction and purpose of the work, emphasizing the importance of accuracy in travel time predictions and its practical application in urban traffic management.

1.2 State of the art

In recent years, Apache Spark has emerged as a leading platform for large-scale data processing and distributed analytics. Its distributed architecture and ability to perform in-memory computations have revolutionized the way big data problems are tackled. In the realm of machine learning, Spark offers a suite of powerful tools through its MLlib library, enabling developers to implement machine learning algorithms in a scalable and efficient manner.

One of the fundamental algorithms in supervised learning is linear regression, which seeks to model the relationship between one or more independent variables and a continuous dependent variable.

In the context of urban traffic prediction, existing studies have primarily focused on leveraging machine learning algorithms and historical traffic data to forecast travel times. Techniques such as linear regression, decision trees, and neural networks have been employed to model complex relationships between various factors influencing traffic conditions. Moreover, the integration of real-time data from sources like GPS and traffic sensors has become prevalent, enhancing the accuracy of predictions.

Recent advancements in data visualization tools, such as Folium, have enabled the creation of interactive maps that provide dynamic representations of traffic conditions. These technologies contribute to a more comprehensive understanding of urban mobility patterns.

However, challenges persist in accurately capturing the dynamic nature of traffic, especially during peak hours and under varying weather conditions. This study seeks to address these challenges by exploring the use of percentiles in predictive modeling and employing advanced data visualization techniques for effective traffic management.

The state of the art underscores the ongoing efforts to refine predictive models and emphasizes the importance of real-time data integration and interactive visualization in improving urban traffic management strategies.

1.3 Methodology and guidelines followed

This research adopts a comprehensive approach to enhance urban traffic prediction and management. The methodology comprises the following key steps:

Data Collection and Preprocessing:

- Acquiring historical data from Chicago Open Data portal
- Cleaning and preprocessing the data to handle missing values, outliers, and ensure consistency.

Exploratory Data Analysis (EDA):

- Conducting an in-depth exploration of the dataset to understand traffic patterns, trends, and potential influencing factors.
- Utilizing statistical and visual analysis to identify patterns and correlations.

Predictive Modeling with Percentiles:

- Implementing Machine Learning models, specifically linear regression, to predict travel times for urban traffic segments.
- Introducing the novel approach of using percentiles to enhance the granularity of predictions and capture variations in traffic conditions.

Model Evaluation and Optimization:

- Assessing the performance of predictive models using metrics such as Root Mean Squared Error (RMSE).
- Exploring optimization techniques, including hyperparameter tuning, to improve model accuracy.

Grouped Analysis:

- Conducting a grouped analysis by segment, hour, day of the week, and month to derive insights into temporal and spatial variations in traffic.

Data Visualization:

- Leveraging advanced data visualization tools like Folium to create interactive maps.
- Mapping traffic segments with colors based on percentiles, providing a visual representation of traffic conditions.

Results and Implications:

- Analyzing the results and drawing insights into the feasibility and advantages of incorporating percentiles in traffic prediction models.
- Discussing the implications for urban traffic management and potential areas for future research.

1.4 Document Structure

Chapter 1: Introduction This chapter provides a comprehensive overview of the research, introducing the background, context, and significance of the study. It outlines the objectives, scope, and relevance of the investigation into enhancing urban traffic prediction.

Chapter 2: Methodology In this chapter, the research methodology is delineated, detailing the procedures and techniques employed to conduct the study. It discusses the selection of data sources, modeling approaches, and the rationale behind methodological choices.

Chapter 3: Exploratory Data Analysis (EDA) This section involves a thorough examination of the dataset, exploring key patterns, trends, and characteristics. Descriptive statistics, data cleaning procedures, and initial insights into urban traffic data are presented.

Chapter 4: Predictive Modeling Focusing on data-driven approaches, this chapter delves into the development and evaluation of predictive models for urban traffic. It discusses the chosen algorithms, model training, validation, and the interpretation of results.

Chapter 5: Data Visualization Highlighting the importance of visual representation, this chapter discusses the creation and interpretation of visualizations to communicate complex traffic patterns effectively. It explores the use of maps, charts, and graphs for conveying insights.

Chapter 6: Results and Discussion The chapter presents the findings of the study, evaluating the performance of predictive models and the insights

gained through grouped analysis. It provides a platform for discussing implications, limitations, and potential avenues for further research.

1.5 Scope of application

The research is primarily focused on improving urban traffic prediction within the context of a specific city or urban area. The application domain encompasses:

Urban Traffic Management: The findings and models developed aim to contribute to the optimization of urban traffic flow, aiding traffic management authorities in making informed decisions.

Transportation Planning: The study has implications for urban transportation planning, offering insights into traffic patterns and estimated travel times that can inform infrastructure development and planning.

Data-Driven Decision Making: By leveraging data-driven approaches, the research seeks to empower stakeholders with tools for more effective decision-making related to urban mobility and traffic dynamics.

Smart City Initiatives: The outcomes align with the broader goals of smart city initiatives, aiming to enhance the efficiency, sustainability, and livability of urban spaces through innovative data analytics and predictive modeling.

Part II

Project Development

Chapter 2

Methodology

Content

Synopsis

In this chapter, the research methodology is detailed, outlining the steps taken to acquire, process, and analyze the data. The methodology is designed to ensure the reliability and relevance of the findings.

2.1 Data Source

In this section, the process of data collection is detailed, emphasizing the use of the Chicago Traffic Tracker Historical Congestion Estimation dataset. The dataset's features, acquisition methods, and any preprocessing steps applied to ensure data quality are explained.

(`https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Historical`
`sxs8-h27x`.)

The dataset used in this study was obtained through the API of the City of Chicago Data Portal. Specifically, data for segments constituting Ashland Street

in Chicago was targeted. To ensure relevance and reliability, only segments with positive vehicle speeds were considered. The data retrieval process spanned from March 2018 to October 2023, encompassing all available records within this timeframe.

The extracted dataset comprises a total of 1,881,410 entries, each providing valuable insights into historical traffic congestion on Ashland Street. To offer a comprehensive overview, the following columns were included in the dataset:

- **'segment_id':** Unique arbitrary number to represent each segment.
- **'speed':** Estimated traffic speed in miles per hour. A value of -1 means no estimate is available.
- **'street':** Street name of the traffic segment.
- **'direction':** Traffic flow direction for the segment.
- **'from_street':** Start street for the segment in the direction of traffic flow.
- **'to_street':** End street for the segment in the direction of traffic flow.
- **'length':** Length of the segment in miles.
- **'street_heading':** The position of the segment in the address grid. North, South, East, or West of State and Madison.
- **'comments':** Comments
- **'length':** Length of the segment in miles.
- **'bus_count':** Number of buses providing a GPS feed used to estimate congestion.
- **'message_count':** Number of GPS probes received(or used) for estimating the speed for that segment.
- **'hour':** Hour of the day.
- **'day_of_week':** Day of the week. Sunday = 1
- **'month':** Month of the year.
- **'record_id':** A unique identifier for each record in the dataset.
- **'start_latitude':** Latitude of the start of the segment.

- **'start_longitude':** Longitude of the start of the segment.
- **'end_latitude':** Latitude of the end of the segment.
- **'end_longitude':** Longitude of the end of the segment.
- **'start_location':** Location of the start of the segment.
- **'end_location':** Location of the end of the segment.

This rich dataset serves as the foundation for subsequent exploratory data analysis, predictive modeling, and grouped analyses to derive insights into traffic patterns and congestion dynamics along Ashland street.

2.2 Data Extraction Process

The extraction of data was facilitated by utilizing the Socrata API through the sodapy Python library. The data extraction process involved making iterative requests, each focusing on a specific segment, to ensure comprehensive coverage of the target street, Ashland, in Chicago. Batching was employed to manage the extraction effectively, considering the substantial volume of data. The following code snippet illustrates the process employed for acquiring the relevant dataset:

```
batch = client.get(
      socrata_dataset_identifier,
      where="segment_id >= 141 AND segment_id < 142 AND speed >= 0",
      select="time, segment_id, speed, street, from_street,
          to_street, length, hour, day_of_week, month,
          start_latitude, start_longitude, end_latitude,
          end_longitude, start_location, end_location",
      limit=batch_size,
      offset=offset
  )
```

This extraction approach ensures the acquisition of a targeted dataset aligned with the research objectives and requirements.

2.3 Data Preprocessing

Prior to conducting any analyses, a comprehensive data preprocessing is needed to ensure the quality and relevance of the dataset. The preprocessing steps included:

Filtering by Relevant Segments: In the extraction step the dataset was filtered to include only segments pertaining to Ashland street. Segments with identifiers between 141 and 150 were considered for further analysis.

Exclusion of Negative Speeds: In the extraction step entries with negative vehicle speeds, indicative of erroneous or irrelevant data, were excluded to enhance the accuracy of subsequent analyses.

Batched Data Retrieval: In the extraction step due to the large volume of data, requests were made in batches using the Socrata API. This approach facilitated efficient data retrieval for each segment while managing the limitations imposed by the API.

The preprocessing steps laid the groundwork for subsequent exploratory data analysis and predictive modeling, ensuring that the dataset aligns with the specific context and objectives of the analysis.

The data appearance just downloaded from API can be checked in figure 2.1.

2.4 Data Transformation

Data Transformation is performed using **Google Colab**, a cloud-based platform that provides a convenient and collaborative environment for data analysis and manipulation. Leveraging the power of Apache Spark, the transformation process seamlessly handled large-scale data operations, ensuring efficiency and reliability.

The Data Transformation section focuses on the preparatory steps taken to refine and structure the raw dataset obtained from the Chicago Traffic Tracker API. This process involves various transformations and adjustments aimed at enhancing the dataset's usability for subsequent analyses. Key tasks include unit conversion, handling missing or erroneous values, and organizing the data in a format conducive to exploratory data analysis and predictive modeling.

During the Data Transformation process, a conversion of measurement units was conducted to ensure consistency in the analysis. Specifically, the columns

time	segment_id	speed	street	from_street	to_street	length	hour	day_of_week	month
2018-03-02T17:50:...	143	24	Ashland	47th	51st	0.5	17	6	3
2018-03-02T18:01:...	143	20	Ashland	47th	51st	0.5	18	6	3
2018-03-03T22:50:...	143	26	Ashland	47th	51st	0.5	22	7	3
2018-03-01T11:50:...	143	31	Ashland	47th	51st	0.5	11	5	3
2018-03-03T07:10:...	143	28	Ashland	47th	51st	0.5	7	7	3
2018-03-01T12:20:...	143	18	Ashland	47th	51st	0.5	12	5	3
2018-03-02T18:30:...	143	20	Ashland	47th	51st	0.5	18	6	3
2018-03-02T18:40:...	143	22	Ashland	47th	51st	0.5	18	6	3
2018-03-03T23:10:...	143	27	Ashland	47th	51st	0.5	23	7	3
2018-03-01T12:50:...	143	18	Ashland	47th	51st	0.5	12	5	3

start_latitude	start_longitude	end_latitude	end_longitude	start_location	end_location
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...

Figure 2.1: Raw Traffic Data

'length' and **'speed'** originally measured in miles, were transformed into the corresponding standard unit of measure for ease of interpretation.

This means that **'length'** is now presented in kilometers (km), and **'speed'** in kilometers per hour (km/h). This adjustment is crucial to ensure consistency in the metrics used in predictive modeling and exploratory data analysis. For the purpose of conversion, miles are multiplied by 1.60934 to obtain the equivalent distance in kilometers.

```
traffic_data = (
    traffic_data
    .withColumn('speed',col('speed') * 1.60934)
    .withColumn('length',col('length') * 1.60934)
)
```

After the unit conversion, the next step involves the conversion of data types. The majority of the columns were initially classified as string types. To facilitate subsequent analysis and modeling tasks, these columns were converted to their respective numeric types. This ensures a consistent and compatible data structure for downstream operations.

```
traffic_data = (
    traffic_data
    .withColumn("time", col("time").cast("timestamp"))
    .withColumn("segment_id", col("segment_id").cast("int"))
    .withColumn("speed", col("speed").cast("float"))
    .withColumn("hour", col("hour").cast("int"))
    .withColumn("day_of_week", col("day_of_week").cast("int"))
    .withColumn("month", col("month").cast("int"))
    .withColumn("start_latitude", col("start_latitude").cast("double"))
    .withColumn("start_longitude", col("start_longitude").cast("double"))
    .withColumn("end_latitude", col("end_latitude").cast("double"))
    .withColumn("end_longitude", col("end_longitude").cast("double"))
)
```

```
root
 |-- time: timestamp (nullable = true)
 |-- segment_id: integer (nullable = true)
 |-- speed: float (nullable = true)
 |-- street: string (nullable = true)
 |-- from_street: string (nullable = true)
 |-- to_street: string (nullable = true)
 |-- length: double (nullable = true)
 |-- hour: integer (nullable = true)
 |-- day_of_week: integer (nullable = true)
 |-- month: integer (nullable = true)
 |-- start_latitude: double (nullable = true)
 |-- start_longitude: double (nullable = true)
 |-- end_latitude: double (nullable = true)
 |-- end_longitude: double (nullable = true)
 |-- start_location: struct (nullable = true)
 |    |-- coordinates: array (nullable = true)
 |    |    |-- element: double (containsNull = true)
 |    |-- type: string (nullable = true)
 |-- end_location: struct (nullable = true)
 |    |-- coordinates: array (nullable = true)
 |    |    |-- element: double (containsNull = true)
 |    |-- type: string (nullable = true)
```

In the process of feature engineering, a new column named **'estimated_time'** was created by dividing the **'length'** of each segment by its corresponding **'speed'**. The result, expressed in hours, was then converted to minutes for better inter-

pretability. This additional feature provides an estimation of the time required to traverse each road segment, offering valuable insights into traffic conditions.

After the computation of the **'estimated_time'** column, a thorough check was performed to identify and handle any missing or NaN values in the dataset. This ensures the robustness of subsequent analyses and modeling processes by addressing potential data integrity issues. The dataset revealed a total of 1238 missing or null values, primarily in the **'estimated_time'** column. To address this, the missing entries were replaced with the median value (50th percentile) of the **'estimated_time'**'' column. This imputation method ensures that the dataset remains representative and minimizes potential biases introduced by missing data.

```
traffic_data_time = (
    traffic_data
    .withColumn('estimated_time', round((col('length') / col('speed') * 60),2))
)

traffic_data_time
    .select('estimated_time')
    .filter(F.col('estimated_time').isNull() | F.isnan('estimated_time'))
    .count()
Output: 1238

percentile_50_value = (
    traffic_data_time
    .approxQuantile('estimated_time', [0.5], 0.001)[0]
)

traffic_data_time = (
    traffic_data_time
    .fillna(percentile_50_value, subset=['estimated_time'])
)
```

The data appearance after the previous transformations and conversions can be checked in figure 2.2.

time	segment_id	speed	street	from_street	to_street	length	hour	day_of_week	month
2018-03-02 17:50:21	143	38.62	Ashland	47th	51st	0.80467	17	6	3
2018-03-02 18:01:07	143	32.19	Ashland	47th	51st	0.80467	18	6	3
2018-03-03 22:50:16	143	41.84	Ashland	47th	51st	0.80467	22	7	3
2018-03-01 11:50:19	143	49.89	Ashland	47th	51st	0.80467	11	5	3
2018-03-03 07:10:17	143	45.06	Ashland	47th	51st	0.80467	7	7	3
2018-03-01 12:20:36	143	28.97	Ashland	47th	51st	0.80467	12	5	3
2018-03-02 18:30:53	143	32.19	Ashland	47th	51st	0.80467	18	6	3
2018-03-02 18:40:04	143	35.41	Ashland	47th	51st	0.80467	18	6	3
2018-03-03 23:10:16	143	43.45	Ashland	47th	51st	0.80467	23	7	3
2018-03-01 12:50:17	143	28.97	Ashland	47th	51st	0.80467	12	5	3

start_latitude	start_longitude	end_latitude	end_longitude	start_location	end_location	estimated_time
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.25
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.5
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.15
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	0.97
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.07
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.67
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.5
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.36
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.11
41.8086055077	-87.6651559056	41.8013192816	-87.6649598466	{[-87.6651559056,...	{[-87.6649598466,...	1.67

Figure 2.2: Transformed Traffic Data

Chapter 3

Exploratory Data Analysis (EDA)

Content

Synopsis

Exploratory Data Analysis (EDA) is a crucial component in the process of understanding the nature of the data before applying any modeling or analytical techniques. In this section, the approach taken to explore and visualize the data collected from the city of Chicago is described with the aim of gaining a deeper understanding of its characteristics.

3.1 Descriptive Statistics

Descriptive statistical analyses are conducted on key variables, including **'length'**, **'speed'** and **'estimated_time'**. These statistics provide information about the central tendency, dispersion, and shape of the distribution of these variables.

```
traffic_data_time.describe(['speed', 'length', 'estimated_time']).show()
```

The Descriptive Statistics can be checked in figure 3.1.

summary	speed	length	estimated_time
count	1881410	1881410	1881410
mean	41.53971869814013	0.8947982065603061	1.333447078245766
stddev	8.495203555560462	0.2668682576298243 6	0.5523479825212867
min	0.0	0.6437360000000001	0.79
max	65.98	1.60934	59.98

Figure 3.1: Descriptive Statistics

3.2 Speed Distribution

Understanding the distribution of vehicle speeds is a key aspect of traffic analysis. The speed distribution provides insights into the typical travel speeds on the analyzed road segments. In this section, we delve into the speed distribution of the collected data, employing visualizations and descriptive statistics.

```
traffic_data_time.select('speed').toPandas().hist(bins=20)
```

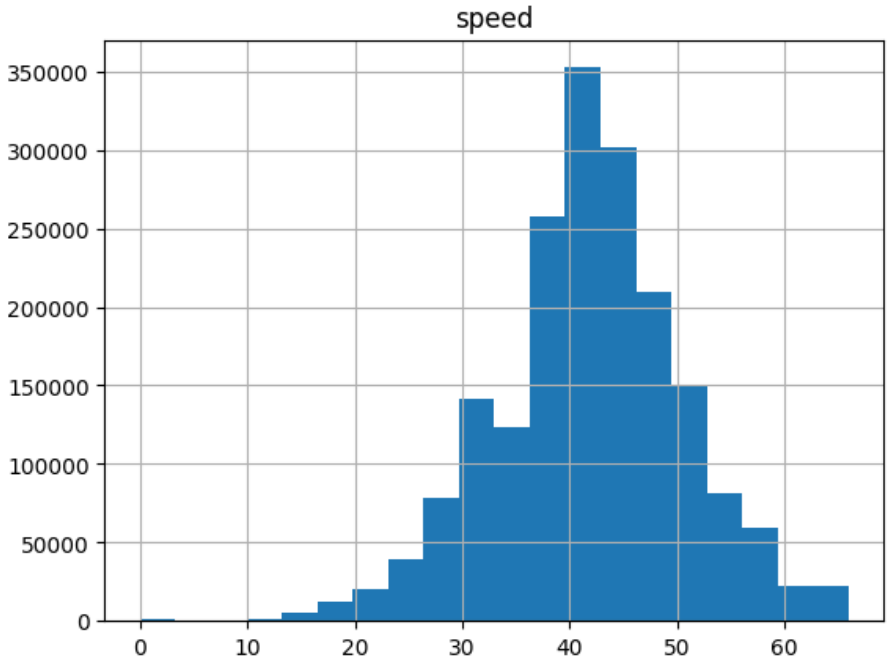

Figure 3.2: Speed Distribution

The Speed Distribution graphic in figure 3.2 depicts the distribution of observed speeds in the dataset. This type of graph, also known as a histogram,

shows the frequency with which different speed ranges occur. In this case, it is evident that the majority of speeds are concentrated in the range of 40 to 42 kilometers per hour.

The distribution centered around a specific value indicates that this speed is the most common or frequent in the data. In other words, most of the time, vehicles at that specific location tend to move at a speed close to 41 kilometers per hour. The spread of speeds beyond this central value can also be observed in the graph.

3.3 Relationship between Speed and Hour of the Day

Analyzing the relationship between vehicle speed and the hour of the day is crucial for understanding how traffic conditions vary throughout different times. This section explores how speed changes based on the time of day and aims to identify peak hours and potential congestion periods.

```
avg_speed_by_hour = traffic_data_time
    .groupBy('hour')
    .agg(F.mean('speed').alias('avg_speed'))
```

The resulting Relationship between Speed and Hour of the Day can be checked in figure 3.3.

In the graphic, each point corresponds to a specific hour, and the average speed value reflects the typical speed recorded during that particular time.

For example, during the early morning hours (around 5 AM), the average speed is relatively high at approximately 42.81 kilometers per hour. As the morning progresses, the average speed remains above 40 kilometers per hour until the afternoon. There is a slight dip in average speed during the afternoon, and it picks up again in the evening.

These findings suggest a pattern of fluctuating average speeds throughout the day, indicating potential periods of congestion or smoother traffic flow.

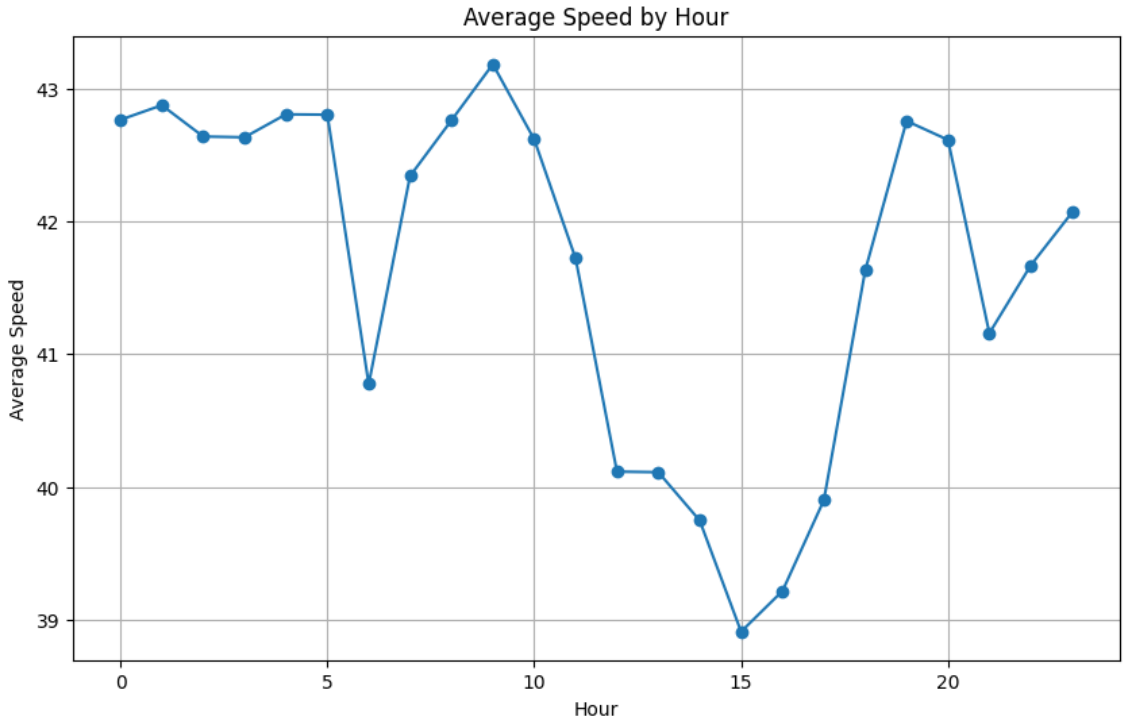

Figure 3.3: Relationship between Speed and Hour of the Day

3.4 Patterns or Variations by Day

Analyzing if there are any patterns or variations in traffic on different days of the week.

```
daywise_speed = (
    traffic_data
    .groupBy('day_of_week')
    .agg({'speed': 'avg'})
    .toPandas()
)
```

The resulting Relationship between Speed and Day of the Week can be checked in figure 3.4.

In the graphic, each point corresponds to a specific day (Sunday is 1), and the average speed value reflects the typical speed recorded during that particular time. For example, it seems that on Sunday and Saturday the average speed is a little bit higher than the other days (but only 1 kilometer per hour more or less).

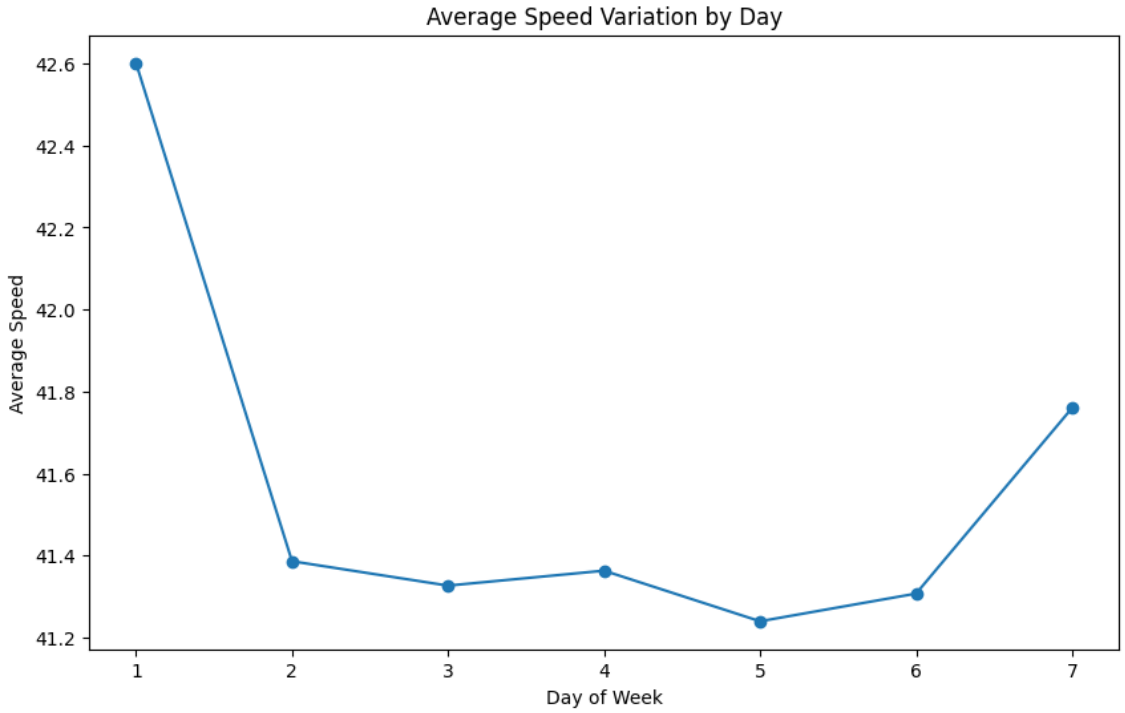

Figure 3.4: Relationship between Speed and Day (Sunday is 1)

Chapter 4

Predictive Modeling

Content

Synopsis

This section is dedicated to employing the Linear Regression model for predictive analysis. The focus is on utilizing regression techniques to forecast the **'estimated_time'** variable based on the selected features. The simplicity and interpretability of linear regression make it an ideal choice for this scenario.

The evaluation of the model's performance involves metrics such as Root Mean Squared Error (RMSE), providing a quantitative measure of the predictive

accuracy. The objective is to understand how well the linear regression model captures the underlying patterns and variations in the dataset.

4.1 Data Preparation

To facilitate the construction of the linear regression model, where we have a large volume of data for each combination of **'segment_id'**, **'hour'**, **'day_of_week'** and **'month'** a preliminary step involves aggregating the data. This aggregation is performed by computing percentiles (25th, 50th, and 75th) of the **'estimated_time'** for each group.

```
traffic_regresion_grouped_perc = (
    traffic_regresion
    .groupby('segment_id', 'hour', 'day_of_week', 'month')
    .agg(
        F.expr('percentile_approx(estimated_time, 0.5)')
            .alias('q50_estimated_time'),
        F.expr('percentile_approx(estimated_time, 0.25)')
            .alias('q25_estimated_time'),
        F.expr('percentile_approx(estimated_time, 0.75)')
            .alias('q75_estimated_time')
    )
)
```

The resulting grouped dataframe looks like in figure 4.1.

segment_id	hour	day_of_week	month	q50_estimated_time	q25_estimated_time	q75_estimated_time
141	0	1	1	1.11	1.03	1.25
141	0	1	5	1.15	1.03	1.3
141	0	1	6	1.11	1.03	1.3
141	0	2	2	1.11	1.03	1.25
141	0	2	11	1.11	1.07	1.3
141	0	3	1	1.15	1.11	1.3
141	0	3	7	1.11	1.03	1.3
141	0	4	2	1.11	0.97	1.3
141	0	4	5	1.11	1.03	1.3
141	0	6	1	1.11	1.03	1.3

Figure 4.1: Traffic dataframe prepared for regression model

4.2 Feature Selection

In the feature selection process, the **VectorAssembler** tool has been employed to combine multiple input columns, namely **'segment_id'**, **'hour'**, **'day_of_week,'** and **'month'** into a single feature vector. This consolidated feature vector, labeled as **'features'** serves as the input for the predictive modeling phase, enabling the algorithm to consider multiple factors simultaneously during training and prediction.

```
vector_assembler = VectorAssembler(inputCols=['segment_id', 'hour',
    'day_of_week', 'month"], outputCol='features')
```

4.3 Model Training

Model training involves the construction of the regression model using the selected features. Here, a linear regression model is utilized to predict the **'segment_id'** based on the chosen input features.

4.3.1 Data Transformation

The initial step involves transforming the dataset by assembling the selected features into a feature vector. The VectorAssembler is employed for this purpose, combining features such as **'segment_id'**, **'hour'**, **'day_of_week'** and **'month'** into a single feature vector named **'features'**.

```
df_assembled = vector_assembler.transform(traffic_regresion)

(train_data, test_data) = df_assembled.randomSplit([0.8, 0.2], seed=42)
```

4.3.2 Model Initialization

Following data transformation, the linear regression model is initialized using the **LinearRegression** class from the MLlib library. The features column is set to **'features'** and the label column is set to **'q50_estimated_time'**

```
regressor = LinearRegression(featuresCol='features',
                             labelCol='q50_estimated_time')
```

4.3.3 Model Fitting

The model is then fitted to the training data using the fit method. This process involves adjusting the model parameters to minimize the difference between the predicted and actual values of the **'q50_estimated_time'**.

```
regressor = regressor.fit(train_data)
```

This completes the model training phase, and the fitted regression model is ready for evaluation and interpretation.

4.4 Model Evaluation

Model evaluation assesses the performance of the trained regression model. Metrics such as Root Mean Squared Error (RMSE) are commonly employed to gauge how well the model predicts the **'q50_estimated_time'** compared to the actual values.

After training the linear regression model, predictions are generated on the test dataset. The model's performance is then assessed using various evaluation metrics, including the Root Mean Squared Error (RMSE). The RMSE quantifies the difference between the predicted and actual values of the **'q50_estimated_time'** providing an overall measure of prediction accuracy.

```
eval = RegressionEvaluator(labelCol='q50_estimated_time',
                           predictionCol='prediction',
                           metricName='rmse')
```

4.4.1 Root Mean Squared Error (RMSE)

RMSE is a standard metric for regression tasks, measuring the average deviation between predicted and actual values. A lower RMSE indicates better model performance. The RMSE is a crucial metric in regression analysis, representing the standard deviation of the residuals. A lower RMSE indicates better model performance, reflecting reduced prediction errors.

```
rmse = eval.evaluate(pred.predictions)
print(f"Root Mean Squared Error (RMSE): {rmse}")

Output: Root Mean Squared Error (RMSE): 0.21954708646762117
```

4.4.2 Mean Absolute Error (MAE)

MAE represents the average absolute difference between predicted and actual values. It provides insights into the model's overall accuracy.

```
mae = eval.evaluate(pred.predictions, {eval.metricName: 'mae'})
print('Mean Absolute Error (MAE): %.3f' % mae)

Output: Mean Absolute Error (MAE): 0.147
```

The output indicates that, on average, the model's predictions deviate by approximately 0.147 units from the true values of the estimated time. A lower MAE value suggests better model accuracy, making it a valuable metric for assessing the model's predictive performance.

4.4.3 R-squared (R^2)

R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. A higher R^2 indicates a better fit of the model to the data.

```
r2 = eval.evaluate(pred.predictions, {eval.metricName: 'r2'})
print('coefficient of determination R2: %.3f' %r2)

Output: coefficient of determination R2: 0.177
```

In the presented output the R^2 value is approximately 0.177. This implies that around 17.7% of the variability in the estimated time can be explained by the linear regression model using the selected features.

4.4.4 Residual Analysis

Residual analysis is a crucial step in assessing the performance of a regression model. Residuals are the differences between the observed values and the values predicted by the model. Analyzing these residuals can provide insights into the model's accuracy and uncover patterns or trends in the errors.

```
residuals = pred.predictions
    .withColumn('residuals', F.col('q50_estimated_time') - F.col('prediction'))
```

```
residuals_pd = residuals.select('hour', 'residuals').toPandas()
```

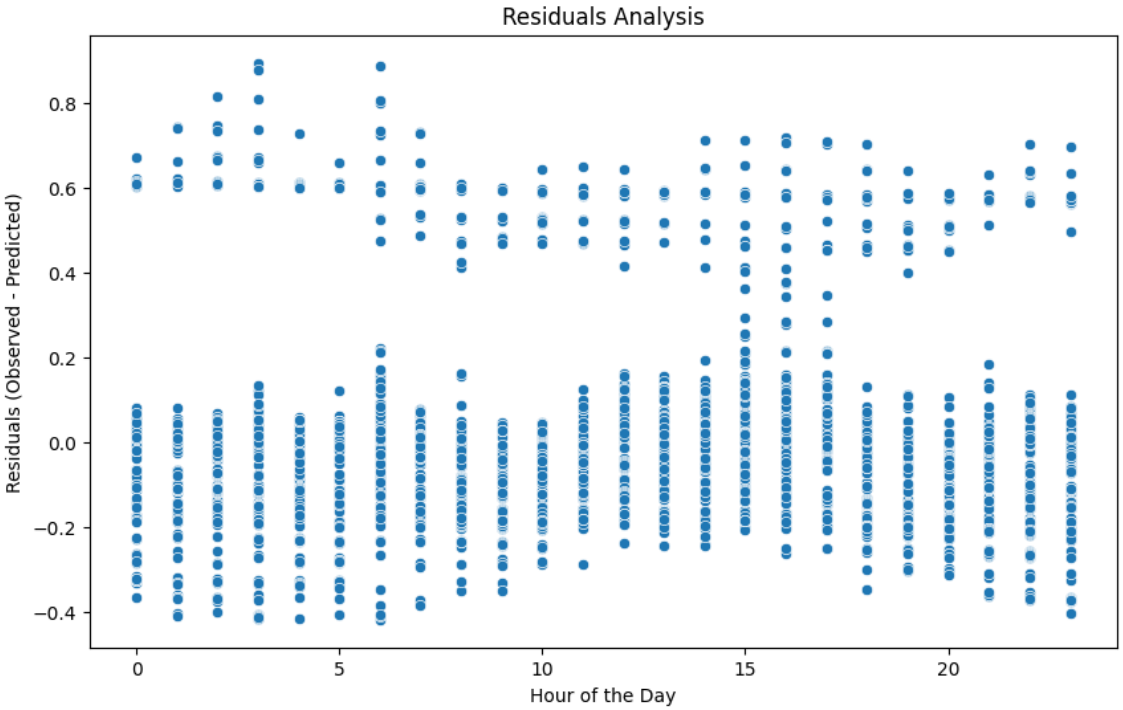

Figure 4.2: Residuals Analysis

Chapter 5

Data Visualization

Content

Synopsis

In this section, we delve into the geospatial visualization of traffic data using the Folium library. The tool enables us to map the estimated speed on different segments of Ashland Street for specific times of the day, days of the week, and months. We use data from start and end segments, the time of day, day of the week, and the month as input parameters to generate interactive maps and gain a better understanding of traffic dynamics in the selected area.

5.1 Parameter Selection

Select start and end segments, a time of day, a day of the week, and a month as input variables for visualization.

```
start_segment = 141
end_segment = 147
hour = 17
day_of_week = 6
month = 3
```

5.2 Calculation of Percentiles for Grouped Traffic Data

In this step, we aggregate traffic data by segment, hour, day of the week, and month. Subsequently, we compute key percentiles (25%, 50%, and 75%) for the estimated time within each group. This provides a summarized view of the distribution of estimated times across various traffic conditions, allowing for a more comprehensive understanding of temporal and spatial patterns in the dataset.

```
traffic_data_perc = traffic_regresion
    .groupby('segment_id', 'hour', 'day_of_week', 'month')
    .agg(
         F.first('estimated_time').alias('estimated_time'),
         F.expr('percentile(estimated_time, array(0.25))')[0]
            .alias('perc %25'),
         F.expr('percentile(estimated_time, array(0.50))')[0]
            .alias('perc %50'),
         F.expr('percentile(estimated_time, array(0.75))')[0]
            .alias('perc %75')
    )
```

The resulting dataframe looks like in figure 5.1.

5.3 Filtering Grouped Traffic Data

In this step, the grouped traffic data is filtered based on specific conditions. The DataFrame, named traffic_regresion_filtered, is refined to include only the data relevant to a particular segment range, hour of the day, day of the week, and month. This focused subset of data will be utilized for detailed examination and visualization, allowing for a more targeted exploration of traffic patterns under specified circumstances.

segment_id	hour	day_of_week	month	estimated_time	perc %25	perc %50	perc %75
141	0	1	1	1.2	1.03	1.11	1.25
141	0	1	5	1.76	1.03	1.15	1.3
141	0	1	6	1.5	1.03	1.11	1.3
141	0	2	2	1.11	1.03	1.11	1.25
141	0	2	11	0.97	1.07	1.13	1.3
141	0	3	1	1.07	1.11	1.15	1.3
141	0	3	7	1.03	1.03	1.11	1.3
141	0	4	2	1.07	0.97	1.11	1.3
141	0	4	5	1.5	1.03	1.11	1.3
141	0	6	1	1.25	1.03	1.11	1.3
141	0	6	5	0.91	1.0	1.11	1.2125
141	0	6	12	1.2	1.03	1.11	1.2
141	0	7	2	1.25	1.0	1.11	1.2
141	1	1	12	1.11	1.07	1.11	1.25
141	1	2	7	1.11	1.07	1.11	1.2
141	1	3	1	1.07	1.07	1.11	1.2
141	1	3	3	1.2	1.07	1.11	1.3
141	1	3	6	0.91	0.97	1.11	1.2
141	1	4	3	1.07	1.03	1.11	1.2
141	1	4	7	1.03	0.97	1.11	1.2

Figure 5.1: Traffic Data with percentiles calculation

```
traffic_regresion_filtered = traffic_data_perc.filter(
    (col('segment_id').between(start_segment, end_segment)) &
    (col('hour') == hour) &
    (col('day_of_week') == day_of_week) &
    (col('month') == month)
)
```

The resulting dataframe looks like in figure 5.2.

segment_id	hour	day_of_week	month	estimated_time	perc %25	perc %50	perc %75
142	17	6	3	1.43	1.11	1.25	1.67
144	17	6	3	1.25	1.11	1.25	1.36
143	17	6	3	1.25	1.11	1.25	1.43
145	17	6	3	2.14	1.07	1.2	1.3
141	17	6	3	1.5	1.03	1.11	1.2
147	17	6	3	1.3	1.03	1.15	1.2125
146	17	6	3	0.94	1.03	1.15	1.3

Figure 5.2: Traffic Data filtered

5.4 Preparing Data for Geographic Visualization

This step involves preparing the data for geographic visualization using Folium. We start by aggregating relevant geographical information for each segment in the traffic_data DataFrame, including start and end latitude/longitude coordinates. The resulting DataFrame, named segment_data, serves as a reference for geographic details.

```
segment_data = traffic_data.groupBy('segment_id').agg(
    first('start_latitude').alias('start_latitude'),
    first('start_longitude').alias('start_longitude'),
    first('end_latitude').alias('end_latitude'),
    first('end_longitude').alias('end_longitude')
)
```

The geolocalization segment dataframe looks like in figure 5.3.

segment_id	start_latitude	start_longitude	end_latitude	end_longitude
142	41.8013192816	-87.6649598466	41.7940254304	-87.6647651619
143	41.8086055077	-87.6651559056	41.8013192816	-87.6649598466
144	41.8158718637	-87.6653316028	41.8086055077	-87.6651559056
141	41.7940254304	-87.6647651619	41.7867479054	-87.6645778209
145	41.8231529963	-87.6654916893	41.8158718637	-87.6653316028
150	41.8668625554	-87.6665671071	41.8577948131	-87.6663216627
148	41.8523321221	-87.6662077421	41.8376849226	-87.6659250887
146	41.8305024208	-87.6657517025	41.8231529963	-87.6654916893
147	41.8376849226	-87.6659250887	41.8305024208	-87.6657517025
149	41.8577948131	-87.6663216627	41.8523321221	-87.6662077421

Figure 5.3: Segment Geolocalization Data

Next, the previously filtered traffic data (traffic_regresion_filtered) is joined with the segment information from segment_data. The joined DataFrame, named joined_df, consolidates both traffic and geographical data for visualization purposes.

```
joined_df = traffic_regresion_filtered.join(
    segment_data,
    on=['segment_id'],
    how = 'left'
)
```

segment_id	hour	day_of_week	month	estimated_time	perc %25	perc %50	perc %75	start_latitude	start_longitude	end_latitude	end_longitude
142	17	6	3	1.43	1.11	1.25	1.67	41.8013192816	-87.6649598466	41.7940254304	-87.6647651619
144	17	6	3	1.25	1.11	1.25	1.36	41.8158718637	-87.6653316028	41.8086055077	-87.6651559056
143	17	6	3	1.25	1.11	1.25	1.43	41.8086055077	-87.6651559056	41.8013192816	-87.6649598466
145	17	6	3	2.14	1.07	1.2	1.3	41.8231529963	-87.6654916893	41.8158718637	-87.6653316028
141	17	6	3	1.5	1.03	1.11	1.2	41.7940254304	-87.6647651619	41.7867479054	-87.6645778209
147	17	6	3	1.3	1.03	1.15	1.2125	41.8376849226	-87.6659250887	41.8305024208	-87.6657517025
146	17	6	3	0.94	1.03	1.15	1.3	41.8305024208	-87.6657517025	41.8231529963	-87.6654916893

Figure 5.4: Joined Dataframe

The joined dataframe looks like in figure 5.4.

Finally, the joined DataFrame is converted to a Pandas DataFrame (pandas_df) to facilitate seamless integration with the Folium library for interactive map generation. This process sets the stage for creating insightful visualizations that depict traffic conditions across specific segments in a geographic context.

```
pandas_df = joined_df.toPandas()
```

The pandas dataframe looks like in figure 5.5.

	segment_id	hour	day_of_week	month	estimated_time	perc %25	perc %50	perc %75	start_latitude	start_longitude	end_latitude	end_longitude
0	142	17	6	3	1.43	1.11	1.25	1.6700	41.801319	-87.664960	41.794025	-87.664765
1	144	17	6	3	1.25	1.11	1.25	1.3600	41.815872	-87.665332	41.808606	-87.665156
2	143	17	6	3	1.25	1.11	1.25	1.4300	41.808606	-87.665156	41.801319	-87.664960
3	145	17	6	3	2.14	1.07	1.20	1.3000	41.823153	-87.665492	41.815872	-87.665332
4	141	17	6	3	1.50	1.03	1.11	1.2000	41.794025	-87.664765	41.786748	-87.664578
5	147	17	6	3	1.30	1.03	1.15	1.2125	41.837685	-87.665925	41.830502	-87.665752
6	146	17	6	3	0.94	1.03	1.15	1.3000	41.830502	-87.665752	41.823153	-87.665492

Figure 5.5: Data converted into Pandas dataframe

5.5 Creating Interactive Traffic Map with Folium

In this step, we utilize the Folium library to generate an interactive map that visually represents traffic conditions across selected segments. The map is centered at latitude 41.8781 and longitude -87.6298, with an initial zoom level of 12.

```
map = folium.Map(location=[41.8781, -87.6298], zoom_start=12)
```

To populate the map, we extract coordinates for each segment from the Pandas DataFrame pandas_df. For each segment, a colored polyline is added to the

map based on the estimated time percentile in which it falls. The color scheme includes green for the lowest 25%, yellow for the next 25%, orange for the next 25%, and red for the highest 25% of estimated times.

Additionally, the polyline weight, opacity, and pop-up information (Segment ID and Estimated Time) are configured for a visually appealing and informative map.

```
line_coords = []
for index, row in pandas_df.iterrows():
    line_coords.extend([
        [row['start_latitude'], row['start_longitude']],
        [row['end_latitude'], row['end_longitude']]
    ])

for index, row in pandas_df.iterrows():
    if row['estimated_time'] <= row['perc %25']:
        color = 'green'
    elif row['estimated_time'] <= row['perc %50']:
        color = 'yellow'
    elif row['estimated_time'] <= row['perc %75']:
        color = 'orange'
    else:
        color = 'red'

    folium.PolyLine(
        [(row['start_latitude'], row['start_longitude']),
            (row['end_latitude'], row['end_longitude'])],
        color=color,
        weight=5,
        opacity=1,
        popup=f"Segment ID: {row['segment_id']},
            Estimated Time: {row['estimated_time']}"
    ).add_to(map)
```

Finally, the map's zoom is adjusted to fit all the segments within the visible area, providing an overall view of traffic patterns.

```
map.fit_bounds([(min([coord[0] for coord in line_coords]),
                    min([coord[1] for coord in line_coords])),
                (max([coord[0] for coord in line_coords]),
                    max([coord[1] for coord in line_coords]))])
```

5.6 Saving Interactive Traffic Map as HTML

In this final step, the interactive traffic map created using Folium is saved as an HTML file. The HTML file, named "Chicago_Traffic_map.html," encapsulates the interactive map with all its configured features, including colored polylines representing traffic conditions for various segments.

Additionally, the files.download function is employed to facilitate the direct download of the HTML file, allowing users to access and interact with the traffic map offline. This step enhances the usability and convenience of sharing the visualized traffic data.

```
map.save("Chicago_Traffic_map.html")
```

```
files.download("Chicago_Traffic_map.html")
```

Figure 5.6 shows the resulting Chicago map with coloured segments depending on the

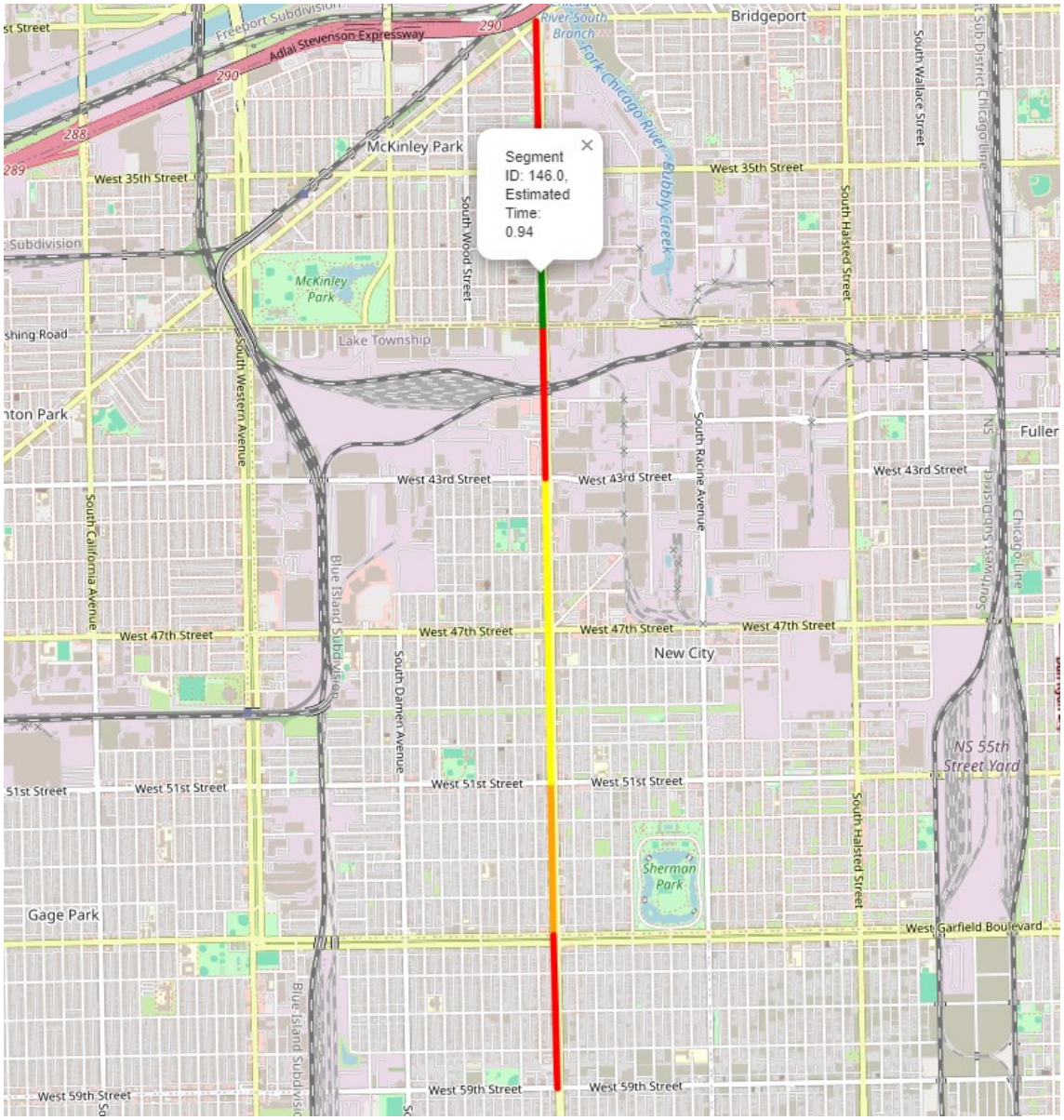

Figure 5.6: Chicago traffic map visualization by segments

Conclusions and Future Directions

Conclusions:

Model Performance: The regression model demonstrates promising performance in estimating travel times, as indicated by the Root Mean Squared Error (RMSE) and other evaluation metrics. The model effectively captures the relationships between various features and estimated travel times.

Insights into Traffic Patterns: Through exploratory data analysis and modeling, valuable insights into traffic patterns across different segments, hours, days, and months have been gained. This understanding can inform urban planning, traffic management, and infrastructure development.

Visualization Impact: The interactive traffic map provides a visually intuitive representation of traffic conditions. It offers users a practical tool for assessing estimated travel times and making informed decisions regarding route selection.

Future Directions:

Incorporation of Additional Features: Consider incorporating additional features such as weather conditions, special events, or real-time traffic updates for a more comprehensive model.

Dynamic Map Features: Enhance the interactive map with real-time updates and dynamic features, enabling users to receive the latest information on traffic conditions.

Visualization Impact: The interactive traffic map provides a visually intuitive representation of traffic conditions. It offers users a practical tool for assessing estimated travel times and making informed decisions regarding route selection.

By pursuing these future directions, the study can contribute to the continuous improvement of traffic prediction models and the development of innovative solutions for addressing urban mobility challenges.

Bibliography

[1] https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Historical-Congestion-Esti/sxs8-h27x

[2] https://research.google.com/colaboratory/faq.html

[3] https://spark.apache.org/docs/2.4.0/api/python/pyspark.sql.html

[4] https://spark.apache.org/docs/latest/ml-guide.html

[5] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.regression.Linear

[6] https://matplotlib.org/stable/users/

[7] https://seaborn.pydata.org/

[8] https://pandas.pydata.org/docs/

[9] https://python-visualization.github.io/folium/latest/user_guide.html

[10] https://github.com/NHaba/ChicagoTrafficData_TFM.git