DATA ANALYSIS : Taxi Passenger Travel Spatial and Temporal Characteristics Analysis and Application Based on Ride sourcing Data

A PROJECT REPORT

Submitted by

JagadishKumar .N[REGISTERNO:211417104088]

JoshuaDhanraj.N [REGISTER NO:211417104100]

EharrshaNarayanaa.M [REGISTER NO:211417104]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025
APRIL 2021

BONAFIDE CERTIFICATE

Certified that this project report **"DATA ANALYSIS : Taxi Passenger Travel Spatial and Temporal Characteristics Analysis and Application Based on Ride sourcing Data"** is the bonafide work of **" JAGADISHKUMAR N (2017PECCS404),JOSHUADHANRAJ N(2017PECCS409),EHARRSHANARAYANAA (2017PECCS386) "** who carried out the project work under my supervision.

[illegible] 17/9/2020

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

[illegible]

SIGNATURE

Mr. M.MOHAN,M.Tech.,(Ph.D.),
ASSISTANT PROFESSOR(GRADE-I),
DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University

Project Viva-Voce Examination held on 22.09.2020

[illegible]

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR,M.E.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mr.M.Mohan,M.Tech.,(Ph.D.)** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

M. ARAVINDH,

PL. ARUN PANDI,

M. BHARATHKANNAN.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

For nearly five decades, livability has been referenced as a key attribute for community and urban planning worldwide . More recently, it has been firmly placed in the global policy lexicon by its inclusion in three of the principles and commitments of the New Urban Agenda (NUA) adopted by the UN in 2016. The NUA is notable as it represents a significant international policy commitment in support of the Sustainable Development Goals (SDG), and more specifically SDG11, and what some have referred to as a pro-urban future. SDG11 sets out a goal for the international community to “make cities inclusive, safe, resilient and sustainable”. While it is clear that the authors of the NUA perceived livability as playing a role in eradicating poverty (Paragraph 14a), and as an indicator of both social inclusion and cohesion (Paragraph 40) and sustainable urban transport and transit systems (Paragraph 114) , nowhere within the NUA or supporting documents the concept of livability is defined. This is not entirely surprising. Indeed, authors have commented on the widespread use of the term, despite the ambiguity in meaning in policy documents and scholarly articles. According to Newton, livability can be defined as a set of attributes of a place, encompassing housing, neighborhood and region aspects that contribute to residents’ quality of life and well-being. A recent review of the literature on relevant indicators of livability suggests a broad range of contributory indicators across 11 policy domains (the natural environment, crime and safety, education, employment and income, health and social services, housing, leisure and culture, food and other goods, public open space, transport, social cohesion and local democracy), although the relative importance of each is unclear. Ruth and Franklin suggest that a “livable city” requires the needs of the inhabitants of the city to be aligned with “built infrastructures and ecosystems that provide the goods and services on which lives and livelihoods in the city depend.” They note that it is difficult to arrive at a genre ally acceptable definition of liveability because globalization, urbanization, new technologies and environmental constraints are impacting the expectations of the inhabitants. Notwithstanding the definitional ambiguity of the term liveability, this tension between the dynamic subjective perception of the inhabitant versus the objective reality of city infrastructure presents significant measurement challenges [5]. Policymakers, industry and scholars have all proposed a number of potential indicators to measure livability for a variety of purposes including discrimination among com petting hypotheses, structuring the understanding of issues and conceptualizing solutions, tracking performance towards goals and objectives, discriminating among alternative policies either for specific decisions or general policy directions, and informing general users (public, stakeholders, community). Indicators can vary widely in terms of geographic granularity (e.g. international, country, city, neighborhood), comprehend sivevs singular perspectives (e.g. transport, health, population cohorts), measure characteristics (e.g. objective/quantitative vs subjective/qualitative), and stakeholder (e.g. individual, industry, local government, national government) . As a result of this variability, livability indices are often difficult to compare. Due to the scale and complexity of livability measurement, it is often resource-intensive, time consuming, and costly. Unsurprisingly, there has been calls for standardization, disaggregation, accurate and timely infor mation on both liveability and sustainability. Historically, transit has been a central organising factor around which communities have been built .For over six decades, commentators have consistently noted the role of transit systems and traffic patterns in livable communities. It is unsurprising therefore that transport and transit systems features heavily and widely referenced liveability indicators and is specifically mentioned in the NUA with regards to livability. This research explores the use of Uber Estimated Time to Arrive (ETA) data as a simple real-time indicator of urban liveability. Uber is a ride-hailing service that offers peer-to-peer ride-sharing and other services to 75 million passengers in over 600 cities worldwide. Due to the nature, scale and coverage of Uber’s operations, it provides a unique real-time objective source of data on the interaction between inhabitants of a city and its infrastructure, and in particular its transport infrastructure. It enables comparison of data at multiple levels, for example cities, districts and neighborhoods, but also provides context sensitive data provid ing insights in to the impact of other factors impacting Uber drivers and trips on a daily basis including traffic incidents, weather and other events. As such, we posit that Uber data may provide a simple, rapid, low cost, time- and context sensitive indicator of urban liveability. We illustrate this with data sourced from the Uber Ride Request (URR) API for the Brazilian city of Natal. To demonstrate the viability of this premise, Uber data is explored by means of a data-driven approach mainly founded on Exploratory Data Analysis (EDA). In short, the main objective of this study is to show that Uber service inherently reacts to city features and dynamics, hence its data can be used as source for a new liveability indicator. The remainder of this paper is organized as follows. The following section discusses prominent urban liveability indi cators and extant scholarly research on Uber and using the Uber Application Programming Interface (API). Section III introduces the empirical context and an outline of the data set used in the study. Section IV presents the methodologies and preliminary findings from a basic EDA on Uber ETA data as an urban liveability indicator. This is illustrated using data from the Brazilian city of Natal and comparing the results with extant liveability research on Natal based on the Urban Life Quality Indices (ULQI). Section V discusses the contribution of the study, limitations, and avenues for future research. The paper then concludes.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

1. Real-Time Uber Data Analysis of Popular Uber Locations in Kubernetes Environment

Data is crucial in today's business and technology environment. There is a growing demand for Big Data applications to extract and evaluate information, which will provide the necessary knowledge that will help us make important rational decisions. These ideas emerged at the beginning of the 21st century, and every technological giant is now exploiting Big Data technologies. Big Data refers to huge and broad data collections that can be organized or unstructured. Big Data analytics is the method of analyzing massive data sets to highlight trends and patterns. Uber is using real-time Big Data to perfect its processes, from calculating Uber's pricing to finding the optimal positioning of taxis to maximize profits. Real-time data analysis is very challenging for the implementation because we need to process data in real-time, if we use Big Data, it is more complex than before. Implementation of real-time data analysis by Uber to identify their popular pickups would be advantageous in various ways. It will require high-performance platform to run their application. So far no research has been done on real-time analysis for identifying popular Uber locations within Big Data in a distributed environment, particularly on the Kubernetes environment. To address these issues, we have created a machine learning model with a Spark framework to identify the popular Uber locations and use this model to analyze real-time streaming Uber data and deploy this system on Google Dataproc with the different number of worker nodes with enabling Kubernetes and without Kubernetes environment. With the proposed Kubernetes environment and by increasing the worker nodes of Dataproc clusters, the performance can be significantly improved. The future development will consist of visualizing the real-time popular Uber locations on Google map.

2. An approach to predict taxi-passenger demand using quantitative histogram on Uber data

The precise prediction of the day to day and monthly transactions is of great value for companies. This information can be beneficial for the companies in analyzing their ups and downs and draw other plans. Moreover, a precise prediction method can optimize the performance of a company. The branch of analytics that deals with prediction is known as predictive analytics. This paper presents the use of data analytics in analyzing the transaction dataset provided by Uber to predict the possible outcomes and the changes to be made. The histograms and heat maps drawn provide us a clear visualization of the dataset and we must predict the rest out of it.

3. A Preliminary Exploration of Uber Data as an Indicator of Urban Liveability

Urban liveability is a key concept in the New Urban Agenda (NUA) adopted by the United Nations (UN) in 2016. The UN has recognized that effective benchmarks and monitoring mechanisms are essential for the successful implementation of the NUA. However, the timely and cost effective collection of objective international quality of life urban data remains a significant challenge. Urban liveability indexes are often complex, resource intensive and time consuming to collect, and as a result costly. At the same time, competing methodologies and agendas may result in subjective or non-comparable data. Historically, transit has been a central organizing factor around which communities have been built. This paper explores the use of Uber data as a simple real-time indicator of urban liveability. Using data from the Uber Ride Request (URR) API for the Brazilian city of Natal, our preliminary findings suggest that Uber Estimated Time to Arrive (ETA) data is strongly correlated with selected quality of life indicators at a neighborhood and region level. Furthermore, unlike other urban liveability indicators, our findings suggest that Uber ETA data is context-sensitive reflecting daily and seasonal factors thereby providing more granular insights. This preliminary study finds strong evidence that Uber data can provide a simple, comparable, low cost, international urban liveability indicator at both city and neighborhood level for urban policy setting and planning.

4. Green Cabs vs. Uber in New York City

This paper reports on the process and outcomes of big data analytics of ride records for Green cabs and Uber in the outer boroughs of New York City (NYC), USA. Uber is a new entrant to the taxi market in NYC and is rapidly eating away market share from the NYC Taxi & Limousine Commission's (NYCTLC) Yellow and Green cabs. The problem investigated revolves around where exactly Green cabs are losing market share to Uber outside Manhattan and what, if any, measures can be taken to preserve market share? Two datasets were included in the analysis including all rides of Green cabs and Uber respectively from April-September 2014 in New York excluding Manhattan and NYC's two airports. Tableau was used as the visual analytics tool, and PostgreSQL in combination with PostGIS was used as the data processing engine. Our findings show that the performance of Green cabs in isolated zip codes differ significantly, and that Uber is growing faster than Green cabs in general and especially in the areas close to Manhattan. We discuss meaningful facts from the analysis, outline actionable insights, list valuable outcomes and mention some of the study limitations.

5. Using big data to enhance the bosch production line performance: A Kaggle challenge

This paper describes our approach to the Bosch production line performance challenge run by Kaggle.com. Maximizing the production yield is at the heart of the manufacturing industry. At the Bosch assembly line, data is recorded for products as they progress through each stage. Data science methods are applied to this huge data repository consisting records of tests and measurements made for each component along the assembly line to predict internal failures. We found that it is possible to train a model that predicts which parts are most likely to fail. Thus a smarter failure detection system can be built and the parts tagged likely to fail can be salvaged to decrease operating costs and increase the profit margins.

6. Intelligent Analysis of City Residents Mobility Data for Transport Simulation

Data analysis is one of the key elements in the process of investigating of the transport processes in cities. Despite the large number of scientific papers in this area, the problem of obtaining of the urban population mobility data is not well studied. The most innovative method is the automated gathering of information by obtaining data from GPS trackers, social networks, mobile operators, etc. This paper describes a method for obtaining and processing population mobility open data taken from the Uber Movement source, as well as creating a theoretical model of population mobility based on people density distribution data. The developed method was tested on the example of the city of Amsterdam. In addition, a comparative analysis was carried out using open source data and data computed by developed method. As a result, we get a working method for obtaining population mobility data and analyzing these data.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Customers are often dissatisfied with traditional cab companies because of their high prices and long waiting time and hence can exploit new and big markets in countries like India. Can tap growing markets in suburban areas where taxi services are not available. Unlimited fleet of vehicles available. Regular Taxi service regulations are not applicable for uber. One key strategy of Uber regarding charging for fake bookings me delayed cancellation has levied against Ola in India. Now comes the deep pocketed backers that Ola has been able to excel in. One documented means of reducing these overheads is to focus the routing optimizations on the paths to popular destinations, and thus be able to shift a large volume of traffic with a small number of path switches, rather than shifting the traffic for all the prefixes. we track traffic with three distinct predictors which vary in degrees of complexity: a very simple predictor, the Last Value (LV), the classical Moving Average (MA) , and an adaptive but more complex predictor, the LpEMA (Low pass Exponential Moving Average),s. This implies that it is enough to take account of only a small fraction of the total number of destinations to control the routing of the majority of the traffic. In view of this, we drew up a practical criterion for the selection of popular destinations. This section sets out by describing the traffic engineering process. We then underline the importance of the consistency of traffic demands with the Zipf's law for the traffic engineering. Lastly, we describe the problem of selecting the popular destinations, and present our proposal.

Disadvantage:

- However, there is a lack of simple and pragmatic methods for selecting popular destinations
- Low-profit margins causes dissatisfaction among the drivers. This might lead to bad publicity, which can in turn discourage the new drivers from joining Uber.
- Uber and its customers have no bonding. Incentive remaining with Uber is low.

3.2 PROPOSED SYSTEM

We proposed that we will find the days on which each basement has more trips. Customers are often dissatisfied with traditional cab companies because of their high prices and long waiting time and hence can exploit new and big markets in countries like India. Find the days on which each basement has more number of active vehicles. Can tap growing markets in suburban areas where taxi services are not available. Estimated Time of Arrival can be reduced with rise in the number of Uber drivers which in turn will make Uber more liked by the customers and hence, the startup will get more revenue and drivers will also be profited. Based on the data, we will find the top 20 destination people travel the most, top 20 locations from where people travel the most, top 20 cities that generate high airline revenues for travel, based on booked trip count. Top 20 destination people travel the most: Based on the given data, we can find the most popular destination that people travel frequently. There are many destinations out of which we will find only first 20, based on trips booked for particular destinations. We are creating an RDD by loading a new dataset which is in HDFS. We have split each record by taking the delimiter as tab because the data is tab separated. We are creating the key-value pair, where key is the destination that is in 3rd column and the value is 1. Since we need to count the cities which are popular, we are using the reduceByKey method to count them. After counting the destinations, we are swapping the key-value pairs. The sortByKey method sorts the data with keys and false stands for descending order. Once the sorting is complete, we are considering the top 20 destinations. We can find the places from where most of the trips are undertaken, based on the booked trip count.

Advantage:

- Estimated Time of Arrival can be reduced with rise in the number of Uber drivers which in turn will make Uber more liked by the customers and hence, the startup will get more revenue and drivers will also be profited.
- Convenient system for the drivers. They can work for flexible hours and can even choose to be a part-time employee. Drivers can also reject unwanted clients.
- There are many destinations out of which we will find only first 20, based on trips booked for particular destinations.
- We are using the sortByKey method which sorts the data with keys where false stands for descending order.

3.3 FEASIBILITY STUDY

Feasibility study is the initial design stage of any project, which brings together the elements of knowledge that indicate if a project is possible or not. All projects are feasible if they have unlimited resources and infinite time. But the development of the software is plagued by the scarcity of resources and difficult delivery rate. It is necessary and prudent to evaluate the feasibility of the project at the earlier times.

3.3.1 TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

3.3.2 ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of drastic increase in the expenses. Python is open source and ready available for all users. Since the project is runned in python and jupyter notebook hence is cost efficient.

3.4 HARDWARE REQUIREMENTS

Processor	: Intel Pentium Dual Core 2.00GHz
Hard disk	: 40 GB
RAM	: 2 GB (minimum)

3.5 SOFTWARE REQUIREMENTS

- Jupyter Notebook
- Python 3.6.4 Version

3.6 SOFTWARE SPECIFICATION

3.6.1 MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers. Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

Here in this thesis, we are providing basic info of the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbour algorithm, decision tree learning, and deep learning.

Need of Machine Learning

Machine Learning is a field which is raised out of Artificial Intelligence (AI). Applying AI, we wanted to build better and intelligent machines. But except for few mere tasks such as finding the shortest path between point A and B, we were unable to program more complex and constantly evolving challenges. There was a realisation that the only way to be able to achieve this task was to let machine learn from itself. This sounds similar to a child learning from its self. So, machine learning was developed as a new capability for computers. And now machine learning is present in so many segments of technology, that we don’t even realise it while using it.

Finding patterns in data on planet earth is possible only for human brains. The data being very massive, the time taken to compute is increased, and this is where Machine Learning comes into action, to help people with large data in minimum time.

If big data and cloud computing are gaining importance for their contributions, machine learning as technology helps analyse those big chunks of data, easing the task of data scientists in an automated process and gaining equal importance and recognition.

The techniques we use for data mining have been around for many years, but they were not effective as they did not have the competitive power to run the algorithms. If you run deep learning with access to better data, the output we get will lead to dramatic breakthroughs which is machine learning.

3.6.2 Tools used

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object sophisticated (broadcasting) functions tools for integrating C/C++ and Fortran code useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project.

MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL),designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

3.6.3 SUPERVISED LEARNING

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X) . The goal is to approximate the mapping function so well that when you have new input data .(x) that you can predict the output variables (Y) for that data. Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Trees and support vector machines. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics. Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification.

3.6.4 UNSUPERVISED LEARNING

Unsupervised learning is where we only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there are no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data. Unsupervised learning problems can be further grouped into clustering and association problems.

3.6.5 Data Analysis

Data analysis is defined as a process of cleaning, transforming, and modeling data to discover useful information for business decision-making. The purpose of Data Analysis is to extract useful information from data and taking the decision based upon the data analysis. A simple example of Data analysis is whenever we take any decision in our day-to-day life is by thinking about what happened last time or what will happen by choosing that particular decision. This is nothing but analyzing our past or future and making decisions based on it. For that, we gather memories of our past or dreams of our future. So that is nothing but data analysis. Now same thing analyst does for business purposes, is called Data Analysis.

Types of Data Analysis: Techniques and Methods

There are several **types of Data Analysis** techniques that exist based on business and technology. However, the major Data Analysis methods are:

- Text Analysis
- Statistical Analysis
- Diagnostic Analysis
- Predictive Analysis
- Prescriptive Analysis

Text Analysis

Text Analysis is also referred to as Data Mining. It is one of the methods of data analysis to discover a pattern in large data sets using databases or data mining tools. It used to transformraw data into business information. Business Intelligence tools are present in the market which is used to take strategic business decisions. Overall it offers a way to extract and examine data and deriving patterns and finally interpretation of the data.

Statistical Analysis

Statistical Analysis shows "What happen?" by using past data in the form of dashboards. Statistical Analysis includes collection, Analysis, interpretation, presentation, and modelling of data. It analyses a set of data or a sample of data. There are two categories of this type of Analysis - Descriptive Analysis and Inferential Analysis.

Descriptive Analysis

Analyses complete data or a sample of summarized numerical data. It shows mean and deviation for continuous data whereas percentage and frequency for categorical data.

Inferential Analysis

Analyses sample from complete data. In this type of Analysis, you can find different conclusions from the same data by selecting different samples.

Diagnostic Analysis

Diagnostic Analysis shows "Why did it happen?" by finding the cause from the insight found in Statistical Analysis. This Analysis is useful to identify behavior patterns of data. If a new problem arrives in your business process, then you can look into this Analysis to find similar patterns of that problem. And it may have chances to use similar prescriptions for the new problems.

Predictive Analysis

Predictive Analysis shows "what is likely to happen" by using previous data. The simplest data analysis example is like if last year I bought two dresses based on my savings and if this year my salary is increasing double then I can buy four dresses. But of course it's not easy like this because you have to think about other circumstances like chances of prices of clothes is increased this year or maybe instead of dresses you want to buy a new bike, or you need to buy a house! So here, this Analysis makes predictions about future outcomes based on current or past data. Forecasting is just an estimate. Its accuracy is based on how much detailed information you have and how much you dig in it.

Prescriptive Analysis

Prescriptive Analysis combines the insight from all previous Analysis to determine which action to take in a current problem or decision. Most data-driven companies are utilizing Prescriptive Analysis because predictive and descriptive Analysis is not enough to improve data performance. Based on current situations and problems, they analyse the data and make decisions.

3.6.6 Data Analysis Process

The **Data Analysis Process** is nothing but gathering information by using a proper application or tool which allows you to explore the data and find a pattern in it. Based on that information and data, you can make decisions, or you can get ultimate conclusions.

Data Analysis consists of the following phases:

- Data Requirement Gathering
- Data Collection
- Data Cleaning
- Data Analysis
- Data Interpretation
- Data Visualization

Data Requirement Gathering

First of all, you have to think about why do you want to do this data analysis? All you need to find out the purpose or aim of doing the Analysis of data. You have to decide which type of data analysis you wanted to do! In this phase, you have to decide what to analyse and how to measure it, you have to understand why you are investigating and what measures you have to use to do this Analysis.

Data Collection

After requirement gathering, you will get a clear idea about what things you have to measure and what should be your findings. Now it's time to collect your data based on requirements. Once you collect your data, remember that the collected data must be processed or organized for Analysis. As you collected data from various sources, you must have to keep a log with a collection date and source of the data.

Data Cleaning

Now whatever data is collected may not be useful or irrelevant to your aim of Analysis, hence it should be cleaned. The data which is collected may contain duplicate records, white spaces or errors. The data should be cleaned and error free. This phase must be done before Analysis because based on data cleaning; your output of Analysis will be closer to your expected outcome.

Data Analysis

Once the data is collected, cleaned, and processed, it is ready for Analysis. As you manipulate data, you may find you have the exact information you need, or you might need to collect more data. During this phase, you can use data analysis tools and software which will help you to understand, interpret, and derive conclusions based on the requirements.

Data Interpretation

After analysing your data, it's finally time to interpret your results. You can choose the way to express or communicate your data analysis either you can use simply in words or maybe a table or chart. Then use the results of your data analysis process to decide your best course of action.

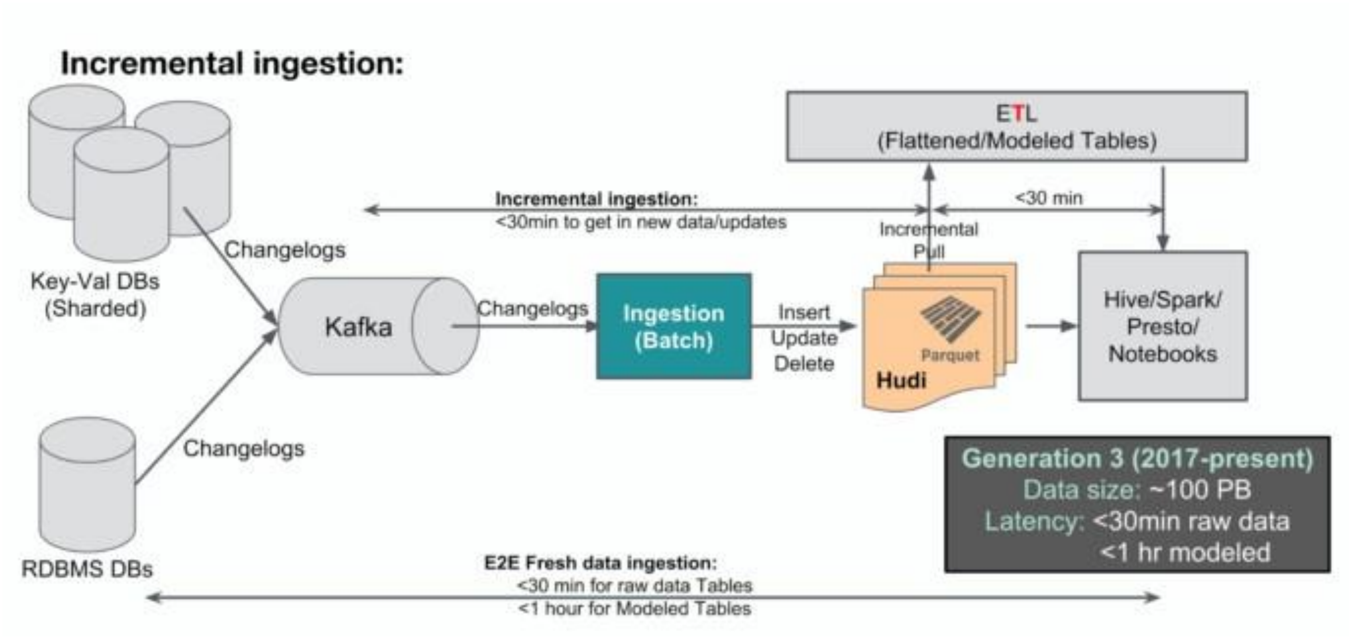
Data Visualization

Data visualization is very common in your day to day life; they often appear in the form of charts and graphs. In other words, data shown graphically so that it will be easier for the human brain to understand and process it. Data visualization often used to discover unknown facts and trends. By observing relationships and comparing datasets, you can find a way to find out meaningful information.

CHAPTER 4

ARCHITECTURE

4.1 SYSTEM ARCHITECTURE



By early 2017, our Big Data platform was used by engineering and operations teams across the company, enabling them to access new and historical data all in one place. Users could easily access data in Hive, Presto, Spark, Vertica, Notebook, and more warehouse options all through a single UI portal tailored to their needs. With over 100 petabytes of data in HDFS, 100,000 vcores in our compute cluster, 100,000 Presto queries per day, 10,000 Spark jobs per day, and 20,000 Hive queries per day, our Hadoop analytics architecture was hitting scalability limitations and many services were affected by high data latency.

Fortunately, since our underlying infrastructure was horizontally scalable to address the immediate business needs, we had enough time to study our data content, data access patterns, and user-specific requirements to identify the most pressing concerns before building the next generation. Our research revealed four main pain points:

1. **HDFS scalability limitation:** This issue is faced by many companies who rely on HDFS to scale their big data infrastructures. By design, HDFS is bottlenecked by its NameNode capacity, so that storing large numbers of small files can significantly affect performance. This limitation usually occurs when data size grows beyond ten petabytes and becomes a real issue beyond 50-100 petabytes. Fortunately, there are relatively straightforward solutions to scale HDFS from a few tens to a few hundreds of petabytes, for instance leveraging ViewFS and using HDFS NameNode Federation. By controlling the number of small files and moving different parts of our data to separate clusters (e.g., HBase and Yarn app logs moved into a separate HDFS cluster), we were able to mitigate this HDFS limitation.

2. **Faster data in Hadoop:** Uber's business operates in real time and as such, our services require access to data that is as fresh as possible. As a result, 24-hour data latency was way too slow for many use cases and there was huge demand for faster data delivery. Our second generation Big Data platform's snapshot-based ingestion method was inefficient and prevented us from ingesting data with lower latency. To speed up data delivery, we had to re-architect our pipeline to the incremental ingestion of only updated and new data.

3. **Support of updates and deletes in Hadoop and Parquet:** Uber's data contains a lot of updates, ranging in age from the past few days (e.g., a rider or driver-partner adjusting a recent trip fare) to a few weeks (e.g., a rider rating their last trip the next time they take a new trip) or even a few months (e.g., backfilling or adjusting past data due to a business need). With snapshot-based ingestion of data, we ingest a fresh copy of the source data every 24 hours. In other words, we ingest all updates at one time, once per day. However, with the need for fresher data and incremental ingestion, our solution must be able to support update and delete operations for existing data. However, since our Big

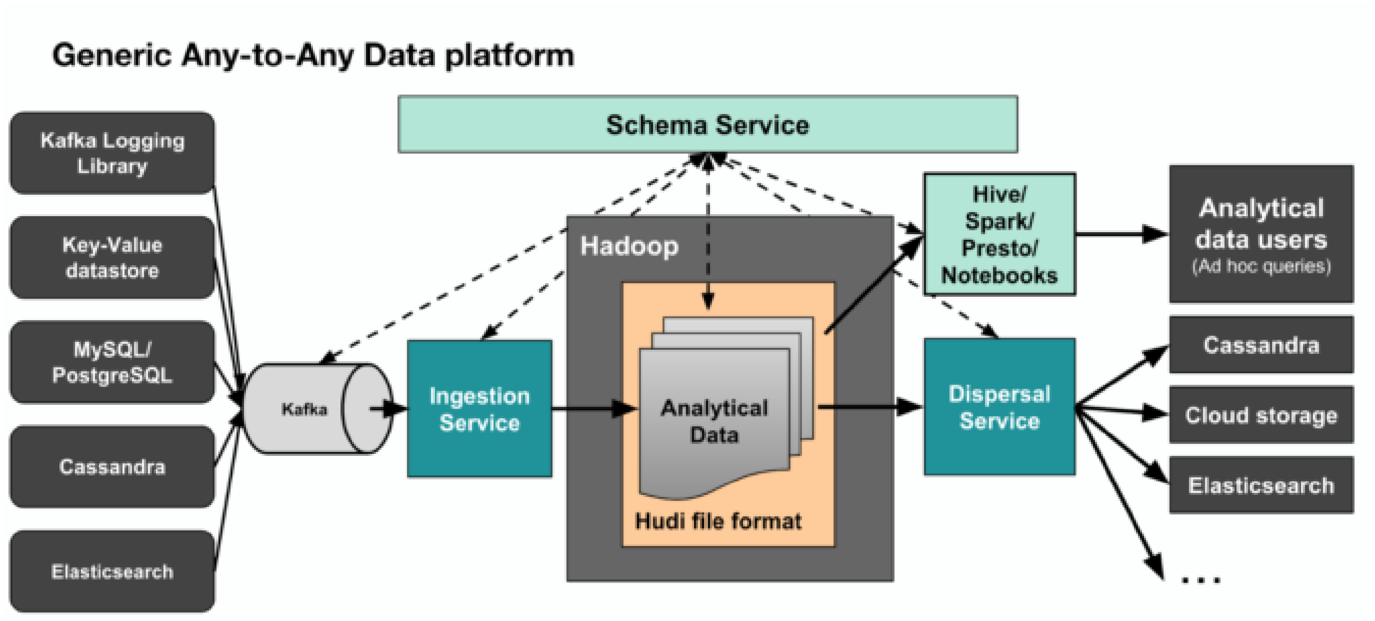
Data is stored in HDFS and Parquet, it is not possible to directly support update operations on the existing data. On the other hand, our data contains extremely wide tables (around 1,000 columns per table) with five or more levels of nesting while user queries usually only touch a few of these columns, preventing us from using non-columnar formats in a cost-efficient way. To prepare our Big Data platform for long-term growth, we had to find a way to solve this limitation within our HDFS file system so that we can support update/delete operations too.

4. **Faster ETL and modeling:** Similar to raw data ingestion, ETL and modeling jobs were snapshot-based, requiring our platform to rebuild derived tables in every run. To reduce data latency for modeled tables, ETL jobs also needed to become incremental. This required the ETL jobs to incrementally pull out only the changed data from the raw source table and update the previous derived output table instead of rebuilding the entire output table every few hours.

Introducing Hudi

With the above requirements in mind, we built Hadoop Upserts anD Incremental (Hudi), an open source Spark library that provides an abstraction layer on top of HDFS and Parquet to support the required update and delete operations. Hudi can be used from any Spark job, is horizontally scalable, and only relies on HDFS to operate. As a result, any Big Data platform that needs to support update/delete operations for the historical data can leverage Hudi.

Hudi enables us to update, insert, and delete existing Parquet data in Hadoop. Moreover, Hudi allows data users to incrementally pull out only changed data, significantly improving query efficiency and allowing for incremental updates ofderived modeled tables.



Raw data in our Hadoop ecosystem is partitioned based on time and any of the old partitions can potentially receive updates at a later time. Thus, for a data user or an ETL job relying on these raw source data tables, the only way to know what date partition contains updated data is to scan the entire source table and filter out records based on some known notion of time. This results in a computationally expensive query requiring a full source table scan and prevents ETL jobs from running very frequently.

With Hudi, users can simply pass on their last checkpoint timestamp and retrieve all the records that have been updated since, regardless of whether these updates are new records added to recent date partitions or updates to older data (e.g., a new trip happening today versus an updated trip from 6 months ago), without running an expensive query that scans the entire source table.

Using the Hudi library, we were able to move away from the snapshot-based ingestion of raw data to an incremental ingestion model that enables us to reduce data latency from 24 hours to less than one hour.

Standardized data model

In addition to providing different views of the same table, we also standardized our data model to provide two types of tables for all raw Hadoop data:

1. **Changelog history table**. Contains the history of all changelogs received for a specific upstream table. This table enables users to scan through the history of changes for a given table and can be merged per key to provide the latest value for each row.

2. **Merged snapshot table.** Houses the latest merged view of the upstream tables. This table contains the compacted merged view of all the historical changelogs received per key.

However, the stream of changelogs may or may not contain the entire row (all columns) for a given key. While merged snapshot tables always provide all the columns for a specific key, the changelog history table may be sparse if the upstream stream of changelogs only provides partial row changelogs, a functionality that improves efficiency by avoiding resending the entire row when only one or a few limited column values are changed. Should users want to fetch the changed values from the changelog history table and join it against the merged snapshot table to create the full row of data, we also include the date partition of the same key from the merged snapshot table in the changelog history table. This allows the two tables to more efficiently join across a specific partition by avoiding a full table scan of the merged snapshot table when joining the two.

4.2 UML Diagrams

UML stands for Unified Modeling Language. It’s a rich language to model software solutions, application structures, system behavior and business processes. There are 14 UML diagram types to help you model these behaviors. Unified Modeling Language™ (UML®) is a standard visual modeling language intended to be used for

- modeling business and similar processes,
- analysis, design, and implementation of software-based systems
- UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.
- UML 1.4.2 Specification explained that process:
- provides guidance as to the order of a team’s activities,
- specifies what artifacts should be developed,
- directs the tasks of individual developers and the team as a whole, and
- offers criteria for monitoring and measuring a project’s products and activities.

UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).UML is not complete and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

Name of an abstract classifier is shown in italics while final classifier has no specific graphical notation, so there is no way to determine whether classifier is final or not from the diagram.

List of UML Diagram Types

So, what are the different UML diagram types? There are two main categories; structure diagrams and behavioral diagrams. Click on the links to learn more about a specific diagram type.

Structure Diagrams

Structure diagrams show the things in the modeled system. In a more technical term, they show different objects in a system. Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

Class Diagram

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. The image below shows a component diagram.

Deployment Diagram

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.

Package Diagram

As the name suggests, a package diagram shows the dependencies between different packages in a system. Check out this wiki article to learn more about the dependencies and elements found in package diagrams.

Composite Structure Diagram

Composite structure diagrams are used to show the internal structure of a class. For a detailed explanation of composite structure diagrams, click here.

Use Case Diagram

As the most known diagram type of the behavioral UML diagrams, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system. You can create use case diagrams using our tool and/or get started instantly using our use case templates.

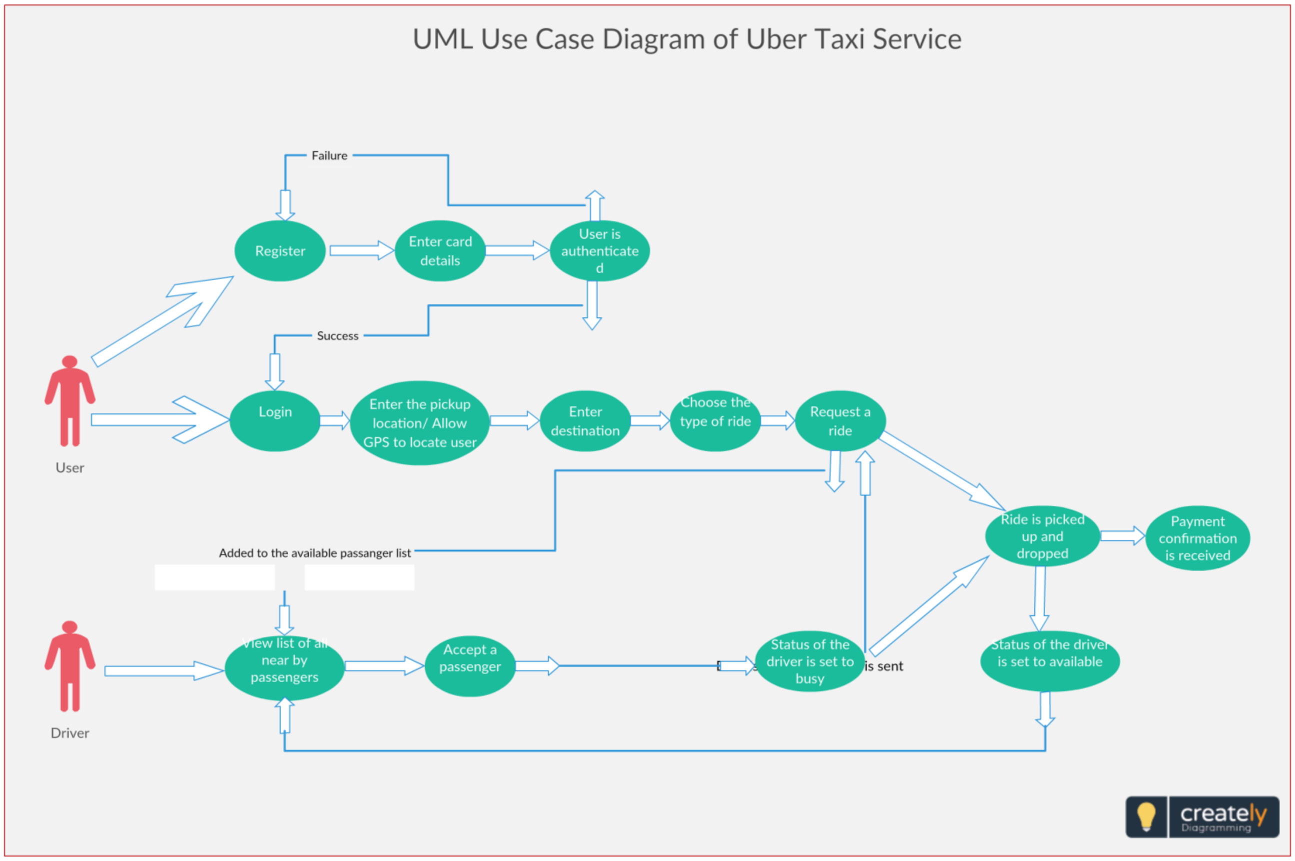
Activity Diagram

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams. Check out this wiki article to learn about symbols and usage of activity diagrams.

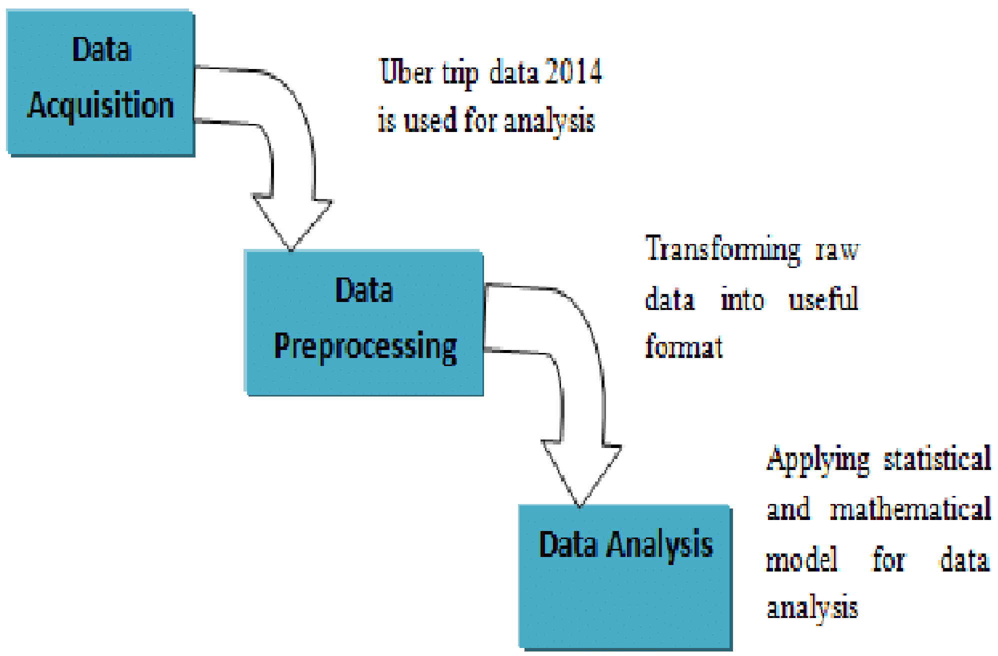
Sequence Diagram

Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows. This article explains the purpose and the basics of Sequence diagrams. Also, check out this complete Sequence Diagram Tutorial to learn more about sequence diagrams.
You can also instantly start drawing using our sequence diagram templates.

4.2.1 USE-CASE DIAGRAM

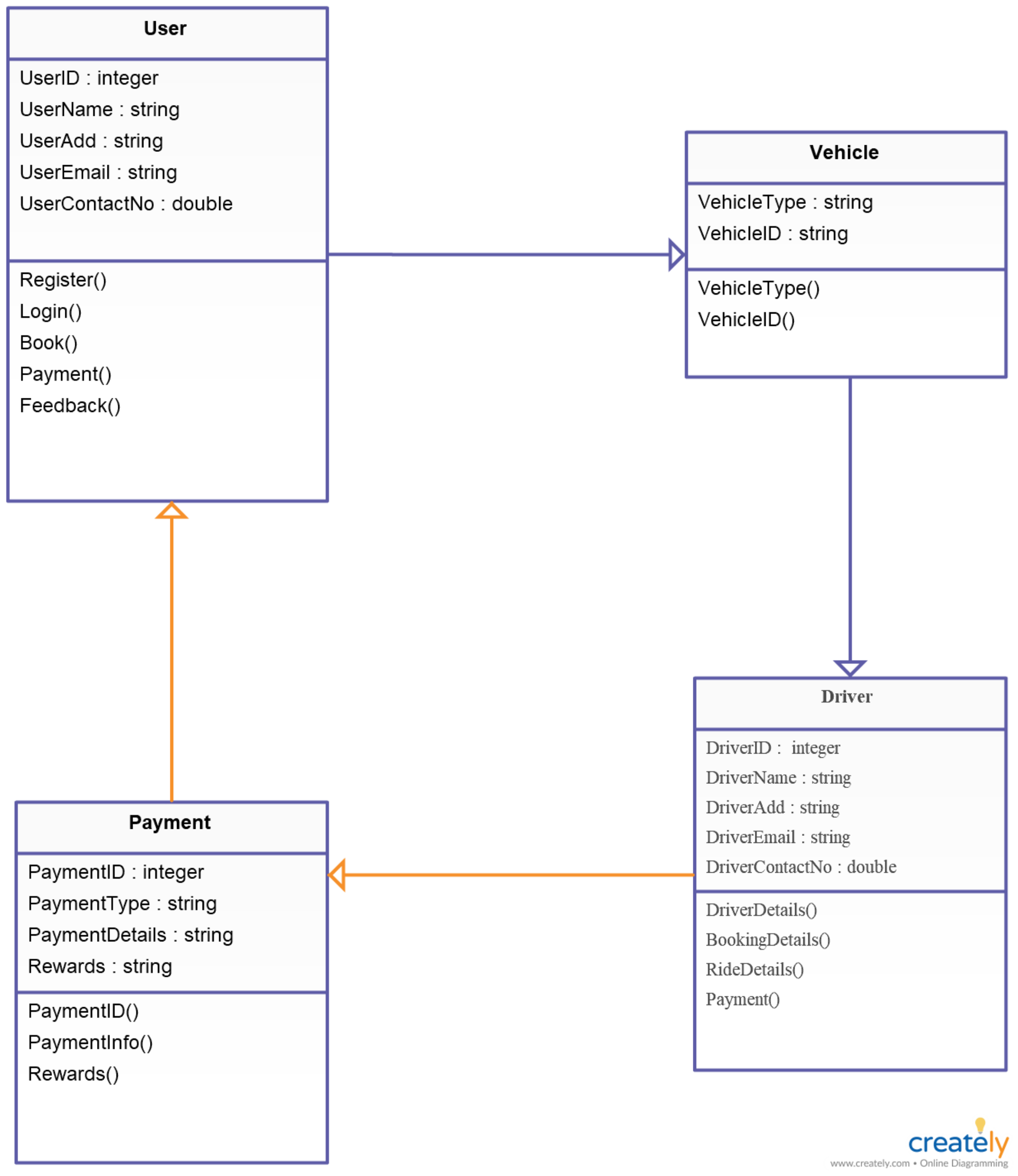

UML Use Case Diagram of Uber Taxi Service
Failure
Register
Enter card details
User is authenticated
Success
Login
Enter the pickup location/ Allow GPS to locate user
Enter destination
Choose the type of ride
Request a ride
User
Ride is picked up and dropped
Payment confirmation is received
Added to the available passanger list
View list of all near by passengers
Accept a passenger
Status of the driver is set to busy
is sent
Status of the driver is set to available
Driver
creately

4.2.2 FLOW DIAGRAM


Data Acquisition
Uber trip data 2014 is used for analysis
Data Preprocessing
Transforming raw data into useful format
Data Analysis
Applying statistical and mathematical model for data analysis

4.2.3 CLASS DIAGRAM

Class Diagram



CHAPTER 5

APPENDICES

A.1 CODING

IN[] : %pylab inline

```
 import pandas
 import seaborn
```

IN[] : data = pandas.read_csv('Desktop/uber-raw-data-apr14.csv')

IN[] : data.tail()

IN[] : data['Date/Time'] = data['Date/Time'].map(pandas.to_datetime)

IN[] : data.tail()

IN[] : def get_dom(dt):

```
 return dt.day
data['dom'] = data['Date/Time'].map(get_dom)
```

IN[] : data.tail()

IN[] : def get_weekday(dt):

```
return dt.weekday()


data['weekday'] = data['Date/Time'].map(get_weekday)


def get_hour(dt):
 return dt.hour


data['hour'] = data['Date/Time'].map(get_hour)


data.tail()
```

IN[] : hist(data.dom, bins=30, rwidth=.8, range=(0.5, 30.5))

```
xlabel('date of the month')
ylabel('frequency')
```

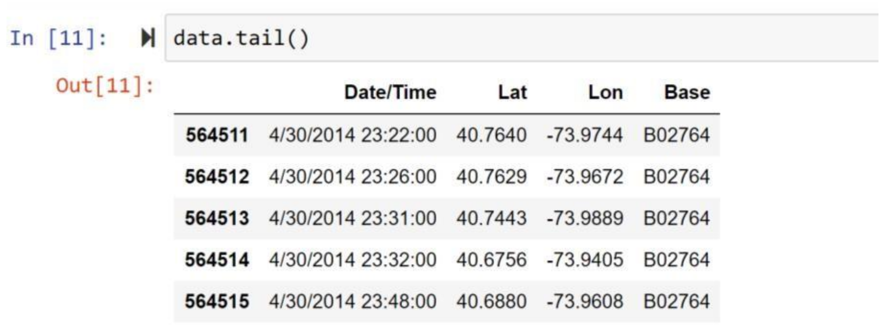
```
            title('Frequency by DoM - uber - Apr 2014')
IN[] : #for k, rows in data.groupby('dom'):
            #print((k, len(rows)))

            def count_rows(rows):
            return len(rows)

            by_date = data.groupby('dom').apply(count_rows)
            by_date
IN[] : bar(range(1, 31), by_date)
IN[] : by_date_sorted = by_date.sort_values()
            by_date_sorted
IN[] : bar(range(1, 31), by_date_sorted)
            xticks(range(1,31), by_date_sorted.index)
            xlabel('date of the month')
            ylabel('frequency')
            title('Frequency by DoM - uber - Apr 2014');
IN[] : hist(data.hour, bins=24, range=(.5, 24))
IN[] : hist(data.weekday, bins=7, range =(-.5,6.5), rwidth=.8, color='#AA6666', alpha=.4)
            xticks(range(7), 'Mon Tue Wed Thu Fri Sat Sun'.split())
IN[] : by_cross = data.groupby('weekday hour'.split()).apply(count_rows).unstack()
IN[] : seaborn.heatmap(by_cross)
IN[] : hist(data['Lat'], bins=100, range = (40.5, 41))
IN[] : hist(data['Lon'], bins=100, range = (-74.1, -73.9));
           IN[] : hist(data['Lon'], bins=100, range = (-74.1, -73.9), color='g', alpha=.5, label =
           'longitude')
```

```
          grid()
          legend(loc='upper left')
          twiny()
          hist(data['Lat'], bins=100, range = (40.5, 41), color='r', alpha=.5, label = 'latitude')
          legend(loc='best');
```

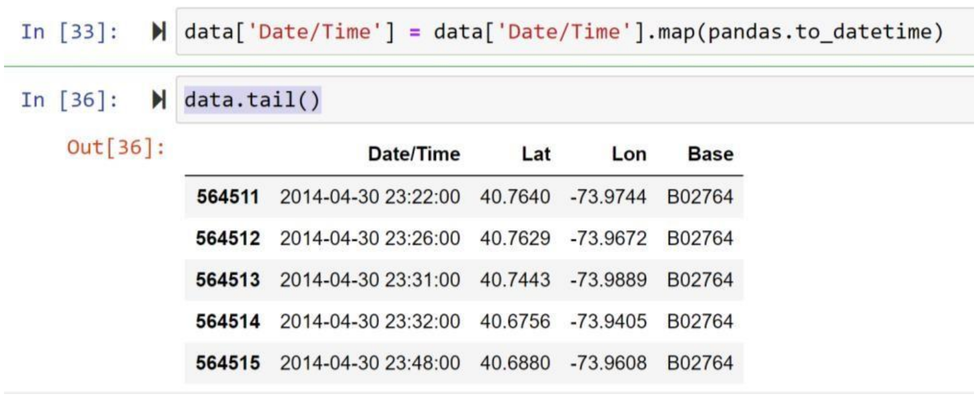
IN[] : figure(figsize=(20, 20))

```
          plot(data['Lon'], data['Lat'], '.', ms=1, alpha=.5)
          xlim(-74.2, -73.7)
          ylim(40.7, 41)
```

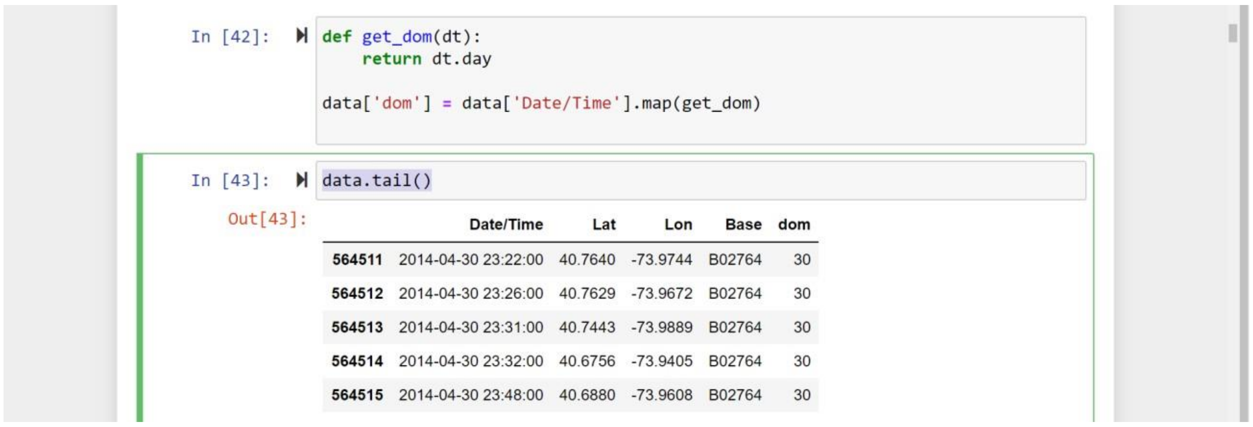
A.2 SAMPLE OUTPUT



```
In [11]:  data.tail()

Out[11]:
             Date/Time      Lat       Lon    Base
564511  4/30/2014 23:22:00  40.7640  -73.9744  B02764
564512  4/30/2014 23:26:00  40.7629  -73.9672  B02764
564513  4/30/2014 23:31:00  40.7443  -73.9889  B02764
564514  4/30/2014 23:32:00  40.6756  -73.9405  B02764
564515  4/30/2014 23:48:00  40.6880  -73.9608  B02764
```



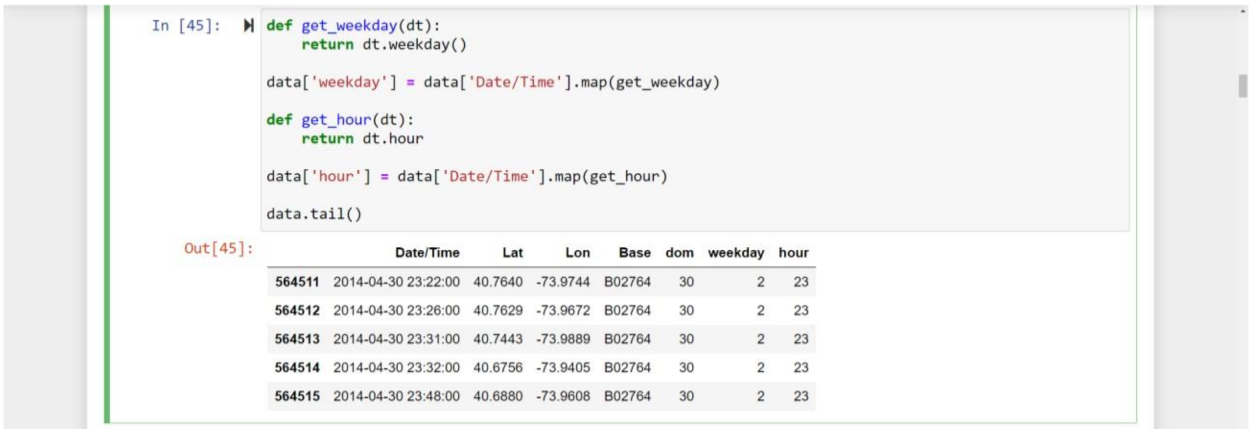
```
In [33]:  data['Date/Time'] = data['Date/Time'].map(pandas.to_datetime)

In [36]:  data.tail()

Out[36]:
             Date/Time       Lat       Lon    Base
564511  2014-04-30 23:22:00  40.7640  -73.9744  B02764
564512  2014-04-30 23:26:00  40.7629  -73.9672  B02764
564513  2014-04-30 23:31:00  40.7443  -73.9889  B02764
564514  2014-04-30 23:32:00  40.6756  -73.9405  B02764
564515  2014-04-30 23:48:00  40.6880  -73.9608  B02764
```



```
In [42]:  def get_dom(dt):
              return dt.day

          data['dom'] = data['Date/Time'].map(get_dom)

In [43]:  data.tail()

Out[43]:
             Date/Time       Lat       Lon    Base   dom
564511  2014-04-30 23:22:00  40.7640  -73.9744  B02764  30
564512  2014-04-30 23:26:00  40.7629  -73.9672  B02764  30
564513  2014-04-30 23:31:00  40.7443  -73.9889  B02764  30
564514  2014-04-30 23:32:00  40.6756  -73.9405  B02764  30
564515  2014-04-30 23:48:00  40.6880  -73.9608  B02764  30
```

IN[] : def get_weekday(dt):

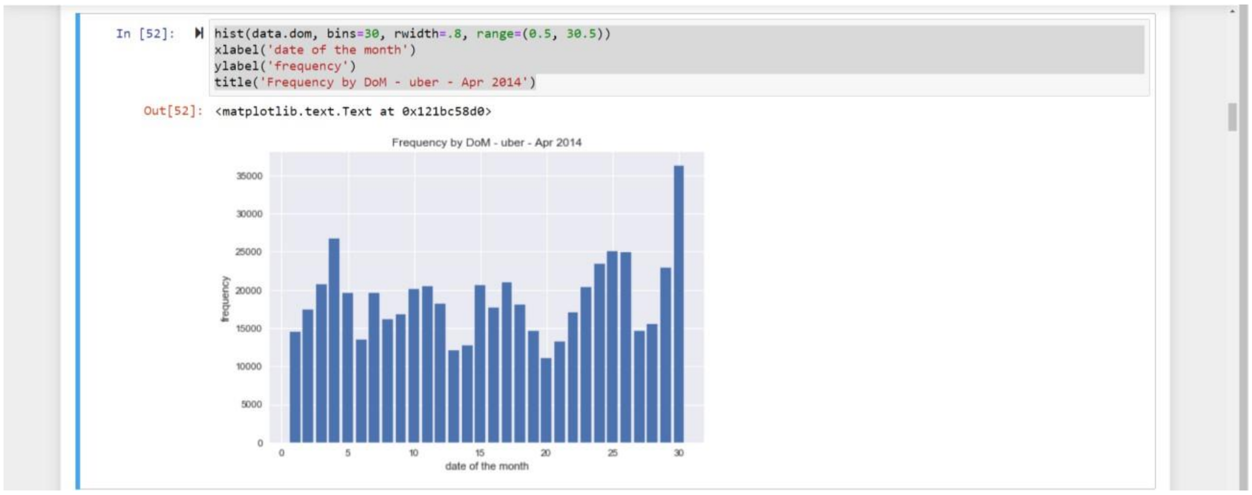
```
    return dt.weekday()
data['weekday'] = data['Date/Time'].map(get_weekday)
def get_hour(dt):
    return dt.hour
data['hour'] = data['Date/Time'].map(get_hour)
data.tail()
```



```
In [45]:  def get_weekday(dt):
              return dt.weekday()

          data['weekday'] = data['Date/Time'].map(get_weekday)

          def get_hour(dt):
              return dt.hour

          data['hour'] = data['Date/Time'].map(get_hour)

          data.tail()
```

Out[45]:

	Date/Time	Lat	Lon	Base	dom	weekday	hour
564511	2014-04-30 23:22:00	40.7640	-73.9744	B02764	30	2	23
564512	2014-04-30 23:26:00	40.7629	-73.9672	B02764	30	2	23
564513	2014-04-30 23:31:00	40.7443	-73.9889	B02764	30	2	23
564514	2014-04-30 23:32:00	40.6756	-73.9405	B02764	30	2	23
564515	2014-04-30 23:48:00	40.6880	-73.9608	B02764	30	2	23

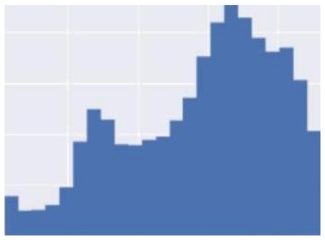


```
In [52]:  hist(data.dom, bins=30, rwidth=.8, range=(0.5, 30.5))
          xlabel('date of the month')
          ylabel('frequency')
          title('Frequency by DoM - uber - Apr 2014')

Out[52]: <matplotlib.text.Text at 0x121bc58d0>
```

```
            return len(ro a)
```

```
Nbar-riii:•:,'1).by_da* e_sarIed)
    x1abe1(' iI'' • *'i•' i '!I ')
```

an alyze th e hour



analyze the weekday

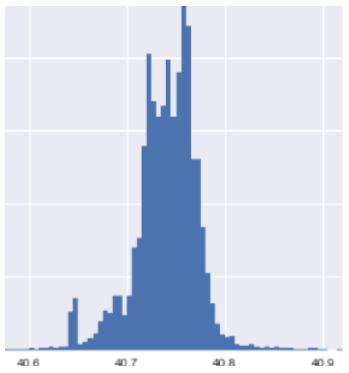
cross analys is (hour, dow)

```
N b c ro ss = d at a rou b ' ee k d ' n . s l it a 1 c ount rows . un st ac k
```

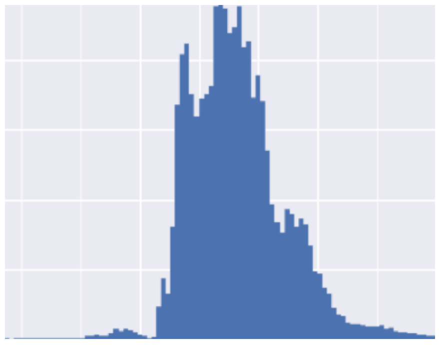
```
Out[89]: <matplotlib.axes._subplots.AxesSubplot at 0x1215bfc50>
```

by lat and Ion

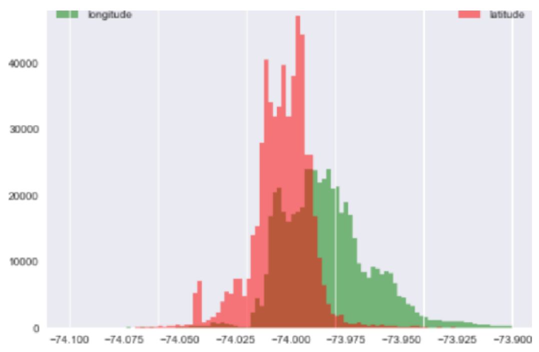
```
n [ ]: H hi st (data [ ' Let ' , bi ns= ra nge = (80. 41 )
°› fi- '
```

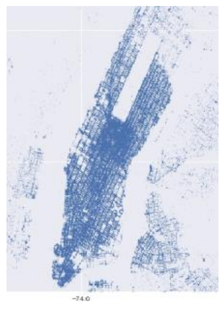


```
[",8 ! N his (data [ ' La n ' ] , bi ns=1ee, ra nge = ( ra . 1, - . 9) ) ;
```



```
In [108]: N hist (data[' Lan'] , bins=100,            = (-74.I, -73. 9), color=' g' , alpha= . 5, label = ' longitude ' )
              legend (1oc= ' uppe r left ' )

              hist(data['Lat'], bius=100,        =        41), color='r', alpha=.5, label =
              legend (loc= ' best ' )
```





CHAPTER 6

REFERENCES

[1] M. Lowe, C. Whitzman, H. Badland, M. Davern, L. Aye, D. Hes, I. Butterworth, and B. Giles-Corti, “Planning healthy, liveable and sustainable cities: how can indicators inform policy?” Urban Policy and Research, vol. 33, no. 2, pp. 131–144, 2015.

[2] The Economist Intelligence Unit, “Global liveability index 2018,” 2018.

[3] United Nations, “New urban agenda,” 2016.

[4] C. Barnett and S. Parnell, “Ideas, implementation and indicators: epis□temologies of the post-2015 urban agenda,” Environment and Urbaniza□tion, vol. 28, no. 1, pp. 87–98, 2016.

[5] F. Caprotti, R. Cowley, A. Datta, V. C. Broto, E. Gao, L. Georgeson, C. Herrick, N. Odendaal, and S. Joss, “The new urban agenda: key opportunities and challenges for policy and practice,” Urban research & practice, vol. 10, no. 3, pp. 367–378, 2017.

[6] United Nations, “Glossary of the habitat iii,” 2016.

[7] P. W. Newton, “Liveable and sustainable? socio-technical challenges for twenty-first-century cities,” Journal of Urban Technology, vol. 19, no. 1, pp. 81–102, 2012.

[8] M. Ruth and R. S. Franklin, “Livability for all? conceptual limits and practical implications,” Applied Geography, vol. 49, pp. 18–23, 2014.

[9] H. J. Miller, F. Witlox, and C. P. Tribby, “Developing context-sensitive livability indicators for transportation planning: a measurement frame□work,” Journal of Transport Geography, vol. 26, pp. 51–64, 2013.

[10] A. Ley and P. Newton, “Creating and sustaining liveable cities,” in Developing living cities: From analysis to action. World Scientific, 2010, pp. 191–229.

[11] R. Cervero, Transit-oriented development in the United States: Experi ences, challenges, and prospects. Transportation Research Board, 2004, vol. 102.

[12] D. Sauter and M. Huettenmoser, “Liveable streets and social inclusion,” Urban Design International, vol. 13, no. 2, pp. 67–79, 2008. [13] Mercer, “Quality of life rankings 2018,” 2018.