Dubai Taxi Demand Hotspots Prediction

Samer Kamal Abu Taleb
ska5805@rit.edu
DUBAI TAXI DEMAND HOTSPOTS PREDICTION

By

Samer Kamal Abu Taleb

A Capstone Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Professional Studies: Data Analytics

Department of Graduate Programs & Research

Rochester Institute of Technology
RIT Dubai
December 10, 2020
RIT

Master of Science in Professional Studies:
Data Analytics

Graduate Capstone Approval

Student Name: Samer Kamal Abu Taleb
Graduate Capstone Title: Dubai Taxi Demand Hotspots Prediction

Graduate Capstone Committee:

Name: Dr. Sanjay Modak
      Chair of committee

Name: Dr. Ehsan Warriach
      Member of committee

Date:

Date:
Acknowledgments

I would like to thank and express my deep gratitude to Dr. Ehsan, for his excellent guidance in this capstone project.
I am also thankful to my parents for their continuous support during this experiment, my friends for their assistance, and all those who have contributed to completing this capstone project directly or indirectly.
I learned many useful skills during this project, and I hope those skills will be put to good use in the years to come.
Abstract

Public transportation mode like taxi service consider as an essential service in every city that can serve all gender, age, and level of people, used for to move at any time and everywhere inside or outside the city. With the new entrance of technology for transportation, the taxi industry is undergoing a rapid digital transition like many other fields, that include new inventory like Uber and Careem for taxi sharing used by smartphones.

Most of the time, taxi vehicles' distribution is imbalanced due to passengers and taxi drivers' unorganized demand. The plan is always left to the driver to estimate the right place to drive in, making passengers waiting time is longer in some areas, and taxi drivers tour without giving exemplary service.

That will lead to loss of income for taxi service providers and reduce the service's passenger satisfaction due to long waiting time without finding the service when needed.

To solve this problem, the ability to forecast the proper place and time for taxi demand will help in solving this issue and increase income and customer satisfaction. Solving this issue will bring advantages for passengers, taxi drivers, and the service provider. Such service providers like Dubai RTA or Uber can reallocate taxi vehicles in advance to service a wider area of demand. Of course, we are not able to know where the passenger will be in a short time. However, through experience, we will know the approximate numbers of people in a particular area that require a certain number of taxis, and this is what we are looking for to reduce the waiting time.

This issue is considered the right question for an approach for competitive study and using different algorithms. Can it provide the service provider with a good view of the number of riders waiting for the taxi vehicle? Moreover, a clear idea of locating the vehicles based on passengers waiting for the service. Passenger demands can also have too irregular patterns for people to understand but can be identified by a competitive study.

**Keywords:** Taxi, Drivers, Public transportation, potential passengers, potential demand, RTA.
List of Figures

Figure 1. 1: Percentage of pickups on Dubai Communities ......................................................... 9
Figure 2. 2: Taxi Demand ........................................................................................................... 12

Figure 3. 1: Machine Learning Methods Process Flow ................................................................. 13
Figure 3. 2: Linear Regression Model ........................................................................................ 15
Figure 3. 3: KNN Regression Model .......................................................................................... 15

Figure 4. 1: Pickups by Hour – 2019 ......................................................................................... 21
Figure 4. 2: Pickups by Hour - 2018 ......................................................................................... 21
Figure 4. 3: Pickups by Day of the Week - 2019 ................................................................. 22
Figure 4. 4: Pickups by Day of the Week - 2018 ................................................................. 22
Figure 4. 5: Pickups by Month - 2019 ..................................................................................... 23
Figure 4. 6: Pickups by Month - 2018 ..................................................................................... 23
Figure 4. 7: Pickups by Community – 2018 ........................................................................... 24
Figure 4. 8: Pickups by Community - 2018 ........................................................................... 24
Figure 4. 9: Avg. Pickups and Dropoffs by Community ......................................................... 25
Figure 4. 10: Correlation Matrix ............................................................................................... 25
Figure 4. 11: $R^2$-squared values for regression models .......................................................... 26
Figure 4. 12: Time Series .......................................................................................................... 32
Figure 4. 13: Autocorrelation Plot .............................................................................................. 32
Figure 4. 14: Trend, Seasonality, and Noise ............................................................................ 33
Figure 4. 15: Observed and Forecast Values ........................................................................... 34
Figure 4. 16: Forecasting Data for three years ........................................................................ 35
Figure 4. 17: Prophet observed values ....................................................................................... 36
Figure 4. 18: Weekly and Daily Trends ..................................................................................... 37
List of Tables

Table 4.1: Data Dictionary ............................................................................................................. 18
Table 4.2: Data Sample ................................................................................................................... 19
Table 4.3: Data Statistics ................................................................................................................. 19
Table 4.4: Final Dataset .................................................................................................................. 19
Table 4.5: Data Types ...................................................................................................................... 20
Table 4.6: Feature to Predict .......................................................................................................... 27
Table 4.7: Features to Make the Prediction ..................................................................................... 27
Table 4.8: Linear Regression MAE, RMSE, and R2 ..................................................................... 28
Table 4.9: Linear Regression Actual vs Predicted Value ............................................................... 29
Table 4.10: KNN Train Data .......................................................................................................... 29
Table 4.11: KNN Actual vs Predicted Value .................................................................................. 30
Table 4.12: Linear Regression vs KNN .......................................................................................... 30
Table 4.13: Error Percentage for KNN and Linear Regression ....................................................... 31
Table 4.14: Prophet Forecast ......................................................................................................... 35
Table 4.15: Lower and Upper Bound of Prophet Forecasts ......................................................... 36
Table 4.16: ARIMA vs Prophet ..................................................................................................... 38
Chapter One - Background

1.1 Background

Rapid development in location and communications technologies, such as Wi-Fi, GPS, and smartphones, enables the theoretically or economically feasible identification and availability of large scale travel paths; trajectory data can include passenger selection paths and other travel information, such as travel time and destination, travel time and trajectory poi.

Most of the taxi vehicles have now been fitted with this technology. They are now successfully integrating these data and/or interfaces, including a rich data source, intelligent transport networks for sufficient taxi demand, time-saving track identification, fuel-saving, and taxi share.

The data of GPS is very essential on service like taxi demand, as it shows the main factors of operation that already achieve success in many literatures on many applications such as smart driving for Z. Deng and M. Ji and Y. Yue, Y. Zhuang [11] [20]. Many projects have been carried out to study traffic, traffic, and user analysis, and previously focused on stimulating transport, travel speed, and traffic using trajectory data, and the second is to analyze users' travel patterns and behaviors.

1.2 Statement of Problem

Most of the time, taxi vehicles' distribution is imbalanced due to passengers and taxi drivers' unorganized demand. The plan is always left to the driver to estimate the right place to drive in, making passengers waiting time is longer in some areas, and taxi drivers tour without exemplary service. A comparative study using different prediction algorithms will help the city re-allocate resources to meet transportation demand and eliminate available taxis on streets that waste energy and worsen traffic congestion. Predicting taxi demand in Dubai city will help coordinate the taxi dispatch center to reduce passenger's and driver's waiting times and traffic congestion, increase the level of service and profit, and reduce carbon emissions for better sustainability.
1.3 Motivation

Accurately forecasting taxi demand will lead to various advantages on many levels. By restricting the number of taxis, passengers would experience a lower estimated wait time; taxi companies would have more productive resource use. Along with a decrease in time spent roaming and queuing for customers, drivers will receive suggestions about searching for customers. Two sub-issues, short-term or real-time predictions and long-term predictions can be divided into forecasting taxi demand. Short-term forecasts influence clients and drivers on a day-to-day basis, and long-term forecasts are made to support resource management and planning on a weekly or monthly basis.

Much like many other industries, the taxi industry is in an age of digital transformation. The ability to collect and store large quantities of data generated by each taxi has increased access to free smartphones and better wireless mobile telecommunications. GPS coordinates of where the taxi was at various times if a customer occupied the car customer and occupied the car so, when and where the customer was picked up and dropped off are examples of data obtained by taxi companies.

Several studies have used this type of data and have shown that taxi demand can be predicted using various algorithms [14, 3, 12, 15]. A typical example is a study conducted by Jun Xu et al. in August 2018, which demonstrates that taxi demand can be reliably predicted using a recurrent neural network (RNN) and a network of mixture density. The recurrent architecture of the neural network used by Jun Xu et al. in their research [10] is referred to as an extended short-term memory network (LSTM), first defined in a paper by S. Hochreiter et al. In 1997 [8]. In several fields and problem domains that use sequential data, it has since gained popularity and has been shown to produce state-of-the-art results [15, 16, 3]. Networks of this type will model long-term trends in data accurately. The forgotten gate and output activation function are essential for the success of the algorithm's success in a large-scale study of different LSTM architectures. The forget gate allows the LSTM to reset its state, and to stabilize learning, the output activation function is needed [17].
The number of trainable parameters is high because of the complexity of convolutional neural networks, making it computationally costly relative to feed-forward networks such as the Convolutional Neural Network (CNN).

### 1.4 Project Goals

Within this project, a comparative study and with different algorithms for predicting taxi demand will be developed and trained for Dubai city using historical taxi ridership data of pickup count, drop off count, date, and time to meet this predicted demand. It will be analyzed using regression models to predict the number of pickups in each community in Dubai, as well as time series models to predict the future trends of Dubai Taxi pickups. A Prediction model with good accuracy and dashboards to visualize the number of taxi pickups for each community in Dubai for different times, including peak hours and off-peak hours, as a future interval of time-based on the historical data extracted to improve the standard of operation.

![Figure 1.1: Percentage of pickups on Dubai Communities](image-url)
1.5 Methodology

To achieve the set objective by predicting the number of pickups request in each community of Dubai based on community ID, date, and time, it is essential to closely follow a well-defined data mining methodology covering the process from data to knowledge. Cross-Industry Standard Process for Data Mining (CRISP-DM) is used during the comparative study using different algorithms. It also provides a logical order to complete the phases of a comparative study by using different algorithms.

Will be achieved by following these steps:

- Understand the business of taxi demand and the collected data.
- Prepare the dataset by following data processing steps.
- Visualize taxi data to find the relations between the attributes.
- Develop a regression algorithms for prediction model.
- Evaluating the model performance.
- Build Time series models and evaluate the performance.

Dubai Road and Transport authority may decide based on this project by studying the demand for taxi service and other transportation modes. Especially during the times of large events that conduct in Dubai will have a large number of gathering people and visitors like the New Year event and Expo 2020.

1.6 Limitations of the Study

The data sets used are supplied by Dubai RTA and consist of real taxi trips that have been collected over two years. Dubai’s demand distribution is greatly affected by the weather, rain, fog, and sometimes humidity. At other times, the demand changes from one community to another according to the distribution of planned events and concerts that change every year due to Dubai’s rapid development. That requires relying on other datasets to accommodate these changes, such as weather data or annual planned events.
Chapter Two - Literature Review

2.1 Taxi Origin-Destination

Paper for Lingbo Liu, Zhilin Qiu, Guanbin Li, Qing Wang, Wanli Ouyang, and Liang Lin in 2019 [4], the article suggests a demand prediction function for a taxi from origin to destination, including collecting the New York City meteorological data in each time interval. They believed that if the demand for taxi origin and goals was well estimated, they might more easily pre-allocate the taxi to satisfy customer demands. The main challenge of the proposed project includes gathering numerous qualitative space-time knowledge to understand the demand patterns. For example, some space-like regions typically have typical demand patterns, i.e., the sum of taxi demands and demand trends.

2.2 Taxi sharing

Further uses for taxi sharing and destination forecasts can be found using historical taxi data [11], a referral framework for taxis and taxi drivers. To make this suggestion by studying from GPS taxi tracks and passenger mobility trends. The plan involves a taxi ride-sharing scheme that serves taxi users' demands in real time[8] virtually. Model the passenger who finds a taxi as an MDP and recommends increasing taxi drivers' revenue performance. "On the mobile intelligence of autonomous vehicles"[19] Discusses the topic of mobility enhancement of automating vehicles by large scale data processing. A research-based on the subject [20] is referred to for a further investigation of various approaches to interpretation and learning from taxi GPS traces.

2.3 Taxi demand

Investigate the limit of predictability for taxi demand in NYC, they divide the city into zones based on large buildings and calculate three different kinds of entropy to approximate how well taxi demand can be predicted for each zone [3][4]. They split up the causes for
taxi demand into temporal and random correlation. A low unexpected correlation indicates that a pure time-series model that only takes historical demand into account can predict the future demand well. A high random correlation indicates that further information is needed. They find that the hourly limit of predictability for their small building zones is 83% on average. To predict the taxi demand, they use a hidden Markov model (HMM) and a shallow neural network. They conclude that the HMM, a pure time-series model like SARIMA, is faster and performs better than their NN in zones where the predictability is high. The NN is slower but performs better in zones with low predictability, i.e., irregular demand [21].
3.1 Machine Learning

Artificial intelligence (AI) is a field that concerns itself with building intelligent entities. There are several different approaches to achieve this mission. One example is the symbolic approach, where hard-coded rules, logic, and search algorithms are combined to solve problems. Another example is the Bayesian approach, where a probability distribution is used to reason with uncertainty. Using optimization algorithms and domain data, the most likely conditional dependencies of the variables, e.g., a probabilistic graph, can then be calculated to generate a model that can infer the probability functions of unobserved variables [1].

Machine Learning algorithms are used to teach the computer how it can teach itself without any direct coding, so the result can help in decision making that rely on the inputs features and find unique instances. As an example of machine learning, if the house has a garden or garage, you would have to do some special tests in a conventional programming environment when pricing a home. The number of specifics needed for implementation will increase rapidly in such an environment. Programming a solution that takes all the relevant inputs and special situations into account becomes complicated.

The idea of allowing a computer to learn and construct a predictive data model is machine learning called mathematical learning, which can then calculate values based on different inputs. Two main subfields of Machine learning:

- Supervised learning: the data used can include both input and output, for which a model trained to predict the outputs based on the input.
Unsupervised learning: Usually used to a large dataset, the objective is to establish some new features using that dataset, which is initially unknown [7].

### 3.2 Supervised Algorithms

A subfield of machine learning known as supervised learning is defined by how it requires labeled data, i.e., mapping the input and output data to learn patterns. Formally, given an input-output pair \( x, y \), a supervised learning model tries to learn \( f(x) = y \). Classification is one of two types of supervised learning; classifying is to assign a class to the output based on the input. Regression is the other form of supervised learning; this is when the output is a numerical value that is not a class or category. The taxi demand domain is an example of regression in that the sought output is an integer that represents a quantity.

#### I. Linear Regression model

Two problems in the general definition of regression have been discussed: (1) does a set of predictor variables perform very well in forecasting an outcome (dependent variable)? (2) How will the effects variable, as the magnitude and sign of the beta figures, be affected? These regression predictions demonstrate the relationship between one dependent variable and one or more independent variables. The formula \( y = c \times b \times x \) represents the simplest form of a regression equation, where \( y \) = predicted depending variable score, \( c \) = constant, \( b \) = coefficient of regression, and \( x \) = independent variable score the single form of a regression equation.
II. K Nearest Neighbors Regression

KNN consider as a type of lazy learning that only approximates the function locally, with all estimation delayed until the process is evaluated. Since it depends on distance for scoring, its accuracy can be significantly improved by simplifying the training data[5][6].

Figure 3. 2: Linear Regression Model

Figure 3. 3: KNN Regression Model
3.3 Time Series Analysis

The timeframe sequence of observations considers a time series, and a variation of the day, week, month, or year is the most simplistic example of a time sequence that we witness every day. ARIMA design is considered among the most typical methods in time forecasting. (p, d, q) are the ARIMA unique integers thus signify ARIMA versions that specifications together represent seasonality, trend, and noise. The ARIMA design can be used to forecast future factors in the series.

3.4 Used Tools

I. Python

In this project, we will use Python as the primary programming language, that provides a wide range of useful libraries for our models, such as:

**NumPy:** Numerical Python is a scientific computing library that provides simple mathematical features and linear algebraic.

**Pandas:** Provide easy to use data structures such as series, dataframes, and panels.

**SciPy:** When doing machine learning, this library is useful for scientific and technological computing.

**Scikit Learn:** This library is used for machine learning and deep learning, regression, classification, and clustering models.

**Matplotlib:** Used for data visualization.

**Datetime:** This library provides all the essential datetime features.

II. Microsoft Power BI

Microsoft Power BI is a platform used to visualize the data and find the needed insight. We used Power BI as a visualization tool to define some charts to illustrate our data, such as the number of pickups in months, days, and hours, and show the highest community of the number of pickups.
Chapter Four - Project Analysis

4.1 Business Understanding

Taxi demand could come from either street hailing or bookings, which are placed through a phone call or a mobile application. There are three significant differences:

- The hailing happens spontaneously while bookings are planned in advance.
- The location of taxi cars influences whether a hailing occurs or not.
- Hailing is influenced by the structure of a city’s road network.

Domain experts and scientific literature have been sources in investigating the parameters that significantly impact taxi demand for an area. Primary parameters are historical taxi demand and temporal factors, i.e., the hour of the day, day of the week, or month. Secondary factors include holidays, promotions, sporting events, special occasions, nearby public transport schedules, taxi drop-offs, weather, and closing times of events. Some of these secondary factors have in common are that they are indicators of how many people there are in an area. Works under the assumption that the number of mobile network connections is a good approximation of the number of people in an area. Google researchers managed to produce a model that accurately predicted taxi demand in Tokyo. Unfortunately, this valuable data is not widely available [7, 12, 13].

4.2 Data Understanding

I. Source of Data

Dubai Taxi pickup data has been collected from RTA - Enterprise Command and Control Center (EC3), a center related to the Road and Transport Authority of Dubai. It can control all types of transportation modes all over Dubai in one centralized center. This center has been established in the middle of 2017 to be a center of all transportation data. We can rely on the data as real data from the main center of RTA data present quantitative and time-series data.
Dubai Taxi pickups data has for each community, the number of the pickup and drop-offs are recorded, and when it occurred. The time is represented as a timestamp, which contains the year, month, day, weekday, and pickup hour. Dubai city is divided into different communities, each community named with a specific ID. The number of drop-offs for each community included supporting our model and further studies. Furthermore, all the trips during the same hour in the same zone are totaled and referred to as the community’s count. Two years of taxi trip data ranged from January 1st, 2018 to December 30th, 2019.

II. Data Dictionary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hour</td>
<td>Hour of the day.</td>
</tr>
<tr>
<td>Community ID</td>
<td>The ID of Dubai Community.</td>
</tr>
<tr>
<td>Community Name</td>
<td>The name of Dubai communities.</td>
</tr>
<tr>
<td>Day Name</td>
<td>The Day of the Week.</td>
</tr>
<tr>
<td>Date</td>
<td>Date that contain year, month, and day.</td>
</tr>
<tr>
<td>Pickups</td>
<td>Count of Pickups inside the community.</td>
</tr>
<tr>
<td>Dropoffs</td>
<td>Count of Dropoffs inside the community.</td>
</tr>
</tbody>
</table>

Table 4.1: Data Dictionary

III. Data Summary

The Taxi-pickups dataset currently contains around 3,300,000 instances of taxi pickups that took place within the city of Dubai. Each record consists of various intrinsic and contextual attributes such as hour, community name, community ID, count of pickups, name of the day, and Date.

<table>
<thead>
<tr>
<th>Hour</th>
<th>Community ID</th>
<th>Community Name</th>
<th>Day Name</th>
<th>Date</th>
<th>Pickups</th>
<th>Dropoffs</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00</td>
<td>111</td>
<td>AL CORNICHE</td>
<td>Sunday</td>
<td>1/7/2018</td>
<td>46</td>
<td>70</td>
</tr>
<tr>
<td>00:00</td>
<td>111</td>
<td>AL CORNICHE</td>
<td>Sunday</td>
<td>1/14/2018</td>
<td>37</td>
<td>115</td>
</tr>
</tbody>
</table>
Table 4.2: Data Sample

<table>
<thead>
<tr>
<th>Time</th>
<th>ID</th>
<th>Community</th>
<th>Day</th>
<th>Date</th>
<th>Pickups</th>
<th>Dropoffs</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00</td>
<td>111</td>
<td>AL CORNICHE</td>
<td>Sunday</td>
<td>1/21/2018</td>
<td>46</td>
<td>97</td>
</tr>
<tr>
<td>00:00</td>
<td>111</td>
<td>AL CORNICHE</td>
<td>Sunday</td>
<td>2/4/2018</td>
<td>47</td>
<td>94</td>
</tr>
<tr>
<td>23:00</td>
<td>991</td>
<td>HEFAIR</td>
<td>Saturday</td>
<td>9/14/2019</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Number of rows = 3294408
Number of columns = 7

Table 4.3: Data Statistics

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>mean</th>
<th>std</th>
<th>min</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pickups</td>
<td>3294408.0</td>
<td>56.053185</td>
<td>120.144978</td>
<td>0.0</td>
<td>1.0</td>
<td>10.0</td>
<td>53.0</td>
</tr>
<tr>
<td>Dropoffs</td>
<td>3294408.0</td>
<td>54.912383</td>
<td>113.570995</td>
<td>0.0</td>
<td>1.0</td>
<td>13.0</td>
<td>55.0</td>
</tr>
</tbody>
</table>

4.3 Data Preparation

Data Cleaning

As data quality is a determining factor for comparative study results, the data had to be cleaned. Ensuring that there were no duplication rows, outliers, null values, invalid community ids, or times outside the present range, a script had validated the company data before it was fed to the pre-processing step. These processes eliminate only 2017 data rows as it is not in the required range, which indicates high-quality data.

The data type has been checked for all attributes, columns Pickups, and Dropoffs changed to be an integer. Date and hour attribute changed to be DateTime type. To get a separate year, month, and day columns that will support the model, we split the date column.

Final dataset:

Table 4.4: Final Dataset

<table>
<thead>
<tr>
<th>Community ID</th>
<th>Community Name</th>
<th>pickups</th>
<th>dropoffs</th>
<th>hour</th>
<th>day</th>
<th>month</th>
<th>year</th>
<th>dayname</th>
<th>weekday</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>AL CORNICHE</td>
<td>46</td>
<td>70</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>2018</td>
<td>Sunday</td>
<td>6</td>
<td>2018-01-07</td>
</tr>
<tr>
<td>111</td>
<td>AL CORNICHE</td>
<td>37</td>
<td>115</td>
<td>0</td>
<td>14</td>
<td>1</td>
<td>2018</td>
<td>Sunday</td>
<td>6</td>
<td>2018-01-14</td>
</tr>
<tr>
<td>111</td>
<td>AL CORNICHE</td>
<td>46</td>
<td>97</td>
<td>0</td>
<td>21</td>
<td>1</td>
<td>2018</td>
<td>Sunday</td>
<td>6</td>
<td>2018-01-21</td>
</tr>
<tr>
<td>111</td>
<td>AL CORNICHE</td>
<td>47</td>
<td>94</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2018</td>
<td>Sunday</td>
<td>6</td>
<td>2018-02-04</td>
</tr>
<tr>
<td>111</td>
<td>AL CORNICHE</td>
<td>41</td>
<td>83</td>
<td>0</td>
<td>11</td>
<td>2</td>
<td>2018</td>
<td>Sunday</td>
<td>6</td>
<td>2018-02-11</td>
</tr>
<tr>
<td>Attributes</td>
<td>Data Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>communityid</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>communityname</td>
<td>object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pickups</td>
<td>int32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dropoffs</td>
<td>int32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hour</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>day</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>month</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>year</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dayname</td>
<td>object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weekday</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>datetime64[ns]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Data Types

New number of rows = 3,289,632
New number of columns = 11

4.4 Data Exploration

Visualize the Data

We use visualizations to identify and discover insights from the data by generating various types of plots and figures.

To obtain a better understanding of the taxi demand distribution based on the features, numerous graphs were made. By looking at its rolling means and standard deviations, the historical demand was investigated. Suspicions such as the demand for taxis on weekdays and weekends should be confirmed in this process. Additionally, findings indicated that holidays would be an exciting feature to add to the data set due to them causing spikes in the data. In addition to understanding the day's peak times in depth between 2018 and 2019, look at the busiest month of the year and try to find a reason for that change.
Total Pickups by Hour

The demand throughout the day for 2018 and 2019 can be seen in Figures 4.1 and 4.2. The demand has been summed up for all communities for the whole year and this total demand is displayed per hour as a fraction of the hour with the highest demand. For both years, the demand is the lowest in the middle of the night but still does not go below one million. For 2019, the demand starts increasing rapidly at 06:00 am up until its peak at 09:00 am with around 5M pickups all over Dubai, then it decreases until the evening hours to rise again after 04:00 pm to reach the peak at 06:00 pm as pm peak hours with 5.2M pickups. The demand in 2018 almost follows the same smooth curve; instead, the peak hour is 08:00 am with around 5M pickups; afterward, the demand decreases slightly and stays level until 05:00 pm when it goes up and hits its peak 06:00 pm with 5M pickups. After that it decreases until 04:00 am the next day.

Figure 4.1: Pickups by Hour – 2019

Figure 4.2: Pickups by Hour - 2018
**Total Pickups and Dropoffs by Day**

The distribution of demand is concentrated during the weekdays in the middle of the week and falls to its lowest levels on the weekend, especially Friday. In 2019, Thursday and Monday were the highest weekdays, with nearly 14 million requests, while Mondays and Tuesdays were respectively in 2018. A significant drop in the number of pickups on Friday in 2019, while Friday and Saturday are the lowest in 2018.

![Pickups and Dropoffs by Day of Week 2019](image1)

**Figure 4.3: Pickups by Day of the Week - 2019**

![Pickups and Dropoffs by Day of Week 2018](image2)

**Figure 4.4: Pickups by Day of the Week - 2018**
Total Pickups by Month

The bar charts illustrate the number of requests distributed over the months showed that the last quarter of 2019 was the highest in the number of pickups and reached 8.8 million in October. The change could explain this in the weather in these months and increase the number of tourists with the increase in the number of shows and festivals in Dubai. In contrast, it is the lowest in the month of August due to the departure of most residents outside the city on vacations with bad weather.

The results differ in 2018, as the months of September 9.2 million and March 8.7 million are the highest, and close rates in the rest of the months of the year.

Figure 4.5: Pickups by Month - 2019

Figure 4.6: Pickups by Month - 2018
**Total Pickups by Community**

The bar charts show the distribution of pickups in Dubai among the top 12 communities. With 5.8 million pickups during the year 2019, the Dubai Marina community is the highest in demand of all Dubai communities, followed by Al Muraqqabat, Burj Khalifa, Al Barsha First, and Dubai Airport. These five communities in 2019 and 2018 are the highest.

In 2018, the order was given as follows: Dubai Marina, Dubai Airport, Al Muraqqabat, Burj Khalifa, and Al Barsha First. This gives us an excellent impression to determine the number of Taxi vehicles that need to be found in these areas, in addition to trying to feed these communities with other modes of transportation.

![Figure 4.7: Pickups by Community – 2018](image1)

![Figure 4.8: Pickups by Community - 2018](image2)
This Bubble chart demonstrates the dominance of the main five pickup and drop-off communities relative to Dubai’s other communities.

Figure 4. 9: Avg. Pickups and Dropoffs by Community

The below correlation matrix showing correlation coefficients between the variables.

Figure 4. 10: Correlation Matrix
4.5 Data Modeling

I. Regression Algorithms

We explore different types of regression algorithms for comparative study and summarize our findings covering the analysis of:

- **Mean Absolute Error (MAE)** is the average error in the predicted feature \( y \). A value of 0 indicates a perfect fit.

  It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

  \[
  \text{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j|
  \]

- **Root Mean Square Error (RMSE)** is the average error in units of the predicted feature \( y \), yet more seriously penalizes larger errors than MAE. Again, 0 indicates a perfect fit. RMSE is a quadratic scoring rule that also measures the average magnitude of the error.

  \[
  \text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2}
  \]

- **R-squared (R2)** is a statistical measure of how near the fitted regression line is to the results.

  ![Plots of Observed Responses Versus Fitted Responses for Two Regression Models](image)

  Figure 4.11: R-squared values for regression models
The regression model's value on the left is 38.0% of the variance, while 87.4% for the right. The more unstable the regression model is, the closest the data points are to the fitted regression line.

- **Select Features and Split into Input and Target Features:**
  We can select pickups as the feature to predict \( y \) and community ID, dropoffs, month, year, weekday, and hour as the features to make the prediction \( x \).

<table>
<thead>
<tr>
<th></th>
<th>pickups</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>41</td>
</tr>
</tbody>
</table>

**Table 4.6: Feature to Predict**

<table>
<thead>
<tr>
<th></th>
<th>community id</th>
<th>dropoffs</th>
<th>hour</th>
<th>month</th>
<th>year</th>
<th>weekday</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>111</td>
<td>70</td>
<td>0</td>
<td>1</td>
<td>2018</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>111</td>
<td>115</td>
<td>0</td>
<td>1</td>
<td>2018</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
<td>97</td>
<td>0</td>
<td>1</td>
<td>2018</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>111</td>
<td>94</td>
<td>0</td>
<td>2</td>
<td>2018</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>83</td>
<td>0</td>
<td>2</td>
<td>2018</td>
<td>6</td>
</tr>
</tbody>
</table>

**Table 4.7: Features to Make the Prediction**

- **Scale Features**
  When we ran `taxidata.describe()` above; we saw that the range of values between the min and max was different for different features. With data on such vastly different sizes, many algorithms will not work optimally. Size the data before constructing the model to avoid this. A few methods are available. I'm going to use `MinMaxScaler()` in this case, which scales the data so that any function ranges from 0 to 1.

- **Split into Training and Test Sets**
  To validate the model, use the train test `split()` function in sklearn to separate the sample set into a training set that we will use to train the model and a test set.
  I have set 1/3 of the data to be stored as the test set. I have set the parameter random-state to a seed of seven. This involves random sampling of the data but means that we will still get the same random sample if we rerun the experiment.
**Linear Regression Algorithm**

Linear regression algorithm demonstrates the relationship between one dependent variable and one or more independent variables. We built the Linear Regression Algorithm by fitting our data on the model and evaluate the algorithm using the testing data. The result summarized as below table:

<table>
<thead>
<tr>
<th>Linear Regression</th>
<th>Train Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MAE</strong></td>
<td>19.55</td>
<td>19.46</td>
</tr>
<tr>
<td><strong>RMSE</strong></td>
<td>52.35</td>
<td>52.07</td>
</tr>
<tr>
<td><strong>R2</strong></td>
<td>0.80</td>
<td>0.80</td>
</tr>
</tbody>
</table>

*Table 4.8: Linear Regression MAE, RMSE, and R2*

K-folds Cross-Validation has been used to evaluate our model. Cross-Validation K-Fold has been used to divide our training data into a different number of folds that are relatively equal in size. The first fold used as a validation set and then train the model on the remaining folds. For K times. We test the process for different values of K and select the best result, k = 10.

**The accuracy of the linear regression model is 81%**.

- In the following sample table, we can see the actual value of pickups and the predicted value with the error percentage:

<table>
<thead>
<tr>
<th>Community ID</th>
<th>Hour of the Day</th>
<th>Month</th>
<th>Year</th>
<th>Weekday</th>
<th>Actual Number of Pickups</th>
<th>Predicted Number of Pickups</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>594</td>
<td>15</td>
<td>4</td>
<td>2018</td>
<td>Wednesday</td>
<td>29</td>
<td>27</td>
<td>8%</td>
</tr>
<tr>
<td>121</td>
<td>16</td>
<td>4</td>
<td>2018</td>
<td>Saturday</td>
<td>9</td>
<td>23</td>
<td>61%</td>
</tr>
<tr>
<td>616</td>
<td>5</td>
<td>5</td>
<td>2018</td>
<td>Sunday</td>
<td>0</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td>393</td>
<td>0</td>
<td>6</td>
<td>2018</td>
<td>Monday</td>
<td>136</td>
<td>195</td>
<td>30%</td>
</tr>
<tr>
<td>617</td>
<td>9</td>
<td>6</td>
<td>2018</td>
<td>Sunday</td>
<td>0</td>
<td>2</td>
<td>100%</td>
</tr>
<tr>
<td>922</td>
<td>11</td>
<td>6</td>
<td>2018</td>
<td>Saturday</td>
<td>2</td>
<td>1</td>
<td>49%</td>
</tr>
<tr>
<td>246</td>
<td>21</td>
<td>7</td>
<td>2018</td>
<td>Sunday</td>
<td>30</td>
<td>39</td>
<td>22%</td>
</tr>
<tr>
<td>357</td>
<td>18</td>
<td>9</td>
<td>2018</td>
<td>Sunday</td>
<td>30</td>
<td>52</td>
<td>42%</td>
</tr>
<tr>
<td>671</td>
<td>23</td>
<td>2</td>
<td>2019</td>
<td>Wednesday</td>
<td>1</td>
<td>19</td>
<td>95%</td>
</tr>
<tr>
<td>376</td>
<td>10</td>
<td>3</td>
<td>2019</td>
<td>Saturday</td>
<td>25</td>
<td>57</td>
<td>56%</td>
</tr>
<tr>
<td>315</td>
<td>18</td>
<td>4</td>
<td>2019</td>
<td>Wednesday</td>
<td>200</td>
<td>265</td>
<td>25%</td>
</tr>
</tbody>
</table>
K Nearest Neighbors Regression Algorithm

KNN consider as a type of lazy learning that only approximates the function locally, with all estimation delayed until the process is evaluated. Since it depends on distance for scoring, its accuracy can be significantly improved by simplifying the training data.

- Build a KNN Algorithm and evaluate it using the training data:

<table>
<thead>
<tr>
<th>KNN</th>
<th>Train Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>8.17</td>
<td>10.18</td>
</tr>
<tr>
<td>RMSE</td>
<td>26.37</td>
<td>32.57</td>
</tr>
<tr>
<td>R2</td>
<td>0.93</td>
<td>0.92</td>
</tr>
</tbody>
</table>

Table 4. 10: KNN Train Data

- Evaluate the KNN Algorithm using the test data
- Evaluate the KNN Algorithm using Cross-Validation

Cross-Validation K-Fold has been used to divide our training data into a different number of folds that are relatively equal in size. The first fold used as a validation set and then train the model on the remaining folds. For K times. We test the process for different values of K and select the best result, k = 10.

The accuracy of the linear regression model is 95%
• We used the same actual data point that predicted on the Linear regression algorithm and show the result by using the KNN algorithm:

<table>
<thead>
<tr>
<th>Community ID</th>
<th>Hour of the Day</th>
<th>Month</th>
<th>Year</th>
<th>Weekday</th>
<th>Actual Number of Pickups</th>
<th>Predicted Number of Pickups</th>
<th>Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>594</td>
<td>15</td>
<td>4</td>
<td>2018</td>
<td>Wednesday</td>
<td>29</td>
<td>31</td>
<td>6%</td>
</tr>
<tr>
<td>121</td>
<td>16</td>
<td>4</td>
<td>2018</td>
<td>Saturday</td>
<td>9</td>
<td>18.4</td>
<td>51%</td>
</tr>
<tr>
<td>616</td>
<td>5</td>
<td>5</td>
<td>2018</td>
<td>Sunday</td>
<td>0</td>
<td>0.4</td>
<td>100%</td>
</tr>
<tr>
<td>393</td>
<td>0</td>
<td>6</td>
<td>2018</td>
<td>Monday</td>
<td>136</td>
<td>127</td>
<td>7%</td>
</tr>
<tr>
<td>617</td>
<td>9</td>
<td>6</td>
<td>2018</td>
<td>Sunday</td>
<td>0</td>
<td>1.4</td>
<td>100%</td>
</tr>
<tr>
<td>922</td>
<td>11</td>
<td>6</td>
<td>2018</td>
<td>Saturday</td>
<td>2</td>
<td>2.4</td>
<td>17%</td>
</tr>
<tr>
<td>246</td>
<td>21</td>
<td>7</td>
<td>2018</td>
<td>Sunday</td>
<td>30</td>
<td>31.4</td>
<td>4%</td>
</tr>
<tr>
<td>357</td>
<td>18</td>
<td>9</td>
<td>2018</td>
<td>Sunday</td>
<td>30</td>
<td>41</td>
<td>27%</td>
</tr>
<tr>
<td>671</td>
<td>23</td>
<td>2</td>
<td>2019</td>
<td>Wednesday</td>
<td>1</td>
<td>5.6</td>
<td>82%</td>
</tr>
<tr>
<td>376</td>
<td>10</td>
<td>3</td>
<td>2019</td>
<td>Saturday</td>
<td>25</td>
<td>30.2</td>
<td>17%</td>
</tr>
<tr>
<td>315</td>
<td>18</td>
<td>4</td>
<td>2019</td>
<td>Wednesday</td>
<td>200</td>
<td>224</td>
<td>11%</td>
</tr>
<tr>
<td>214</td>
<td>6</td>
<td>5</td>
<td>2019</td>
<td>Tuesday</td>
<td>70</td>
<td>42.4</td>
<td>65%</td>
</tr>
<tr>
<td>412</td>
<td>10</td>
<td>5</td>
<td>2019</td>
<td>Tuesday</td>
<td>90</td>
<td>143.6</td>
<td>37%</td>
</tr>
<tr>
<td>231</td>
<td>23</td>
<td>5</td>
<td>2019</td>
<td>Monday</td>
<td>125</td>
<td>179.4</td>
<td>30%</td>
</tr>
</tbody>
</table>

Table 4.11: KNN Actual vs Predicted Value

**Result**

We can see some differences in the resulting accuracy of these models based on the performance of our data's used regression algorithms; however, the accuracy of each model improved by using Cross-Validation 10 folds KNN algorithm has the highest accuracy of 95%.

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Linear Regression</th>
<th>KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>19.54</td>
<td>8.17</td>
</tr>
<tr>
<td>RMSE</td>
<td>52.28</td>
<td>26.37</td>
</tr>
<tr>
<td>R-squared</td>
<td>0.80</td>
<td>0.93</td>
</tr>
<tr>
<td>Cross Validation Accuracy</td>
<td><strong>0.81</strong></td>
<td><strong>0.95</strong></td>
</tr>
</tbody>
</table>

Table 4.12: Linear Regression vs KNN

The below table shows the actual value and the predicted value of pickups for KNN and Linear regression algorithms with the error percentage. We can indicate the the error percentage for KNN is less that Linear regression and the value is near to the actual number.
<table>
<thead>
<tr>
<th>Actual Number of Pickups</th>
<th>KNN</th>
<th>Linear Regression</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Predicted Number of Pickups</td>
<td>Error %</td>
<td>Predicted Number of Pickups</td>
<td>Error %</td>
</tr>
<tr>
<td>29</td>
<td>31</td>
<td>6%</td>
<td>27</td>
<td>8%</td>
</tr>
<tr>
<td>9</td>
<td>18.4</td>
<td>51%</td>
<td>23</td>
<td>61%</td>
</tr>
<tr>
<td>136</td>
<td>127</td>
<td>7%</td>
<td>195</td>
<td>30%</td>
</tr>
<tr>
<td>2</td>
<td>2.4</td>
<td>17%</td>
<td>1</td>
<td>49%</td>
</tr>
<tr>
<td>30</td>
<td>31.4</td>
<td>4%</td>
<td>39</td>
<td>22%</td>
</tr>
<tr>
<td>30</td>
<td>41</td>
<td>27%</td>
<td>52</td>
<td>42%</td>
</tr>
<tr>
<td>1</td>
<td>5.6</td>
<td>82%</td>
<td>19</td>
<td>95%</td>
</tr>
<tr>
<td>25</td>
<td>30.2</td>
<td>17%</td>
<td>57</td>
<td>56%</td>
</tr>
<tr>
<td>200</td>
<td>224</td>
<td>11%</td>
<td>265</td>
<td>25%</td>
</tr>
<tr>
<td>70</td>
<td>42.4</td>
<td>65%</td>
<td>97</td>
<td>28%</td>
</tr>
<tr>
<td>90</td>
<td>143.6</td>
<td>37%</td>
<td>208</td>
<td>57%</td>
</tr>
<tr>
<td>125</td>
<td>179.4</td>
<td>30%</td>
<td>168</td>
<td>25%</td>
</tr>
</tbody>
</table>

Table 4.13: Error Percentage for KNN and Linear Regression
II. Time Series Analysis

Time series offers the potential for future values to be predicted. The time series can be used to predict patterns in taxi demand based on previous values. The essential features of time-series data indicate that it usually requires advanced statistical methods.

Some differentiated signs appear when we plot the results, as it has a seasonality pattern in the time series. Also, the number of pickups is always high at the end of the year. With a low pickups number in the middle of the year months, there is always a strong upward trend at the end of the year.

The plot shows a repeated pattern, which indicates that the data has a seasonal element.
Figure 4.14: Trend, Seasonality, and Noise

This chart illustrates the number of pickups, and we can notice it is unstable and has apparent seasonality.

**ARIMA**

Autoregressive Integrated Moving Average (ARIMA) is one of the most commonly used methods for time-series forecasting.

**ARIMA Time Series model parameter selection:**

1- Find the three parameters (p, d, q) using function called grid_search().

2- Use SARIMAX() function to fit the seasonal ARIMA model.

3- Result: ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC: 4.0

4- Output: choose SARIMAX(0, 0, 0)x(0, 1, 1, 12) for the output as it has the lowest AIC value (4.0).

**Fitting an ARIMA Time Series Model**

We have defined the set of parameters that generate the best fitting model for our time series data by using grid search. We will continue to analyze in more detail this particular model. So, begin by plugging in the new SARIMAX model with the optimal parameter values.

|                | coef   | std err | z     | P>|z| | [0.025 0.975] |
|----------------|--------|---------|-------|-----|----------------|
| ma.S.L12       | -0.0275| 0.068   | -0.405| 0.686| -0.161 0.106   |
| sigma2         | 1.004e+06| 4.74e+05| 2.119 | 0.034| 7.52e+04 1.93e+06 |
**Forecast Validation**

We obtained the optimal forecast model and compared the output using `get_prediction()` and `conf_int()` functions to correlate the real-time values with the predicted value.

![Observed and Forecast Values](image)

**Figure 4.15: Observed and Forecast Values**

The line chart displays the observed values according to the rolling forecast. Our forecast matches the appropriate values and follows the same trends comparatively. We would use the MSE and RMSE to review the prediction model average error:

- **The MAE of ARIMA Analysis is: 929.93**
- **The RMSE of ARIMA Analysis is: 1179.96**

**Producing and visualizing forecasts**

The `get_forecast()` attribute of the time series object can calculate expected values for a given number of steps forward, which are 36 steps (months) in this plot.
Figure 4.16: Forecasting Data for three years

Our model captured pickups seasonality clearly, as shown in the above line chart that forecasts taxi pickups for 36 months forward.

Facebook Prophet

To begin, we need to instantiate the new object of the Prophet. The Prophet allows us to specify the number of arguments to be used. By setting the interval-width parameter. The Prophet default of the uncertainty interval is 80%, but we will set it to 95%. 'make future DataFrame' function is used to create DataFrame ds that holds the dates we want to forecast.

<table>
<thead>
<tr>
<th>ds</th>
<th>2022-08-01 23:00:00</th>
</tr>
</thead>
<tbody>
<tr>
<td>16951</td>
<td></td>
</tr>
<tr>
<td>16952</td>
<td></td>
</tr>
<tr>
<td>16953</td>
<td></td>
</tr>
<tr>
<td>16954</td>
<td></td>
</tr>
<tr>
<td>16955</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.14: Prophet Forecast

While working with the Prophet, we will consider the frequency of our time series. Thirty-six monthly timestamps were created using the 'make future dataframe' that's means three years into the future.
These are the outputs as the columns that are relevant to our prediction: ds: the datestamp. yhat: the forecasted value of our metric. yhat_lower: the lower bound. yhat_upper: the upper bound.

**The MAE of Prophet Analysis is: 68.54**

**The RMSE of Prophet Analysis is: 120.06**

Results of our forecasts:

![Prophet observed values](image)

The black spots are the model's detected values, and the blue line is the forecasted values, while the blue-shaded regions are the intervals of our forecasts.
Forecasts Components:

Interesting observations are given in the above table. The first chart indicates a small rise in the monthly number of taxi pickups. The second graph shows the weekly trend, during weekdays the trend is very high, while at the lowest point in the weekend "Friday and Saturday". The daily line graph shows the maximum daily traffic at peak hours.
**Result**

The MAE and RMSE for Facebook Prophet Analysis showing less error rate than ARIMA, but when we plot two of these analyses, we found ARIMA following the same trends as the actual data approximately and perform better in predicting the number of pickups.

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>ARIMA</th>
<th>Prophet</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>929.93</td>
<td>68.54</td>
</tr>
<tr>
<td>RMSE</td>
<td>1179.96</td>
<td>120.06</td>
</tr>
</tbody>
</table>

*Table 4.16: ARIMA vs Prophet*
Chapter Five - Conclusion

5.1 Conclusion

This paper presented a comparative study using different prediction algorithms to forecast taxi demand in Dubai. The main objective is to analyze taxi ridership data of pickup count, dropoff count, date, and time to meet a predicted demand in the city of Dubai. The project utilized different regression algorithms to predict the number of pickups in each community in Dubai and apply time series analysis to forecast taxi pickups' future demand.

The first part covered the collected data from the Roads and Transport Authority of Dubai. It contains Dubai Taxi pickups information for each community, including the number of pickups and dropoffs and when it occurred. The data is timestamped, and containing the year, month, day, weekday, and pickup hour. We closely followed the Cross-Industry Standard Process for Data Mining (CRISP-DM). The Project Analysis chapter covered:

• Understanding the business of taxi demand and the collected data.
• Prepare the dataset by following data processing steps.
• Visualize taxi data to find the relations between the attributes.
• Develop a regression algorithm for the prediction model.
• Evaluating model performance.
• Build Time series models and evaluate the performance.

First step was using data pre-processing to ensure the data is cleaned, with no duplication rows, outliers, null values, invalid community IDs, or times outside the range. Next, we moved to explore the data through data visualization. It is a vital element that we utilize; to further develop our understanding of the data. We use it to identify and discover insights from the data by generating various plots and figures.

In the modeling part, we started working on the regression models by exploring different types of candidate algorithms and summarizing our findings covering the models’ performance analysis. The used algorithms were Linear Regression and KNN. The "pickups" attribute was the output that we were trying to predict, which presented the
number of pickups in each community. For the regression analysis, we summarize our findings covering the analysis of:

Mean Absolute Error (MAE) is the average error in units of the predicted feature \((y)\). A value of 0 indicates a perfect fit.

Root Mean Square Error (RMSE) is the average error in units of the predicted feature \((y)\), yet more seriously penalizes larger errors than MAE. Again, 0 indicates a perfect fit.

R-squared \((R^2)\) is a statistical measure of how near the fitted regression line is to the results.

The model that provides the best result is KNN using 10 folds Cross-Validation with an accuracy of 95%.

The next part of our analysis was building a prediction model using time series analysis. To be able to build a time series model, data has been converted to a time-series format. Another critical step is to check the data for any trend or seasonality elements. The Autocorrelation test clearly shows that our data followed a seasonal pattern. So, We examined two different time series analysis, "ARIMA and Facebook Prophet". We used the parameters mean absolute error (MAE), and root mean squared error (RMSE) to review the models' performance.

The MAE and RMSE for Facebook Prophet Analysis showing less error rate than ARIMA, but when we plot two of these analyses, we found ARIMA following the same trends as the actual data approximately and perform better in predicting the number of pickups, so we prefer to use ARIMA in future analysis and studies.

Finally, we provide a list of recommendations to improve the accuracy of predicting by using data fusion from other entities, such as using weather data, incident data, or traffic data, especially on the main roads, and planned events like festivals and concerts. These data can positively improve the accuracy of the models.

### 5.2 Recommendation

Other factors should be considered to develop the study and improve the project's performance, such as weather data, incident data, and traffic, especially on main roads and planned events such as festivals and concerts. These data can improve the accuracy of the models positively.
5.3 Future Works

In future work, I will use Neural Network models to predict the demand in very small zones rather than full community and time intervals in future work. Also, I will further explore how to use the data fusion techniques to use all information from other entities to connect with all events in Dubai, so this can give accurate results to predict the big events need.
Bibliography


Source Code

1. # Import needed packages
2.
3. import warnings
4. warnings.filterwarnings("ignore")
5. import numpy as np
6. import pandas as pd
7. import pandas_profiling
8. import dask.dataframe as dd
9. #import matplotlib.pyplot as plt # Plotting
10. from matplotlib import pyplot as plt
11.
12. import os
13. from mpl_toolkits.mplot3d import Axes3D
14. from sklearn.preprocessing import StandardScaler
15. import scipy
16. import seaborn as sns
17.
18.
19. import datetime
20. import time
21. import seaborn as sns #Plots
22.
23. from datetime import datetime, date, time, timedelta

1. # set working Directory
2. os.chdir('C:/Users\gaming\Desktop\Personal\Master\Capestone\Final')
3. print("Current Working Directory ", os.getcwd())

1. Data Preparation
2.
3. # Read dataset
4. taxidata = pd.read_csv('taxidataset.csv')
5. print(taxidata.columns)
6.
7.1. Data Understanding
8.
9. # Rename Columns
10. taxidata.columns = ["hour","communityid","communityname","day-
    name","date","pickups", "dropoffs"]
11. print(taxidata.columns)
12. taxidata.tail(10)
13. # Summary of the Data Types
14. taxidata.info()
15. print("Number of rows = "+str(len(taxidata)))
16. print("Number of columns = "+str(len(taxidata.columns)))
17.
18. ii. Data Preprocessing
19. #convert Pickups and Dropoffs to int
20. taxidata["pickups"] = taxidata["pickups"].astype(str)
21. taxidata["dropoffs"] = taxidata["dropoffs"].astype(str)
22.
23. taxidata["pickups"] = taxidata.pickups.str.replace(',', '').astype(int)
24. taxidata["dropoffs"] = taxidata.dropoffs.str.replace(',', '').astype(int)
25. taxidata.dtypes
26. taxidata.describe().transpose()
27. # Convert Date and Hour data type to 'datetime'
28. taxidata["date"] = pd.to_datetime(taxidata["date"])  
29. taxidata["hour"] = pd.to_datetime(taxidata["hour"])  
30. taxidata.dtypes
31. taxidata["month"] = taxidata["date"].dt.month
32. taxidata["day"] = taxidata["date"].dt.day
33. taxidata["year"] = taxidata["date"].dt.year
34. taxidata["weekday"] = taxidata["date"].dt.weekday
35. taxidata["hour"] = taxidata["hour"].dt.hour
36. 
37. taxidata
38. taxidata["date"] = pd.to_datetime(taxidata["date"]).dt.date
39. taxidata["date"] = pd.to_datetime(taxidata["date"])  
40. taxidata
41. taxidata.dtypes

1. iii. Data Cleaning
2.
3. #Check that everything has converted by looking at the non-null counts in each column match
4. taxidata.isnull().mean()
5. # Check data for null values
6. taxidata.isnull().sum().sort_values(ascending=False)
7. taxidata.isnull().values.any()
8. taxidata.isnull().sum()
9. # Remove duplication rows
10. taxidata = taxidata.drop_duplicates()
11. # Remove 2017 rows
12. taxidata = taxidata[taxidata.year != 2017]
13. taxidata
14. # Rearrange columns
15. taxidata = taxidata["communityid", "communityname", "pickups", "dropoffs", "hour", "day", "month", "year", "dayname", "weekday", "date"]
16. taxidata.dtypes

1. # Save new dataset
2. taxidata.to_csv(r'C:\Users\gaming\Desktop\Personal\Master\Capstone\Final\taxi-data_md.csv', index = False)
3. # Read dataset new dataset
4. taxidata_md = pd.read_csv('taxi-data_md.csv')
5. taxidata_md
6. taxidata_md["date"] = pd.to_datetime(taxidata_md["date"])
7. taxidata_md.dtypes
8.

1. Data Visualization
2.
3. # Correlation Matrix Plot
4. plt.figure(figsize=(14,14))
5. sns.heatmap(taxidata.corr(),annot=True,)
6.
7. # Another way for Correlation Matrix Plot
8. corrMat = taxidata[:,:].corr();
9. ax = plt.subplots(figsize=(13, 12))
10. ax = sns.heatmap(corrMat,vmin=-1, vmax=1, annot=True, square = True,linewidths=2);
11.

1. Segregating variables: Independent and Dependent Variables
2. # Sklearn processing
3. from sklearn.preprocessing import MinMaxScaler
4. from sklearn.model_selection import train_test_split
5.
# Sklearn regression algorithms

```python
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVC
```

# Sklearn regression model evaluation functions
```python
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
```

```python
from sklearn.model_selection import cross_val_score
```

# Separating independent and dependent variables
```python
x = taxidata.iloc[['communityid', 'dropoffs', 'hour', 'month', 'year', 'weekday']] # the features to make the prediction
y = taxidata['pickups'] # the feature to predict
```

```python
x.shape, y.shape
```

Scaling the data (Using MinMax Scaler)
```python
# Rescale the input features
```
```python
scaler = MinMaxScaler(feature_range=(0,1))
x = scaler.fit_transform(x)
```

# Convert X back to a Pandas DataFrame, for convenience
```python
# x = pd.DataFrame(x)
x.describe().transpose()
```

# Split into train (2/3) and test (1/3) sets
```python
# Split into train (2/3) and test (1/3) sets
```
```python
test_size = 0.33
seed = 1
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)
```

1. i. Linear Regression model
2.
3. Build and check the Linear Regression model using the training data
4. lr_model = LinearRegression()
5. lr_model.fit(x_train, y_train)
6. lr_predict = lr_model.predict(x_train)
7.
8. print(type(lr_model).__name__)
9. print(" MAE", mean_absolute_error(y_train, lr_predict))
10. print(" RMSE", sqrt(mean_squared_error(y_train, lr_predict)))
11. print(" R2", r2_score(y_train, lr_predict))
12. Evaluate the Linear Regression model using the test data
13. # Evaluation the models against test data using MAE, RMSE and R2
14.
15. lr_test_predict = lr_model.predict(x_test)
16. print(type(lr_model).__name__)
17. print(" Test MAE", mean_absolute_error(y_test, lr_test_predict))
18. print(" Test RMSE", sqrt(mean_squared_error(y_test, lr_test_predict)))
19. print(" Test R2", r2_score(y_test, lr_test_predict))
20. Evaluate the Linear Regression model using Cross Validation
21. # Evaluate the Linear Regression model using Cross Validation
22. print(cross_val_score(lr_model, x, y, cv=3))

1. ii. KNN model
2.
3. # Build KNN model and check them against training data using MAE, RMSE and R2
4.
5. knn_model = KNeighborsRegressor()
6. knn_model.fit(x_train, y_train)
7. knn_predict = knn_model.predict(x_train)
8.
9. print(type(knn_model).__name__)
10. print(" MAE", mean_absolute_error(y_train, knn_predict))
11. print(" RMSE", sqrt(mean_squared_error(y_train, knn_predict)))
12. print(" R2", r2_score(y_train, knn_predict))
13. # Evaluation KNN model against test data using MAE, RMSE and R2
14.
15. knn_test_predict = knn_model.predict(x_test)
16. print(type(knn_model).__name__)
17. print(" Test MAE", mean_absolute_error(y_test, knn_test_predict))
18. print(" Test RMSE", sqrt(mean_squared_error(y_test, knn_test_predict)))
19. print(" Test R2", r2_score(y_test, knn_test_predict))
20. # Evaluate the KNN model using Cross Validation
21. print(cross_val_score(knn_model, x, y, cv=3))
1. # Test data predictions
2. knn_df = x_test.copy()
3. knn_df['Prediction'] = knn_predict
4. knn_df['Actual'] = y_test
5. knn_df['Error'] = y_test - knn_predict
6. knn_df

1. **iii. SVM**
2. # Build SVM model and check them against training data using MAE, RMSE and R2
4. svm_model = SVC()
6. svm_model.fit(x_train, y_train)
7. svm_predict = svm_model.predict(x_train)
8. print(type(svm_model).__name__)
9. print("MAE", mean_absolute_error(y_train, svm_predict))
10. print("RMSE", sqrt(mean_squared_error(y_train, svm_predict)))
11. print("R2", r2_score(y_train, svm_predict))
13. # Evaluation SVM model against test data using MAE, RMSE and R2
14. svm_test_predict = svm_model.predict(x_test)
15. print(type(svm_model).__name__)
17. print("Test MAE", mean_absolute_error(y_test, svm_test_predict))
18. print("Test RMSE", sqrt(mean_squared_error(y_test, svm_test_predict)))
19. print("Test R2", r2_score(y_test, svm_test_predict))

1. Elbow for classifier
2. def Elbow(K):
3. # initiating empty list
4. test_mse = []
5.
6. # training model for every value of K
7. for i in K:
8. # Instance of KNN
9. reg = KNN(n_neighbors = i)
10. reg.fit(train_x, train_y)
11. # Appending mse value to empty list calculated using the predictions
12. tmp = reg.predict(test_x)
13. tmp = mse(tmp,test_y)
14. test_mse.append(tmp)
15.
1. `return` test_mse
2. # Defining K range
3. `k = range(1, 40)`
4. # calling above defined function
5. `test = Elbow(k)`
6. # plotting the Curves
7. `plt.plot(k, test)`
8. `plt.xlabel('K Neighbors')`
9. `plt.ylabel('Test Mean Squared Error')`
10. `plt.title('Elbow Curve for test')`
11. # Creating instance of KNN
12. `reg = KNN(n_neighbors = 3)`
13. # Fitting the model
14. `reg.fit(train_x, train_y)`
15. # Predicting over the Train Set and calculating F1
16. `test_predict = reg.predict(test_x)`
17. `k = mse(test_predict, test_y)`
18. `print('Test MSE ', k)`

1. `import` matplotlib
2. `matplotlib.rcParams['axes.labelsize'] = 16`
3. `matplotlib.rcParams['xtick.labelsize'] = 8`
4. `matplotlib.rcParams['ytick.labelsize'] = 8`
5. `matplotlib.rcParams['text.color'] = 'k'
6. # Read dataset
7. `timedata = pd.read_csv('taxidata_time2.csv')`
8. `timedata = timedata.iloc[:, 0:14]`
9. `timedata1 = timedata[['pickups', 'DateTime']]`
10. `timedata1 = timedata1.sort_values('DateTime')`
11. `timedata1`
12. # start and end date
13. `timedata1['DateTime'].min(), timedata1['DateTime'].max()`
14. `timedata1 = timedata1.groupby('DateTime')['pickups'].sum().reset_index()`
4. timedata1
5. Indexing with Time Series Data
6. timedata1 = timedata1.set_index('DateTime')
7. timedata1.index
8. y = timedata1['pickups'].resample('MS').mean()
9. y
10. y['2019']:
11. ####
12. timedata1.groupby('DateTime')['pickups'].mean().plot.bar()
13. #######
14. timedata1.groupby('month')['pickups'].mean().plot.line()
15. #######
16. timedata1.groupby('weekday')['pickups'].mean().plot.line()
17. #######
18. timedata1.groupby('hour')['pickups'].mean().plot.line()

1. Visualizing Taxi pickups Time Series Data
2. y.plot(figsize=(15, 6))
3. plt.show()
4. from pylab import rcParams
5. rcParams['figure.figsize'] = 18, 8
6. decomposition = sm.tsa.seasonal_decompose(y, model='additive')
7. fig = decomposition.plot()
8. plt.show()
9. # test for stationarity
10. from statsmodels.tsa.stattools import adfuller, kpss
11. # ADF Test
12. result = adfuller(timedata1.pickups, autolag='AIC')
13. print(f'ADF Statistic: {result[0]}')
14. print(f'p-value: {result[1]}')
15. for key, value in result[4].items():
16. print('Critical Values: ')
17. print(f' {key}, {value}')
18.
19. # KPSS Test
20. result = kpss(timedata1.pickups, regression='c')
21. print('\nKPSS Statistic: %f % result[0])
24. `print('p-value: %f' % result[1])`
25. for key, value in result[3].items():
26.     `print('Critical Values: ')
27.     `print(f' {key}, {value}')`

1. **Time series forecasting with ARIMA**
2. `p = d = q = range(0, 2)`
3. `pdq = list(itertools.product(p, d, q))`
4. `seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
5. `print('Examples of parameter combinations for Seasonal ARIMA...')`
6. `print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))`
7. `print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[2]))`
8. `print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))`
9. `print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))`
10. `for param in pdq:`
11. ```
12.     for param_seasonal in seasonal_pdq:
13.         try:
14.             mod = sm.tsa.statespace.SARIMAX(y,
15.                 order=param,
16.                 seasonal_order=param_seasonal,
17.                 enforce_stationarity=False,
18.                 enforce_invertibility=False)
19.                 results = mod.fit()
20.                 `print('ARIMA({}x{})12 - AIC:{}'.format(param, param_seasonal, results.aic))`
21.             except:
22.                 continue
1. **Fitting the ARIMA model**
2. `mod = sm.tsa.statespace.SARIMAX(y,
3.                 order=(0, 0, 0),
4.                 seasonal_order=(0, 1, 1, 12),
5.                 enforce_stationarity=False,
6.                 enforce_invertibility=False)
7. `results = mod.fit()`
8. `print(results.summary().tables[1])`
9. `results.plot_diagnostics(figsize=(16, 8))
10. plt.show()
11. Validating forecasts`
```python
12. pred = results.get_prediction(start=pd.to_datetime('2019-06-01'), dynamic=False)
13. pred_ci = pred.conf_int()
14.
15. ax = y['2018'].plot(label='observed')
16. pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
17. ax.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1], color='k', alpha=.2)
18. ax.set_xlabel('Date/Time')
19. ax.set_ylabel('pickups')
20. plt.legend()
21. plt.show()
22.
23. y_forecasted = pred.predicted_mean
24. y_truth = y['2018-01-01':]
25. mse = ((y_forecasted - y_truth) ** 2).mean()
26. print('The Root Mean Squared Error of our forecasts is {}' .format(round(mse, 2)))
27.
1. Producing and visualizing forecasts
2. pred_uc = results.get_forecast(steps=100)
3. pred_ci = pred_uc.conf_int()
4. ax = y.plot(label='observed', figsize=(14, 7))
5. pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
6. ax.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1], color='k', alpha=.25)
7. ax.set_xlabel('Date/Time')
8. ax.set_ylabel('pickups')
9. plt.legend()
10. plt.show()
11.
12. pred_dynamic = results.get_prediction(start=pd.to_datetime('2019-09-01'), dynamic=True, full_results=True)
13. pred_dynamic_ci = pred_dynamic.conf_int()
14.
15. # Extract the predicted and true values of our time series
16. y_forecasted = pred_dynamic.predicted_mean
17. y_truth = y['2019-09-01':]
18.
19. # Compute the mean square error
20. mse = ((y_forecasted - y_truth) ** 2).mean()
21. print('The Root Mean Squared Error of our forecasts is {}' .format(round(mse, 2)))
```
22. # Get forecast 500 steps ahead in future
23. pred_uc = results.get_forecast(steps=500)
24.
25. # Get confidence intervals of forecasts
26. pred_ci = pred_uc.conf_int()
27. ax = y.plot(label='observed', figsize=(20, 15))
28. pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
29. ax.fill_between(pred_ci.index,
30. pred_ci.iloc[:, 0],
31. pred_ci.iloc[:, 1], color='k', alpha=.25)
32. ax.set_xlabel('Date')
33. ax.set_ylabel('pickups')
34.
35. plt.legend()
36. plt.show()

1. Facebook Prophet
2.
3. pip install pystan
4. lllpip install pandas matplotlib numpy cython
5. pip install pystan
6. pip install pandas matplotlib numpy cython
7. conda install libpython m2w64-toolchain -c msys2
8. pip install fbprophet
9. %matplotlib inline
10.
11. from fbprophet import Prophet
12.
13. import matplotlib.pyplot as plt
14. plt.style.use('fivethirtyeight')
15. timedata2 = timedata1
16. timedata2
17. # timedata2['DateTime'] = pd.to_datetime(timedata2['DateTime']).dt.date
18. timedata2.dtypes
19. # timedata2 = timedata2.set_index('DateTime')
20. # timedata2.index
21. timedata2 = timedata2.rename(columns = {'DateTime': 'ds', 'pickups': 'y'})
22. timedata2
23. ax = timedata2.set_index('ds').plot(figsize=(12, 8))
24. ax.set_ylabel('Daily Number of pickups')
25. ax.set_xlabel('Date')
26.
27. plt.show()
28. Time Series Forecasting with Prophet
29.
30. # set the uncertainty interval to 95% (the Prophet default is 80%)
31. my_model = Prophet(interval_width=0.95)
32. my_model.fit(timedata2)
33. future_dates = my_model.make_future_dataframe(periods=36, freq='MS')
34. future_dates.tail(5)
35. forecast = my_model.predict(future_dates)
36. forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
37. my_model.plot(forecast,
38.   uncertainty=True)
39. my_model.plot_components(forecast)