
# Bike-Sharing Demand Prediction System

**School of Engineering and Computer Science**

**University of the Pacific**

**ANLT-214-ON2-81929 Data Engineering**

**Vamshi Krishna Perabathula & Sai Niharika Hari**

**December 07, 2024**

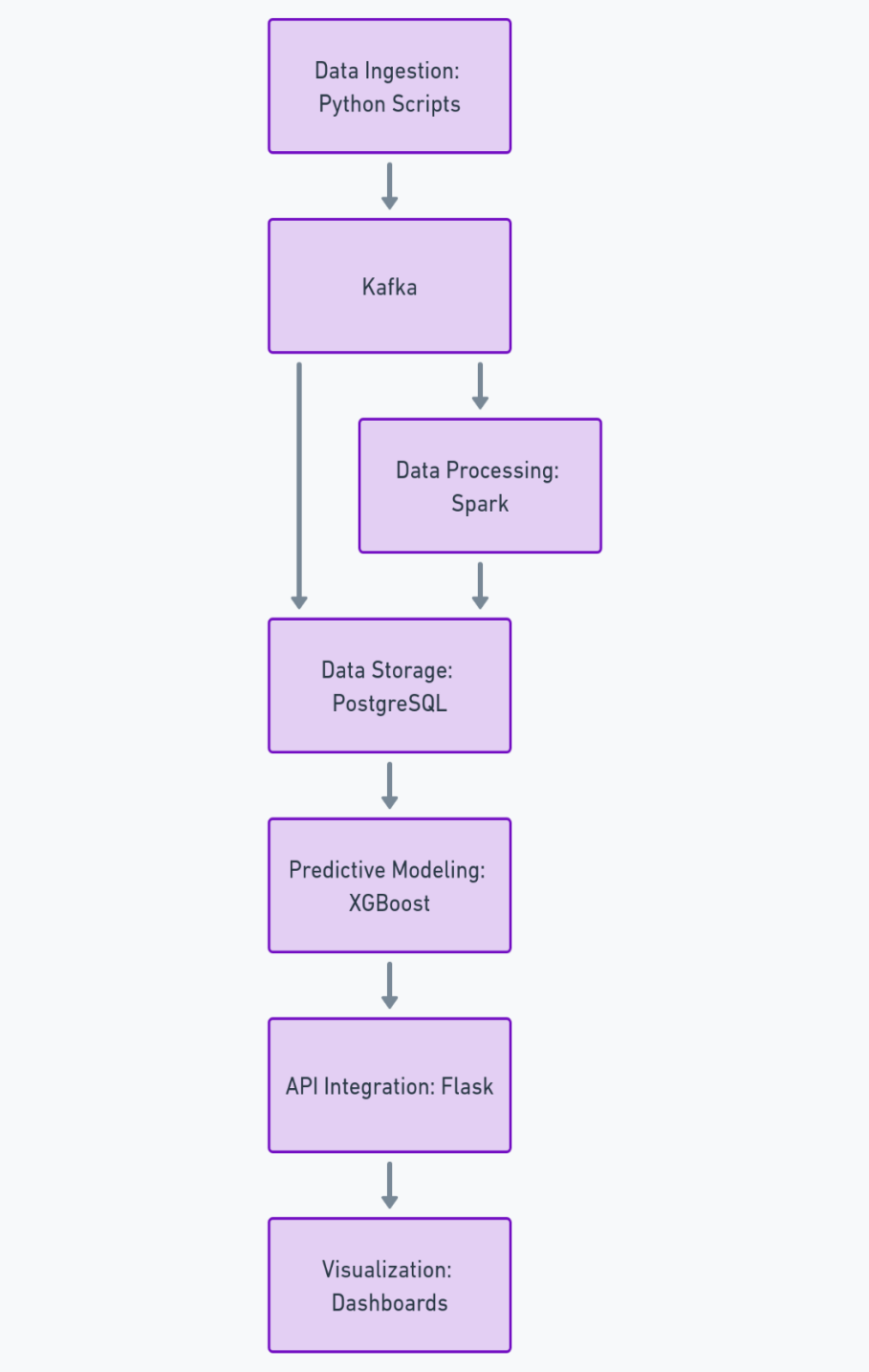
## Introduction

In modern urban environments, bike-sharing services have emerged as a sustainable, cost-effective, and convenient solution for commuters. These systems play a significant role in reducing traffic congestion and lowering carbon emissions, making them an essential part of the urban transportation ecosystem. However, these systems face operational challenges such as the imbalance of bike distribution, inefficiencies in station utilization, and difficulty in predicting user demand.

Our project addresses these issues by developing a comprehensive demand prediction system for bike-sharing services. By leveraging real-time data ingestion, machine learning algorithms, and interactive visualization dashboards, the system empowers operators to predict demand at each station and allocate resources more effectively. The end goal is to improve customer satisfaction, enhance operational efficiency, and ensure seamless service availability across stations.

## Architecture of the System

The architecture of the system is designed to process large volumes of data, extract meaningful insights, and present them in a way that enables real-time decision-making. The system integrates various components, each of which is critical to its functionality.



## Core Components

### 1. Data Ingestion

The system uses Python to simulate real-time data ingestion from bike-sharing stations. Key data points include trip start and stop times, station information, user demographics, and trip durations. Apache Kafka serves as the backbone for real-time data streaming, allowing the system to handle continuous data inflows effectively. Kafka's ability to partition topics ensures scalability and smooth data ingestion, even as the dataset grows.

### 2. Data Storage

The ingested data is stored in a PostgreSQL database, chosen for its ability to handle large datasets and support complex queries. PostgreSQL is used to store both raw trip data and processed features. The structured schema includes:

- **Raw trip data**: timestamps, station IDs, bike IDs, and user types.
- **Processed features**: station-level demand, user activity trends, and time-based aggregations.

### 3. Data Processing

Apache Spark handles the processing of raw data, ensuring scalability and speed. The data is cleaned, transformed, and aggregated to create meaningful features. For example:

- Missing values are imputed, and duplicates are removed.
- Hourly trip counts, average trip durations, and station popularity metrics are calculated.
- Temporal patterns, such as peak usage times and seasonal trends, are extracted.

**4. Predictive Analytics**

The predictive engine is powered by XGBoost, a highly efficient and accurate machine learning library. XGBoost uses processed features to forecast hourly bike demand at each station. Key inputs include historical usage patterns, station-specific metrics, and temporal trends. The output is a set of demand forecasts that allow operators to make proactive decisions.

**5. API Integration**

A Flask API serves as the interface between the predictive engine and external systems. Operators can use the API to query demand forecasts for specific stations and time periods. This API ensures seamless integration with other operational tools used by bike-sharing companies.

**6. Visualization**

The final component of the architecture is the visualization dashboard, built using Dash and Plotly. The dashboard provides an interactive interface for exploring data trends, visualizing predictions, and analysing station-level performance. Key visualizations include:

- Heatmaps of peak usage times and locations.
- Tree maps showing station utilization.
- Correlation matrices highlighting relationships between variables.

## System Scalability and Maintenance

The system is designed with scalability in mind. Apache Kafka's partitioning capability and Spark's distributed processing architecture ensure that the system can handle increasing data volumes. PostgreSQL indexing and partitioning further optimize query performance.

Additionally, the modular design allows individual components to be scaled or updated independently, ensuring long-term maintainability.

## Process and Data Flow

The system's data flow transforms raw trip data into actionable insights through a series of well-defined stages. Below is a detailed explanation of each stage:

### 1. Data Collection

The data for this project comes from the New York City Citi Bike dataset, a publicly available source of historical trip information. This dataset includes trip details such as start and stop times, station names, user types, and bike IDs. To simulate real-world conditions, Python scripts were developed to generate real-time data streams, capturing thousands of bike trips per day.

### 2. Data Ingestion

The simulated data is ingested into Apache Kafka, which organizes it into topics such as trip data and station data. Kafka ensures that the data streams are processed in real-time, providing the foundation for downstream analysis.

### 3. Data Storage

The ingested data is stored in a PostgreSQL database. The schema is structured to facilitate efficient querying and analysis. For instance:

- Raw trip data is stored for historical analysis.
- Processed features, such as hourly demand and station utilization, are stored for predictive modelling and visualization.

**4. Data Processing**

Apache Spark processes the raw data in two stages:

- **Cleaning and Transformation**: This includes removing invalid records, standardizing date formats, and handling missing values.
- **Feature Engineering**: New features are created to enhance the predictive model's performance. Examples include: Hourly trip counts for each station, peak usage times segmented by user type (Subscriber or Customer), temporal patterns such as daily, weekly, and seasonal trends.

These processed features are stored back in PostgreSQL for further analysis.

**5. Predictive Modelling**

Using the processed data, an XGBoost model is trained to predict hourly demand at each station. The model captures:

- **Temporal trends**: Peak hours, seasonal patterns, and holiday effects.
- **Station-specific demand**: Stations with consistently high or low activity levels.
- **Demographic influences**: Differences in usage patterns between Subscribers and Customers.

The model outputs demand forecasts, which are stored in PostgreSQL and served via the Flask API.

**6. Visualization**

The final stage involves presenting insights through an interactive dashboard. Key features include:

- **Historical trends**: Users can explore bike usage by hour, day, or month.

- **Heatmaps**: Highlight peak demand times and locations.
- **Tree maps**: Visualize station utilization and relative importance.
- **Correlation matrices**: Explore relationships between variables such as trip duration and user demographics.

These visualizations allow operators to identify trends, compare actual vs. predicted demand, and make data-driven decisions.

## Purpose of the System

The bike-sharing demand prediction system aims to solve the operational inefficiencies faced by bike-sharing services. By predicting demand at each station, the system enables proactive resource allocation, reducing manual redistribution efforts and enhancing customer satisfaction.

## Key Problems Addressed

- Imbalanced Bike Distribution: Ensures bikes are available where and when they're needed most.
- Operational Inefficiencies: Reduces costs associated with manual redistribution of bikes.
- Customer Dissatisfaction: Improves the user experience by minimizing scenarios where bikes or docking spaces are unavailable.

## Improvements Over Existing Systems

Traditional bike-sharing systems rely on reactive measures to address operational challenges. This system offers a proactive, data-driven approach by:

- **Predicting Demand**: Provides hourly forecasts for each station, enabling better planning.
- **Enhancing Visualization**: Dashboards offer dynamic, user-friendly interfaces for exploring data trends.
- **Improving Scalability**: Modular architecture ensures the system can handle increasing data volumes.

The integration of real-time data ingestion, predictive modelling, and interactive visualization sets this system apart, making it a powerful tool for managing urban mobility.

## Implementation, Configuration, and Management

**Implementation:**

- Install and configure PostgreSQL for data storage.
- Set up Apache Spark for data processing and feature engineering.
- Train the XGBoost model using historical data and validate its performance.
- Deploy the Flask API for serving predictions and integrate it with the visualization dashboard.

**Configuration:**

- Optimize PostgreSQL for large-scale queries using indexing.
- Configure Spark clusters for distributed processing.
- Fine-tune XGBoost hyperparameters to enhance prediction accuracy.

**Management:**

- Retrain the predictive model periodically to incorporate new data.

- Monitor the Flask API for performance and downtime.
- Enhance the dashboard based on user feedback, adding new filters or visualizations as needed.

## Expected Inputs and Outputs

**Inputs:** Trip data with metrics like start/stop times, station names, and user demographics, Historical data for training the predictive model.

**Outputs:** Hourly demand forecasts for each station, Insights into temporal patterns and station usage, Interactive visualizations for decision-making.

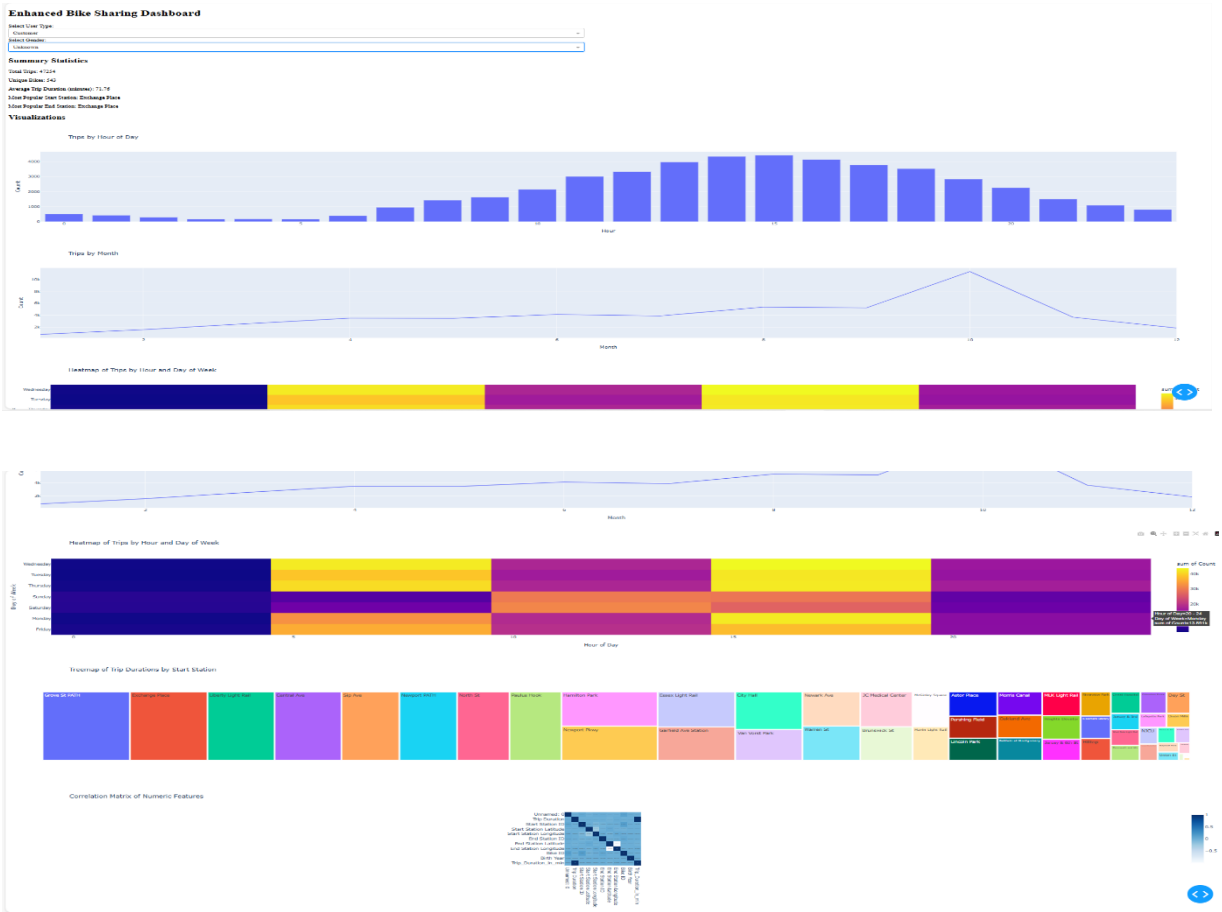



figure 1: customer unknown

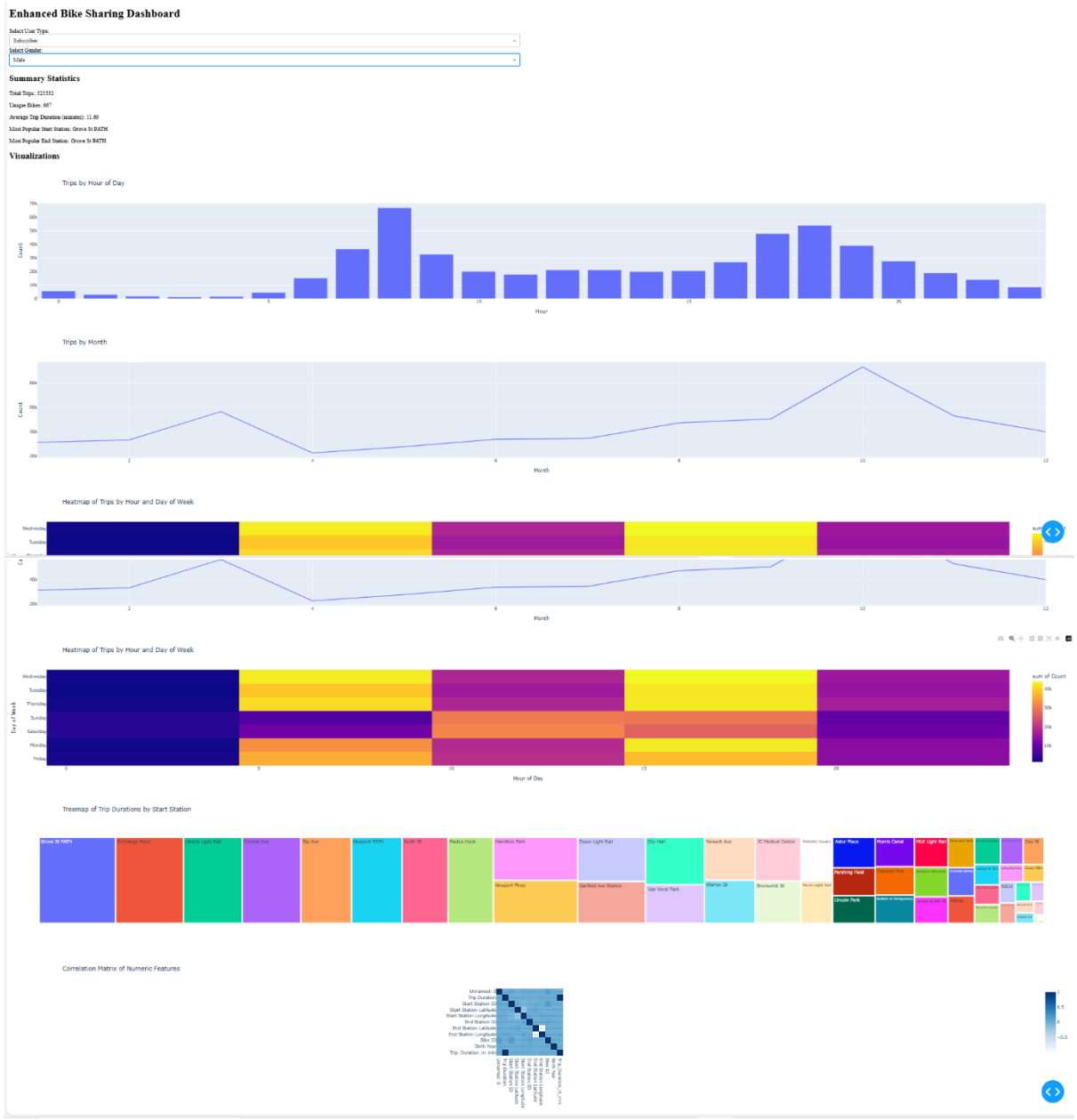



Figure 2: subscriber male


Figure 3: customer female

## Conclusion

The bike-sharing demand prediction system represents a significant step forward in optimizing urban transportation. By combining advanced analytics, machine learning, and intuitive dashboards, it empowers bike-sharing operators to make informed decisions that benefit both users and service providers. As cities continue to adopt sustainable mobility solutions, systems like this will play a crucial role in ensuring their success. With further enhancements, such as

integrating weather data or event schedules, this system has the potential to become the gold standard for bike-sharing management.

## Datasets

### NYC Citi Bike Dataset

- Source: https://www.kaggle.com/datasets/akkithetechie/new-york-city-bike-share-dataset
- Description: This dataset contains historical trip data, including start and end times, station names, bike IDs, and user types.
- Usage: Used to simulate real-time data streams and generate features for predictive modelling.

## External Tools and Libraries

### Apache Kafka

- Documentation: Kafka Documentation
- Tutorials: Introduction to Kafka
- Usage: Simulates real-time data ingestion for trip data streams.

### Apache Spark

- Documentation: Apache Spark Official Documentation
- Tutorials: Introduction to Spark
- Usage: Processes raw data for cleaning, feature engineering, and aggregation.

### PostgreSQL

- Documentation: PostgreSQL Official Documentation

- Tutorials: PostgreSQL Tutorial
- Usage: Stores raw and processed data with structured schemas for efficient querying.

**XGBoost**

- Documentation: XGBoost Official Documentation
- Tutorials: XGBoost Python Example
- Usage: Machine learning library used to build the predictive model for bike demand.

**Flask**

- Documentation: Flask Documentation
- Tutorials: Flask Tutorial
- Usage: Provides an API interface for accessing predictions.

**Dash and Plotly**

- Documentation: Dash Official Documentation
- Tutorials: Getting Started with Dash
- Usage: Creates the interactive dashboard for visualizing data trends and predictions.

**Docker Desktop**

Used for combing and installing all the above resources at one place and getting the expected output with the inputs of software's used as the pipeline.

## Supporting Documentation and Tutorials

### General Machine Learning Workflow

- Resource: Scikit-Learn Machine Learning Guide
- Description: Provides foundational concepts applicable to building and training models like XGBoost.

### Data Visualization Techniques

- Resource: Plotly Visualizations Guide
- Description: A comprehensive guide to building interactive visualizations in Python.

### Data Engineering Practices

- Resource: Data Engineering Cookbook
- Description: Includes best practices for building scalable data pipelines and systems.

### Real-Time Data Processing

- Resource: Confluent Real-Time Event Streaming Guide
- Description: Explains real-time processing principles relevant to Kafka and Spark integration.

### SQL Optimization

- Resource: SQL Indexing and Optimization
- Description: Tips for optimizing queries and improving database performance.

## Citations


1. New york Bike Dataset: https://www.kaggle.com/datasets/akkithetechie/new-york-city-bike-share-dataset
2. Apache Kafka: Kafka Documentation
3. Apache Spark: Apache Spark Official Documentation
4. PostgreSQL: PostgreSQL Official Documentation
5. XGBoost: XGBoost Official Documentation
6. Flask: Flask Documentation
7. Dash and Plotly: Dash Official Documentation
8. ChatGPT, Google,StackOverflow