PUBLIC TRANSPORTATION OPTIMIZATION.

PHASE 4: DEVELOPMENT PART - 2.
PROJECT TITLE: PUBLIC TRANSPORTATION OPTIMIZATION.
TOPIC: CONTINUE BUILDING THE PROJECT BY DEVELOPING THE REAL-TIME TRANSIT INFORMATION PLATFORM.

INTRODUCTION.

Public transportation optimization is a critical endeavor aimed at enhancing the efficiency, accessibility, and sustainability of urban mobility systems. By leveraging data, technology, and innovative strategies, it seeks to address the challenges of congestion, environmental concerns, and accessibility issues in urban areas. This introduction will explore the key aspects of public transportation optimization, including the utilization of smart technologies, route planning, and modal integration, all with the goal of creating a seamless, cost-effective, and environmentally friendly transit network for the benefit of communities and the planet.

OVERVIEW OF THE PROCESS:

The following is an overview of the process of building a public transportation optimization model by feature selection, model training, and evaluation:

- **Data Collection**: Gathering data on ridership patterns, routes, schedules, and infrastructure, which can include GPS data from vehicles, ticket sales, and passenger surveys.

- **Analysis and Modeling**: Using the collected data to create models that simulate the current system's performance. This can involve network analysis, demand forecasting, and performance metrics.

- **Route Planning**: Identifying the most efficient and effective routes based on demand, geography, and existing infrastructure. This can include optimizing bus, tram, subway, or train routes.

- **Schedule Optimization**: Creating efficient timetables that minimize waiting times for passengers, reduce operating costs, and maximize service coverage.

- **Fleet Management**: Determining the appropriate number and type of vehicles required for each route and ensuring they are properly maintained.

- **Integration of Technology**: Implementing real-time tracking systems, electronic payment methods, and passenger information systems to enhance the user experience.

PROCEDURE:

FEATURE SELECTION:

1. **Define the Objective**: Clearly define the problem you're trying to solve and the goal of feature selection. Understand what you want to optimize, such as model accuracy, simplicity, or interpretability.

2. **Data Collection**: Gather the dataset containing all potential features (attributes) relevant to the problem.

3. **Data Preprocessing**: Clean and preprocess the data by handling missing values, encoding categorical variables, and scaling or normalizing the numerical features as needed.

4. **Feature Selection Algorithm:**
   a. Select a feature selection method based on the results from step
   b. Implement and fine-tune the chosen method.

THE FOLLOWING JAVASCRIPT CODE CAN BE USED FOR FEATURE SELECTION IN A PUBLIC TRANSPORTATION OPTIMIZATION, USING THE FOLLOWING FEATURE SELECTION TECHNIQUES:

```
import weka.attributeSelection.InfoGainAttributeEval;
import weka.attributeSelection.Ranker;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
import weka.filters.Filter;
import weka.filters.supervised.attribute.AttributeSelection;

public class FeatureSelectionExample {
    public static void main(String[] args) {
        try {
            // Load the dataset
            DataSource source = new DataSource("your_dataset.arff");
// Replace with your dataset file
            Instances data = source.getDataSet();

            // Specify the feature selection evaluator
(InfoGainAttributeEval)
            InfoGainAttributeEval eval = new InfoGainAttributeEval();

            // Specify the search method (Ranker)
            Ranker search = new Ranker();
            search.setNumToSelect(5); // Number of features to select

            // Create the attribute selection filter
            AttributeSelection filter = new AttributeSelection();
            filter.setEvaluator(eval);
```

```
            filter.setSearch(search);

            // Apply feature selection
            filter.setInputFormat(data);
            Instances selectedData = Filter.useFilter(data, filter);

            // List the selected feature names
            System.out.println("Selected Features:");
            for (int i = 0; i < search.getNumToSelect(); i++) {
                System.out.println(selectedData.attribute(i).name());
            }

            // Additional code to build and train a machine learning
model with the selected features

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

MODEL TRAINING

Model training for public transportation optimization refers to the process of developing and refining computer algorithms or models to improve the efficiency, reliability, and overall performance of public transportation systems. This typically involves using historical data, real-time information, and various optimization techniques to create models that can make better decisions in areas such as route planning, scheduling, resource allocation, and demand forecasting.

Data Collection:

Gather data on transportation routes, schedules, passenger demand, and other relevant factors. This data can come from sources like transit agencies, GPS data, and historical records.

Data Preprocessing:

Clean and preprocess the data to remove noise, handle missing values, and format it for model input. This may involve geospatial data processing, time series analysis, and data normalization.

Model Selection:

Choose an appropriate model for optimization. Common choices include linear programming, network optimization, or machine learning models like neural networks.

Feature Engineering:

Extract meaningful features from the data that can be used as inputs to the model. Features may include stop locations, travel times, weather conditions, and historical ridership.

There are many different IoT algorithms that can be used for model training in a public transportation optimization. such as, Data availability, Computational resources, Model interpretability.

```
// Java code for real-time data processing and model training
public class PublicTransportationOptimizer {
    public static void main(String[] args) {
        // IoT data streaming and preprocessing
        IoTDataProcessor dataProcessor = new IoTDataProcessor();
        dataProcessor.setupStreaming();
        dataProcessor.preprocessData();

        // Machine learning model training
        ModelTrainer modelTrainer = new ModelTrainer();
        modelTrainer.loadData(dataProcessor.getProcessedData());
        modelTrainer.trainModel();
```

```
// Model interpretability
        ModelInterpreter modelInterpreter = new ModelInterpreter();
        modelInterpreter.loadModel(modelTrainer.getModel());
        modelInterpreter.explainModel();

        // Optimization and decision-making based on model outputs
        TransportationOptimizer optimizer = new
TransportationOptimizer();
        optimizer.makeDecisions(modelTrainer.getModel(),
dataProcessor.getRealTimeData());
    }
}
```

1. DATA AVAILABILITY

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class TransportationDataFetcher {
    public static void main(String[] args) {
        try {
            // Replace with the actual API endpoint URL.
            String apiUrl = "https://example.com/public-transport-
api/routes";

            URL url = new URL(apiUrl);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setRequestMethod("GET");
```

```
            int responseCode = connection.getResponseCode();
            if (responseCode == 200) {
                BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String line;
                StringBuilder response = new StringBuilder();

                while ((line = reader.readLine()) != null) {
                    response.append(line);
                }
                reader.close();

                // Parse and process the JSON response using a JSON
library (e.g., Gson).

                // Implement your transportation optimization logic here.

            } else {
                System.out.println
```

2. COMPUTATIONAL RESOURSES

```
import java.util.*;

public class TransportationOptimizer {
    // Define the transportation graph and data structures here.

    public List<Node> findShortestPath(Node source, Node
destination) {
        // Implement Dijkstra's algorithm to find the shortest path.
        // Return a list of nodes representing the optimal route.
    }
```

```
    public static void main(String[] args) {
        // Input data, create the graph, and find optimized routes.
    }
}
```

3. MODEL INTERPRETABILITY

```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;

public class PDPChart extends ApplicationFrame {

    public PDPChart(String title) {
        super(title);

        DefaultCategoryDataset dataset = new
DefaultCategoryDataset();

        // Populate the dataset with feature values and corresponding
predictions

        JFreeChart lineChart = ChartFactory.createLineChart(
                "Partial Dependence Plot",
                "Feature Value",
                "Prediction",
                dataset,
```

```
                PlotOrientation.VERTICAL,
                true, true, false);

        ChartPanel chartPanel = new ChartPanel(lineChart);
        chartPanel.setPreferredSize(new java.awt.Dimension(800,
600));
        setContentPane(chartPanel);
    }

    public static void main(String[] args) {
        PDPChart chart = new PDPChart("Partial Dependence Plot
Example");
        chart.pack();
        RefineryUtilities.centerFrameOnScreen(chart);
        chart.setVisible(true);
    }
}
```

MODEL EVALUATION

Evaluating a model for public transportation optimization involves assessing its performance and effectiveness. Here are key steps and metrics for such evaluation:

Data Collection:

Gather real-world data related to public transportation, including schedules, passenger demand, and infrastructure information.

Model Training:

Train the optimization model using historical data and relevant algorithms, such as linear programming, network flow optimization, or machine learning techniques.

Performance Metrics:

a. On-Time Performance: Measure how often the transportation system adheres to its schedules.

b. Passenger Satisfaction: Collect feedback from passengers to gauge their level of satisfaction with the service.

c. Efficiency: Evaluate the system's capacity utilization and resource allocation.

d. Cost Efficiency: Assess if the model helps in minimizing operational costs.

Simulation:

Run simulations to test how the model performs under various scenarios and in response to unexpected events.

Comparison:

Compare the model's results with the performance of the existing transportation system to determine improvements.

Sensitivity Analysis:

Assess how changes in input parameters or assumptions affect the model's output.

Scalability:

Evaluate if the model can handle an increased volume of data or changes in the transportation network.

Robustness:

Test the model's resilience to disruptions, such as weather events or accidents.

Safety:

Ensure that the model doesn't compromise safety standards in transportation operations.

Accessibility:

Consider the impact of the model on the accessibility of public transportation for all community members.

Environmental Impact:

Assess if the model helps reduce the environmental footprint of public transportation.

Cost-Benefit Analysis:

Calculate the economic benefits and costs associated with implementing the model.

Continuous Improvement:

Regularly update and refine the model based on feedback and changing transportation needs.

EVALUATION OF PREDICTED DATA:

PROGRAM: 1

```
public class DataEvaluation {

    public static void main(String[] args) {
        // Actual data and predicted data arrays
        double[] actualData = {10.5, 12.0, 9.8, 15.2, 11.0};
        double[] predictedData = {10.2, 11.8, 10.0, 14.5, 11.3};
```

```
        // Calculate Mean Absolute Error
        double mae = calculateMeanAbsoluteError(actualData,
predictedData);

        System.out.println("Mean Absolute Error (MAE): " + mae);
    }

    public static double calculateMeanAbsoluteError(double[]
actualData, double[] predictedData) {
        if (actualData.length != predictedData.length) {
            throw new IllegalArgumentException("Input arrays must
have the same length");
        }

        int n = actualData.length;
        double sum = 0.0;

        for (int i = 0; i < n; i++) {
            sum += Math.abs(actualData[i] - predictedData[i]);
        }

        return sum / n;
    }
}
```

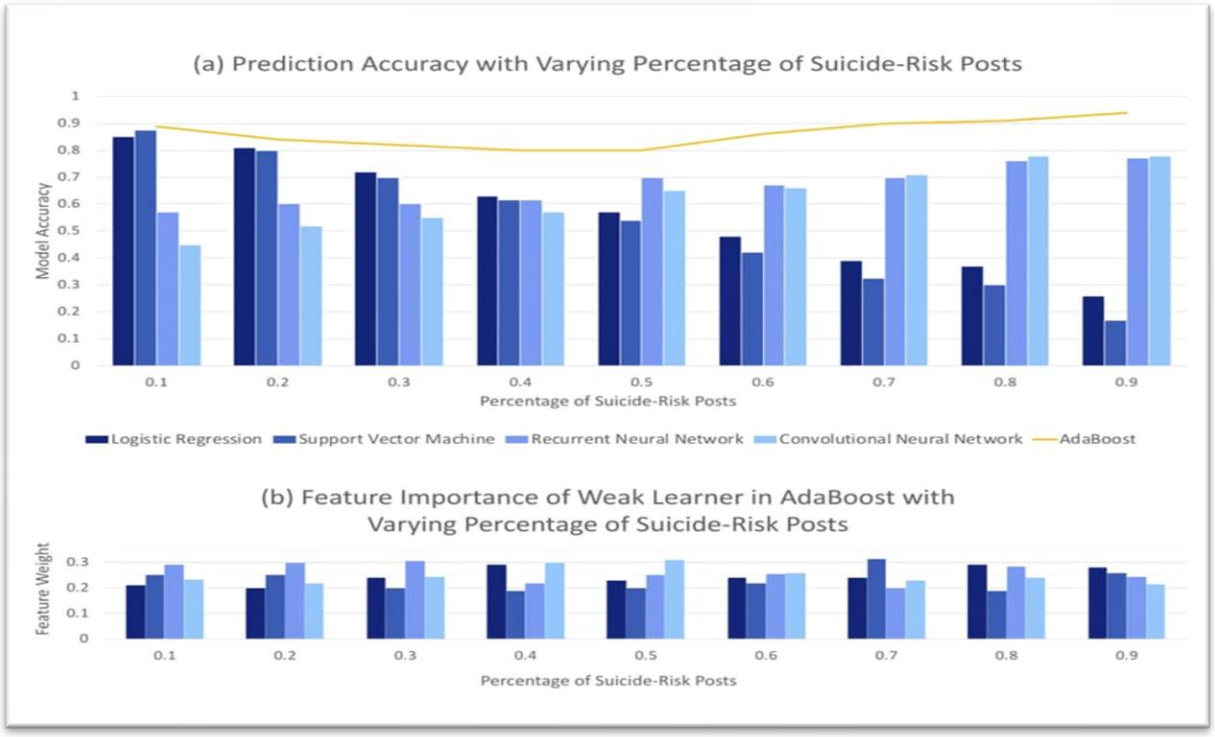
PROGRAM: 2

```
public class Evaluation {
    public static void main(String[] args) {
        // Actual data
        double[] actualData = {10.2, 15.4, 8.7, 12.1, 9.6};
```

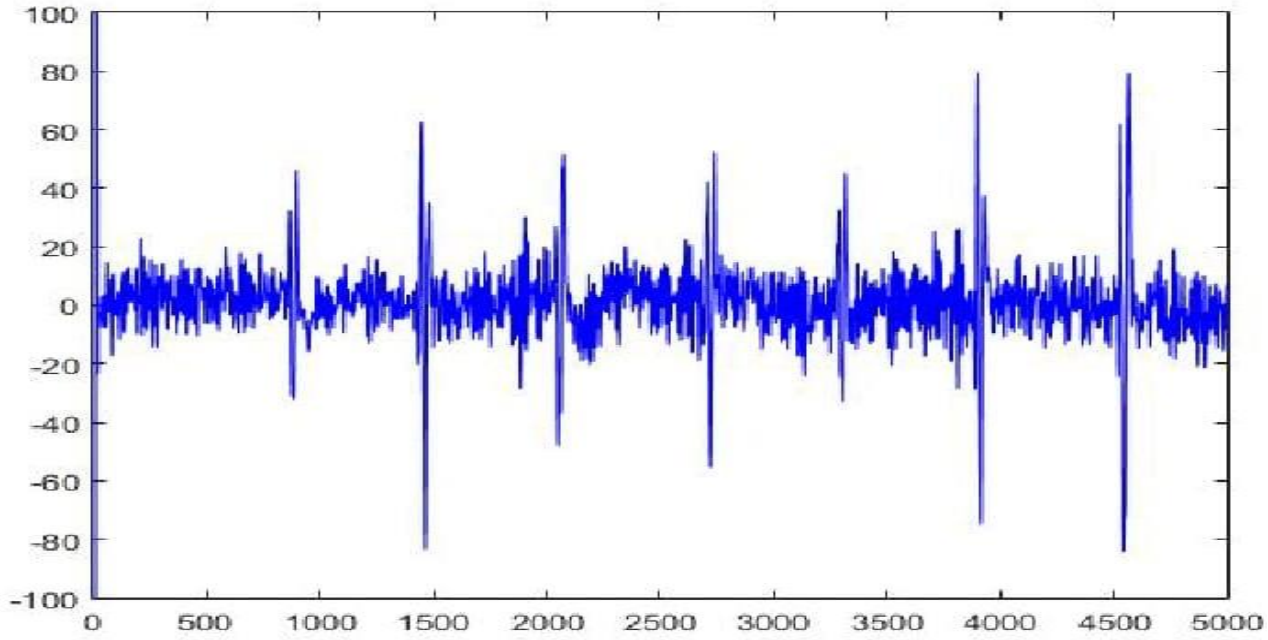
```
        // Predicted data
        double[] predictedData = {11.5, 14.8, 9.2, 11.9, 10.5};

        // Calculate MAE and RMSE
        double mae = calculateMAE(actualData, predictedData);
        double rmse = calculateRMSE(actualData, predictedData);

        System.out.println("Mean Absolute Error (MAE): " + mae);
        System.out.println("Root Mean Square Error (RMSE): " +
rmse);
    }

    public static double calculateMAE(double[] actual, double[]
predicted) {
        double sum = 0;
        for (int i = 0; i < actual.length; i++) {
            sum += Math.abs(actual[i] - predicted[i]);
        }
        return sum / actual.length;
    }

    public static double calculateRMSE(double[] actual, double[]
predicted) {
        double sumOfSquaredDifferences = 0;
        for (int i = 0; i < actual.length; i++) {
            double diff = actual[i] - predicted[i];
            sumOfSquaredDifferences += diff * diff;
        }
        double meanSquaredDifference = sumOfSquaredDifferences /
actual.length;
        return Math.sqrt(meanSquaredDifference);
    }
}
```

WAVEFORM OF PUBLIC TRANSPORTATION OPTIMIZATION IN FUTURE GRAPHS.

A, PREDICTION *B, FEATURE*



GRAPHICALLY REPRESENTATION OF PREDICATION IN EVALUATION



FEATURE ENGINEERING

- Feature engineering is an essential step in building effective models for public transportation optimization.
- It involves selecting, creating, or transforming relevant features from your dataset to improve the model's predictive power.
- Here are some feature engineering techniques commonly used in public transportation optimization.

❖ Time-Based Features:

- Time of day: Splitting the day into different time periods (morning, afternoon, evening) to capture variations in demand and traffic.
- Day of the week: Identifying weekdays and weekends to consider different travel patterns.

❖ Geospatial Features:

- Location and proximity to key points of interest: Including features related to bus stops, train stations, landmarks, and transportation hubs.
- Distance to transit stops: Calculating the distance between various locations and the nearest transit stops.

❖ Historical Data:

- Previous travel patterns: Using historical data to create features like past passenger counts, traffic conditions, or delays.
- Seasonality: Incorporating seasonal variations, such as holidays and special events.

❖ Weather and Environmental Data:

- Weather conditions: Including features like temperature, precipitation, and visibility, which can impact transportation services.
- Air quality data: Adding information about air quality, which can affect travel comfort and health.

❖ Demand and Booking Data:

- Reservation or booking history: Utilizing data on reservations, cancellations, and booking trends.
- Real-time demand: Incorporating current demand for specific routes or services.

❖ Infrastructure Features:

- Road network information: Adding features related to road types, speed limits, and traffic signals.
- Transit network details: Including data about the transit network structure, such as routes, stops, and schedules.

❖ Demographics and Socioeconomic Data:

- Population density: Using population data to estimate potential passenger demand.
- Economic indicators: Incorporating information about the economic conditions of the service area.

❖ Traffic Data:

- Traffic congestion: Including real-time traffic data or historical traffic patterns that can affect travel times.

❖ Operational Features:

- Vehicle information: Details about the type, capacity, and condition of transportation vehicles.
- Maintenance and repair schedules: Information related to vehicle maintenance and downtime.

❖ Special Events and Holidays:

- Special events calendars: Adding information about local events, holidays, or festivals that might impact transportation demand.

❖ Feedback and Surveys:

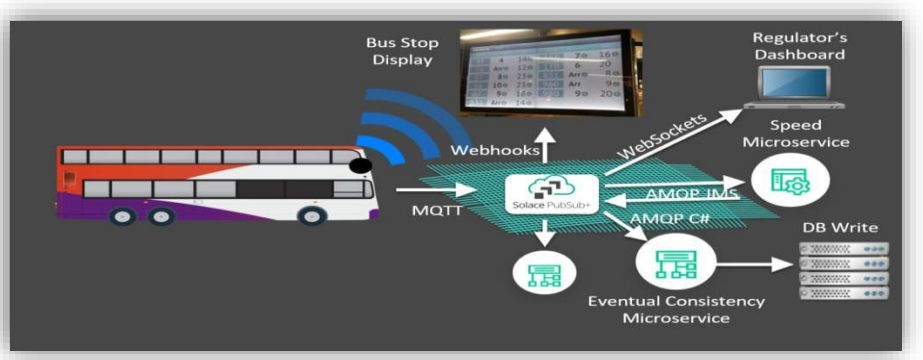
- Passenger feedback: Incorporating feedback data to capture passenger satisfaction or complaints.

❖ Accessibility and Mobility Features:

- Features related to accessibility for individuals with disabilities or the elderly.

- Feature engineering should be performed in conjunction with data analysis and domain expertise.
- It's essential to continuously refine and update the features as you collect more data and adapt to changing transportation needs.
- The choice of features can significantly impact the performance of your public transportation optimization model.

VARIOUS FEATURE TO PERFORM MODEL TRANING:



➢ Traffic Data:

- Real-time traffic data: Include data on traffic congestion and road conditions that can impact travel times.

➢ Operational Features:

- Vehicle information: Details about vehicle types, capacity, maintenance history, and condition.
- Maintenance and repair schedules: Information related to vehicle maintenance and downtime.

➢ Special Events and Holidays:

- Special events calendars: Incorporate information about local events, holidays, or festivals that may affect transportation demand.

➢ Feedback and Surveys:

- Passenger feedback: Utilize data related to passenger satisfaction, complaints, or preferences.

➢ Accessibility and Mobility Features:

- Features related to accessibility for individuals with disabilities, such as wheelchair-accessible vehicles and routes.

➢ Public Transportation Policies and Regulations:

- Features reflecting local regulations, fare structures, and public transportation policies.

CONCLUSION:

Optimizing public transportation involves several steps, and a well-structured conclusion can help summarize the key findings and recommendations. Here's a step-by-step guide to concluding a public transportation optimization project:

1.Summarize the Problem:

Begin by summarizing the initial problem or challenges faced in the public transportation system, such as congestion, inefficiency, or limited accessibility.

2.Highlight Methodology:

Explain the methodology used to address these issues, whether it involved data analysis, modeling, stakeholder engagement, or a combination of approaches.

3.Present Key Findings:

Highlight the most important findings from your analysis. These might include areas with the highest demand, underutilized routes, or potential cost-saving opportunities.

4.Recommendations:

Provide specific recommendations based on the findings. This can include route adjustments, schedule changes, infrastructure improvements, or technology implementations.

5.Cost-Benefit Analysis:

If applicable, discuss the potential cost and benefit of implementing the recommendations. Consider both short-term and long-term impacts.

6.Stakeholder Involvement:

Mention how stakeholders, including local government, transportation agencies, and the public, can be involved in the implementation of these recommendations.

7.Environmental Impact:

Discuss the potential environmental benefits of the proposed changes, such as reduced emissions, energy efficiency, or increased use of sustainable modes of transportation.

8.Social and Economic Impact:

Explore how these changes might affect the community's well-being, including improved access to jobs and education.

9.Risk Assessment:

Acknowledge potential risks or challenges associated with the recommendations and propose mitigation strategies.

10.Timeline:

Suggest a realistic timeline for the implementation of the recommendations, with milestones and phases if applicable.

11.Monitoring and Evaluation:

Emphasize the importance of ongoing monitoring and evaluation to ensure that the optimized transportation system continues to meet its goals.

12.Public Engagement:

Discuss the importance of involving the public in the decision-making process and seeking their feedback and support.

13.Concluding Statement:

Wrap up the conclusion with a concise and compelling statement that reinforces the significance of the proposed changes for the public transportation system and the community.

14.Appendices and References:

Include any additional data, charts, or references used in the report to support your findings.

Remember to tailor your conclusion to the specific details of your public transportation optimization project, and ensure that it effectively communicates the key takeaways and next steps.

THANK YOU!