Ride-Hailing Demand Prediction Task

Problem

In a ride-hailing marketplace, efficient supply-demand matching is critical. The challenge here is to guide drivers toward high-demand areas, ensuring that riders get timely rides and drivers maintain steady earnings. Understanding demand fluctuations across time and location is key to optimizing this system.

Project Objective

1. Analyze data and propose a solution to guide drivers to high-demand areas.
2. Build and document a baseline demand prediction model.
3. Outline deployment and model communication strategies for drivers.
4. Design an experiment to validate the solution's impact in real operations.

Data Overview

The dataset contains ~630,000 rows of synthetic demand data, representing ride requests in Tallinn:

- **start_time**: Order timestamp
- **start_lat, start_lng**: Pickup location
- **end_lat, end_lng**: Destination location
- **ride_value**: Value of the ride (EUR)

Project Plan

The goal is to develop a model to predict demand hotspots across Tallinn, enabling drivers to optimize routes and maximize potential earnings.

Solution Outline

1. Imports and Preprocessing

- Load and clean the data.

2. Exploratory Data Analysis (EDA)

- Visualize pickup locations and analyze demand patterns by day, week, and hour.
- Identify outliers in ride value and filter to improve model accuracy.

3. Data Filtering and Feature Engineering

- **Outlier Treatment**: Percentile-based capping to handle ride value outliers.
- **Location Clustering(MiniBatchKMeans)**: Cluster pickup locations to create demand hotspots, facilitating area-based analysis.

Further Development: To enhance our clustering analysis, we will use K-means clustering and determine the optimal number of clusters by applying the silhouette score, which assesses cluster cohesion, and the elbow method, which identifies where adding more clusters offers limited improvement. This approach will ensure that the resulting groups are meaningful and useful.

4. Baseline Model Development

Models tested:

- **Linear Regression**: Simple and interpretable, but it struggles to capture cyclic patterns and does not handle multicollinearity effectively.
- **Random Forest Regressor**: Effective for non-linear relationships, slight overfitting.
- **XGBRegressor**: High predictive power, minimal overfitting, but computationally intensive.

Selected Model: **Random Forest Regressor** for balanced performance.

Further Development: Incorporate additional data sources such as weather and traffic to enhance model accuracy. Experiment with more advanced models like **XGBRegressor** and fine-tune hyperparameters using **Grid Search** or **Random Search**. Apply **Cross-Validation** for robust model evaluation, ensuring better generalization. Monitor learning curves to track how model performance evolves with varying data sizes and training iterations.

5. Model Prototype for Driver Recommendations

Developed a solution prototype with the following features:

- **Recommended Pickup Cluster**: Area with the highest demand.
- **Expected Waiting Time**: Estimated wait time for a customer.
- **Expected Ride Value**: Anticipated earnings from the new location.
- **Performance Comparison**: Metrics on earnings and wait time changes.

6. Deployment and Communication Strategy

Deployment Tools:

- **Model**: Built using Python and Streamlit for an interactive UI.
- **Database**: PostgreSQL to store demand data and predictions.
- **Monitoring**: Grafana for real-time data and performance visualization.
- **Containerization**: Docker Compose to run services.

Driver Communication:

- **User-Friendly Interface**: Streamlit dashboard to display real-time demand hotspots.
- **Real-Time Updates**: Notifications and visual indicators for high-demand areas.
- **Clear Metrics**: Simplified metrics for driver comprehension and decision-making.

7. Experiment Design for Validation

A/B Testing:

- **Groups**: Control (no recommendations) vs. Treatment (with recommendations).
- **Metrics**: Average wait time, rides completed, driver earnings.
- **Data Collection**: Over 4-6 weeks to capture demand variability.
- **Analysis**: Statistical tests to confirm the recommendation system's impact on driver performance.