Ride Aunty Documentation

Problem Statement:

The problem at hand involves developing a ride-hailing price prediction system to assist users in comparing prices across different ride-hailing applications within Singapore. The challenge lies in accurately predicting ride fares based on various factors such as the origin and destination regions within Singapore, the time of the day, and the specific ride-hailing application chosen by the user.

Objective:

The objective of this project is to build a robust predictive model that can accurately estimate ride fares for different ride-hailing apps based on the user's input regarding the origin, destination, and time of travel. By achieving this objective, we aim to provide users with valuable insights into pricing dynamics across multiple ride-hailing platforms, thereby empowering them to make informed decisions when choosing transportation options within Singapore.

Dataset:

The dataset used in this project was obtained from a Reddit post on r/singapore titled "I collected 60,868 price points from 3 popular taxi apps in Singapore" (https://www.reddit.com/r/singapore/comments/18wpk37/i_collected_60868_price_points_from_3_popular/). The data includes information on prices collected from three popular ride-hailing applications in Singapore: Tada, Gojek, and Zig. The dataset was collected programmatically from various locations around Singapore to provide a comprehensive understanding of ride fares across different regions and times.

Insights:

The analysis of the dataset unveiled significant insights into ride fares and surge pricing across three prominent ride-hailing apps. Notably, Tada emerged as the most economical option overall, with Gojek following closely behind, and Zig exhibiting the highest fare variability, indicating dynamic pricing strategies. Moreover, the examination emphasized the prevalence of surge pricing, particularly evident in Gojek and Zig, where surge indicators were observed for a considerable portion of queries. Of particular note is Zig's frequent surge pricing occurrences, with approximately 60.19% of queries triggering surge indicators. These insights shed light on the pricing dynamics of these ride-hailing platforms, revealing their reliance on regional data derived from URA mapping borders, thereby empowering users to make informed decisions when navigating Singapore's ride-hailing landscape.

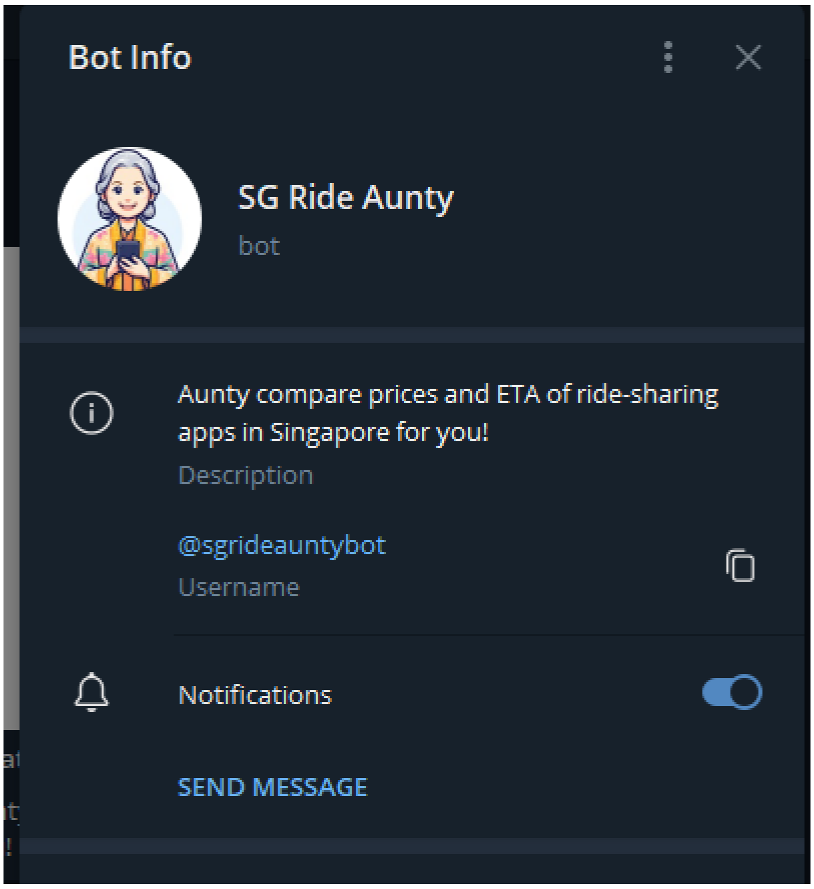
Methodology:

1) First, dummy data is generated using Python scripts. The DummyData.py script creates synthetic ride-hailing data based on constants and random distributions, including regions, ride-hailing apps, times of the day, base costs, price increases, and surge indicators. This data is then used to create a DataFrame, visualize distributions, and save the data to a CSV file.

2) Secondly, a predictive model is trained using the generated dummy data in the train_predictive_model.py script. The script loads the CSV data, preprocesses categorical features using label encoding, splits the data into training and testing sets, and trains separate Gradient Boosting Regressor models for each ride-hailing app. Model performance is evaluated using Root Mean Squared Error (RMSE), and predictions are made for each app based on user input.

3) Lastly, a Telegram bot named **RideAuntyTelegramBot** is created and initiated in the RideAuntyTelegramBot.py script. This bot uses the trained models and label encoders to predict ride prices based on user input for their current and destination regions and the time of day. The bot guides the user through a conversation flow, stores user input, performs predictions, and responds with predicted price ranges for different ride-hailing apps. The conversation can be reset at any point using the /reset command.

Link: https://t.me/sgrideauntybot


Bot Info

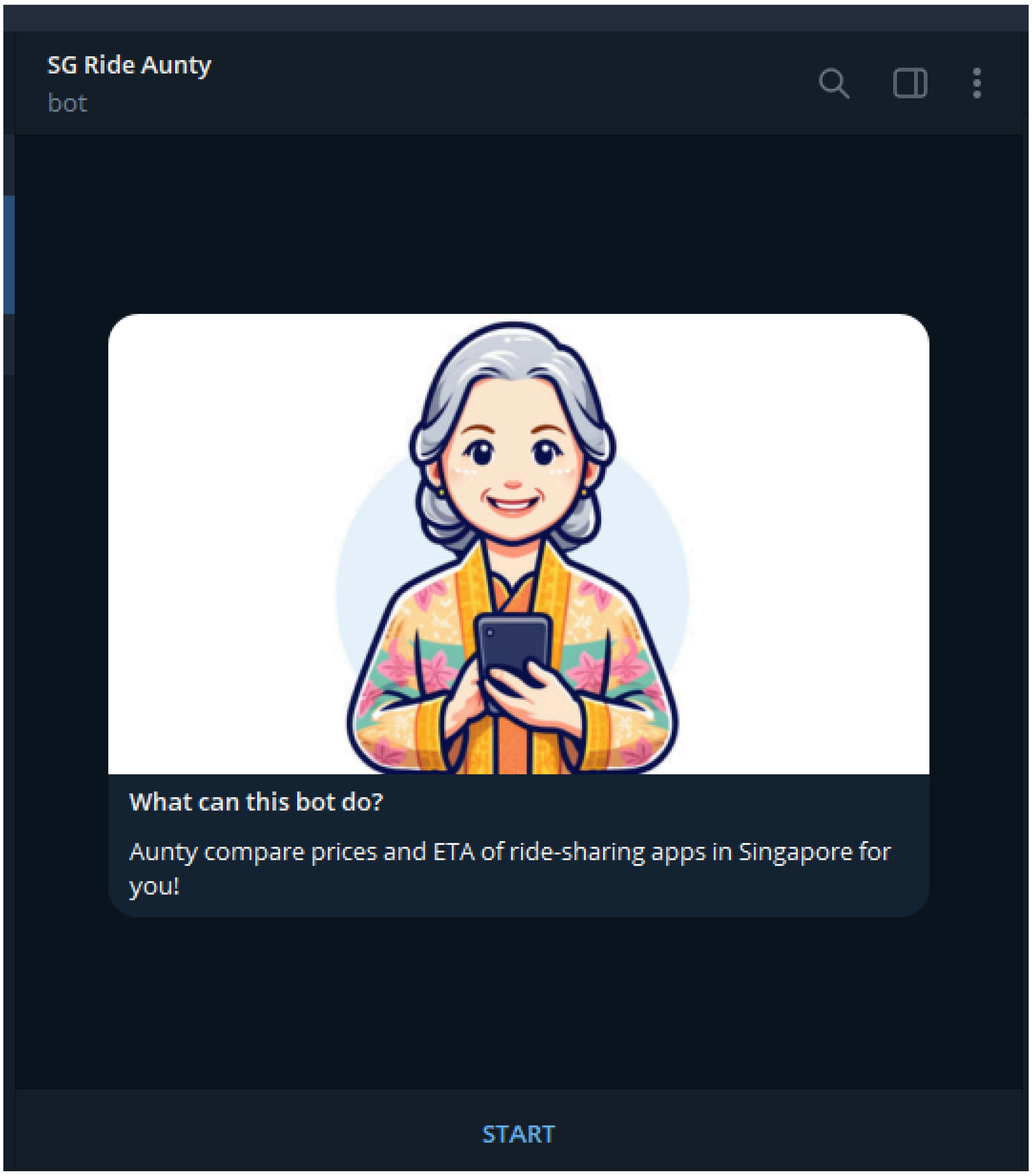
SG Ride Aunty
bot

Aunty compare prices and ETA of ride-sharing apps in Singapore for you!
Description

@sgrideauntybot
Username

Notifications

SEND MESSAGE



What can this bot do?

Aunty compare prices and ETA of ride-sharing apps in Singapore for you!

Challenges & Limitations:

One limitation of the RideAunty Telegram bot is the lack of access to proprietary pricing formulas and APIs of private ride-hailing companies. The data used for training the predictive models are sourced from publicly available datasets scraped by other individuals, restricting the accuracy and comprehensiveness of the models. Additionally, the bot is constrained to providing estimates for rides from only three platforms (TADA, GoJek, Zig) due to the unavailability of data from other major companies like Grab and Ryde. Moreover, while the bot prompts users to input specific house addresses, it only processes information based on region-to-region travel and time of day, simplifying the complexity of the task. Implementing full address-based predictions would be resource-intensive due to the need for extensive databases, although the current version serves as a proof of concept for the functionality.

Code Documentation

1. **Install Dependencies:** Before running the code, ensure that you have installed the required dependencies by executing the command **pip install -r requirements.txt**. This command will install all the necessary Python packages specified in the requirements.txt file.

2. **Run the Telegram Bot:** Execute the script RideAuntyTelegramBot.py to start the Telegram bot. This script sets up and runs the Telegram bot, allowing users to interact with it through the Telegram platform.

3. **Access the Telegram Bot:** Visit the link https://t.me/sgrideauntybot to access the RideAunty Telegram bot. This link will direct you to the chatbot interface where you can interact with the bot.

4. **Start Conversation:** Begin the conversation with the bot by sending the /start command. This will initialize the conversation and prompt the bot to ask for your current address, destination address, and time of day to provide you with estimated ride prices. Follow the prompts provided by the bot to receive price estimates for your ride.

DummyData.py

In the data generation process implemented in dummydata.py, I utilized Python libraries such as Pandas, NumPy, and Matplotlib. The dataset was constructed with 1000 rows featuring various attributes such as 'From Region', 'Destination Region', 'App', 'Time of Day', 'Cost', 'Surge Indicator', and 'Fare Type'. Base costs and price increases were defined for three ride-hailing apps: TADA, GoJek, and Zig, with associated standard deviations. Random data was generated for each attribute, considering predefined lists of regions, ride-hailing apps, and times of the day. Normal distributions were used to determine ride costs, ensuring a minimum value of $7.88. Surge indicators were determined based on predefined conditions for each app. A DataFrame was created from the generated data, and a function was implemented to visualize the distributions of various features. The DataFrame was then saved to a CSV file named 'dummy_ride_data.csv'. Additionally, a sample of the DataFrame was displayed to provide an overview of the generated data.

train_predictive_model.py

In the script train_predictive_model.py, the input variables used for training the predictive models are 'From Region', 'Destination Region', and 'Time of Day'. These features are utilized to predict the target variable 'Cost', representing the cost of ride-hailing trips. The models employed for prediction are Gradient Boosting Regressors, with separate models trained for each ride-hailing app (TADA, GoJek, Zig). Following training, the models are evaluated using root mean squared error (RMSE) to assess their performance. However, it's important to note that the models are limited by the available dataset, as all the data used for training and prediction are derived from publicly available sources, including data scraped from other sources. Moreover, the script's current implementation focuses only on predicting ride costs based on three input variables, simplifying the input data to 'From Region', 'Destination Region', and 'Time of Day'. This limitation arises from the lack of access to more granular data, such as specific house addresses, due to the unavailability of comprehensive databases. Additionally, the choice of ride-hailing apps is constrained to TADA, GoJek, and Zig, as Grab and Ryde do not publicly share their data. Despite these limitations, the script provides a proof of concept for predicting ride costs based on region and time of day, laying the foundation for future enhancements and more robust modeling approaches.

RideAuntyTelegramBot.py

In the rideauntytelgrambot.py script, a Telegram bot named "Ride Aunty" is implemented using the python-telegram-bot library version 21.1.1. The bot facilitates comparing ride prices between different regions in Singapore. It utilizes a conversation handler with states for capturing the user's current address, destination address, and time of day. The conversation begins with the /start command, prompting the user for their current address. Upon receiving the necessary inputs, the bot predicts ride prices using pre-trained models based on the user's inputs. Predicted prices are then relayed back to the user through the Telegram interface. Additionally, the bot allows for conversation reset using the /reset command. The required libraries for the predictive model, such as pandas, numpy, scikit-learn, and joblib, are imported, and the pre-trained models and label encoders are loaded from the 'models' directory. The bot is run using a Telegram Application instance with a provided bot token, and it continuously polls for updates until the user terminates the program.