Space Traffic Density Prediction

Overview

Space Traffic Density Prediction is an innovative application built using Streamlit, aimed at estimating traffic density in outer space. This tool is critical for managing the growing number of objects in space, including satellites, space debris, and spacecraft, ensuring safer navigation and operation in Earth's orbit and beyond. The application takes user inputs such as date, time, location, and object type, combines these with machine learning models, and predicts the traffic density effectively. Such a solution addresses the need for better collision prevention, resource allocation, and planning in the rapidly expanding domain of space exploration and utilization.

Features of the Application

1. **Input Parameters:**
    - **Date and Time:** The user selects a specific date and time for prediction, enabling detailed temporal analysis of space traffic.
    - **Location:** Various critical regions of space are included, such as:
        - Lagrange Point L1: A stable region between Earth and the Sun.
        - Lagrange Point L2: A stable point beyond Earth's shadow, commonly used for telescopes.
        - Low Earth Orbit (LEO): A region heavily populated with satellites.
        - Geostationary Orbit (GEO): The region where satellites orbit synchronously with Earth's rotation.
        - Medium Earth Orbit (MEO): Used primarily for navigation satellites.
        - Mars Transfer Orbit: A transitional orbit for missions to Mars.
    - **Object Type:** The user selects from various object types to predict traffic density for specific categories:
        - Space Station: Large, manned facilities like the ISS.
        - Satellite: Communication, navigation, or Earth observation satellites.
        - Scientific Probe: Spacecraft designed for exploration and data collection.
        - Space Debris: Defunct satellites and fragments posing collision risks.
        - Manned Spacecraft: Vehicles carrying astronauts.

        - Asteroid Mining Ship: Hypothetical crafts for resource extraction from asteroids.

2. **Machine Learning Models:**
    - Linear Regression
    - Ridge Regression
    - Lasso Regression
    - K-Nearest Neighbors (KNN)
    - Decision Tree
    - Random Forest

3. **Prediction Output:** Based on the user inputs, the application generates an estimated traffic density, aiding decision-making for various stakeholders in space missions.

Steps to Build the Application

1. Setting Up the Environment

Before development, the necessary environment is prepared. This involves installing required libraries for building the Streamlit application and implementing machine learning models. Libraries include:

- Streamlit: For creating the interactive user interface.
- Scikit-learn: For implementing machine learning models.
- Pandas and Numpy: For data manipulation and processing.
- Matplotlib: For optional visualizations.

Installation Command:

```
pip install streamlit scikit-learn pandas numpy matplotlib
```

2. Creating the Dataset

The accuracy of predictions depends on the dataset used. A historical dataset containing records of space traffic density is essential. Key features to include are:

- Date: The specific day of observation.
- Time: The hour and minute of observation.
- Location: The specific space region (e.g., LEO, GEO).
- Object Type: Classification of space objects present.
- Traffic Density: A numerical value indicating the density of objects in the specified location and time.

Data can be sourced from space agencies, scientific research papers, or simulated using statistical distributions. The dataset is preprocessed to handle missing values and ensure it is in a machine-readable format.

3. Developing the Application in Streamlit

Streamlit is chosen for its simplicity in creating web-based data applications. The application is built in the following stages:

a. Importing Libraries

```
import streamlit as st

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, Ridge, Lasso

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor
```

b. Building the User Interface

The interface allows users to provide input for prediction. Widgets such as date pickers, dropdowns, and buttons are implemented to make the application interactive.

```
# Title and Description

st.title("Space Traffic Density Prediction")
st.write("This application predicts traffic density in space based on input features like location, object type, and time.")


# Input Widgets
date = st.date_input("Select Date")
time = st.time_input("Select Time")
location = st.selectbox("Select Location", ["Lagrange Point L1", "Lagrange Point L2", "Orbit LEO", "Orbit GEO", "Orbit MEO", "Mars Transfer Orbit"])
object_type = st.selectbox("Select Object Type", ["Space Station", "Satellite", "Scientific Probe", "Space Debris", "Manned Spacecraft", "Asteroid Mining Ship"])


# Model Selection
model_choice = st.selectbox("Choose a Model", ["Linear Regression", "Ridge Regression", "Lasso Regression", "KNN", "Decision Tree", "Random Forest"])
```

<u>c. Model Implementation and Prediction</u>

The application allows users to select from various machine learning models. Each model is trained on the dataset, and predictions are displayed based on user inputs.

```
# Placeholder for Prediction
if st.button("Predict"):
  # Sample Data Preparation
  X = np.random.rand(100, 4)  # Replace with actual feature data
  y = np.random.rand(100)     # Replace with actual target data
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


  # Model Selection
  if model_choice == "Linear Regression":
    model = LinearRegression()
  elif model_choice == "Ridge Regression":
    model = Ridge()
  elif model_choice == "Lasso Regression":
    model = Lasso()
  elif model_choice == "KNN":
    model = KNeighborsRegressor()
  elif model_choice == "Decision Tree":
    model = DecisionTreeRegressor()
  elif model_choice == "Random Forest":
    model = RandomForestRegressor()

  # Train and Predict
  model.fit(X_train, y_train)
  prediction = model.predict([[1, 2, 3, 4]])  # Replace with actual input features
  st.write(f"Predicted Traffic Density: {prediction[0]:.2f}")
```

4. Deploying the Application

Deployment ensures the application is accessible to users. This is achieved using Streamlit's sharing platform or cloud services like AWS or Azure. The steps include:

- Save the script as app.py.
- Test locally with the command:

```
streamlit run app.py
```

- Deploy to a hosting platform for public access.

5. Enhancements and Future Scope

- **Real-Time Data Integration:** Connect with APIs like NORAD for live space object tracking.
- **Advanced Machine Learning Models:** Incorporate ensemble methods or deep learning for improved predictions.
- **Visualization:** Add graphs and heatmaps to represent traffic density visually.
- **Global Collaboration:** Enable data sharing and analysis among space agencies worldwide.

Conclusion

The Space Traffic Density Prediction application is a groundbreaking tool for addressing the challenges of increasing space traffic. By leveraging machine learning and an intuitive Streamlit interface, the application provides actionable insights for mission planning and collision prevention. It lays the foundation for more sophisticated tools in the future, ensuring the sustainable use of space resources.

Team Members

1. Shravani R S
2. Himaja S
3. Karanbir Singh
4. Tarunaa A C
5. Anushka Dutta

Date: 18th December 2024

Prepared By: Shravani R S