Smart City Traffic Analysis

Data Processing and Preparation

Overview

We sourced the datasets for this project from Kaggle, a well-known platform for data science and machine learning resources. The initial dataset was extensive, containing a large volume of information. To ensure efficiency and manageability, we reduced the dataset size by filtering and selecting only the most relevant data points for our analysis. Our data is only relevant for the year 2017.

Furthermore, the dataset included a column named WktGeom, which represented geographical data in WKT (Well-Known Text) format. To make this information more interpretable and practical for our purposes, we converted WktGeom into latitude and longitude coordinates. This transformation facilitated easier integration with mapping tools and geospatial analysis, streamlining the analytical process.

Key Functionalities

1. **Traffic Volume Analysis**
    - Creates visualizations for average traffic volume by hour, weekday, and day of the year.
    - Includes features such as trend lines, value labels, and proper axis formatting.
    - Outputs key statistics like peak hours, busiest weekday, and average daily volume.
2. **Geographical Clustering**
    - Processes geographical coordinates to remove outliers based on traffic volume.
    - Maintains the size of the dataset by resampling valid points when necessary.

3. **NYC Traffic Collision Analysis**
    - Analyzes and visualizes traffic collision data by day, weekday, and hour.
    - Implements a trend line to indicate collision trends over time.

4. **Traffic Volume Prediction Using Neural Network**

- **Data Preparation**: The dataset is efficiently prepared for machine learning by utilizing specific data types during the CSV read operation and encoding categorical variables. Time-based features such as hour, day of the week, month, weekend status, rush hour indicators, and season are generated using vectorized operations for optimal performance.
- **Model Training**: A custom neural network model, TrafficModel, is defined with multiple layers, including dropout for regularization. The model is trained using the AdamW optimizer and a Huber loss function, with the option for mixed precision training for improved performance on supported hardware. The training loop includes early stopping to prevent overfitting based on validation loss.
- **Model Evaluation**: After training, the best model is saved, and a function for making predictions is implemented. This function loads the necessary preprocessing models, scales input features, and uses the trained neural network to predict traffic volume.
- **Performance Metrics**: The model's performance is evaluated on a sample of data, calculating metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared score. Sample predictions are displayed to illustrate the model's accuracy in estimating traffic volume.
- **Visualization**: The model's predictions can be visualized to analyze seasonal variations and traffic trends, allowing for better understanding and insights into traffic patterns.

5. **Heatmap Creation**
    - Uses geographical coordinates to create a heatmap of traffic data or collision data.
    - Removes outliers to improve visualization accuracy.
    - Saves the heatmap to an HTML file for interactive exploration.
6. **Least volumed way from A to B**
    - Turning Steets into nodes and road connecting them into edges
    - Using Dijkstra's algorithm to find least occupied way
7. **Volume of cars on road**
    - Combining data to get mean result
    - Using fig to show on real map with dots how crowded road is on average
8. **Traffic Collision by Weather Type**
    - The data is sourced from a dataset containing weather and collision information.
    - The bar plot created with Seaborn displays these counts.
9. **Crashes by Precipitation Level**
    - The pie chart visualizes how the crashes are distributed across different precipitation levels

Observations and Recommendations

- **Code Organization:** While the code contains rich functionality, separating concerns into distinct modules or scripts for data preprocessing, analysis, and visualization would enhance maintainability.
- **Error Handling:** We implemented robust error handling, including mechanisms to address missing values for Latitude and Longitude in the HeatMap functions. This ensured the reliability and accuracy of our geospatial visualizations.
- **Outlier Handling:** The IQR-based outlier removal is used effectively but could benefit from parameterization to allow flexibility based on data characteristics.

- **Heatmap Improvements:** The coordinate transformation logic could be more robust by validating input formats before transformation.
- **RMSE and MAE:** current model shows considerable prediction errors. Code can be more Feature Enhanced to make lesser errors
- **Shortest Path:** nodes are placed randomly and don't really shows way to follow. Coordinates can be added for it to show at least correct way

Bonus project

House Price Prediction Model

Overview

This script builds a machine learning model to predict house prices based on features such as area, bedrooms, bathrooms, and amenities. It includes data preprocessing, exploratory data analysis (EDA), model training, evaluation, and hyperparameter tuning.

Required Libraries

- **pandas**: Data manipulation
- **numpy**: Numerical operations
- **seaborn**: Data visualization
- **matplotlib**: Plotting
- **sklearn**: Machine learning tasks

Data Loading and Preprocessing

- **Load the Dataset**: Loads data from Housing.csv.
- **Missing Values**: Fills missing values with the median of numerical columns.
- **Feature Columns**: Defines numerical and categorical features.
- **Outlier Removal**: Uses the IQR method to remove outliers from price and numerical columns.

Exploratory Data Analysis (EDA)

- **Correlation Matrix**: Visualizes relationships among numerical features.
- **Data Visualization**:
  - **Heatmap**: Displays feature correlations.
  - **Scatter Plot**: Shows relationship between area and price.
  - **Price Distribution**: Visualizes price distribution.

- **Feature Engineering**: Creates price_per_sqft for additional insight.

Model Building

- **Define Features and Target**: Sets features (X) and target variable (y).
- **Train-Test Split**: Splits data into training (80%) and testing (20%) sets.
- **Preprocessing Pipeline**: Scales numerical features and one-hot encodes categorical features.
- **Model Pipeline**: Combines preprocessing with a LinearRegression model.
- **Training and Evaluation**: Trains the model and evaluates performance using RMSE, MAE, and R^2 metrics.

Hyperparameter Tuning

- **Grid Search**: Optimizes model parameters using grid search and cross-validation.
- **Final Evaluation**: Assesses the tuned model's performance.

Model Saving

Saves the best model to house_price_model.pkl using joblib.

Conclusion

This script effectively constructs a predictive model for house prices, incorporating essential steps from data preprocessing to model evaluation, ensuring robustness and accuracy.