Price Prediction

Models

Utilizing the Random Forest algorithm for Price Range Prediction involves a systematic approach to ensure accurate modeling. The process entails several key steps:

Data Acquisition and Inspection: Begin by reading the dataset in CSV format, examining its structure, and identifying any missing values (NaN).

Feature Engineering: Address missing values using the KNNImputer from the Scikit-learn library. This technique employs the K-nearest neighbors algorithm to impute missing data effectively.

Data Preprocessing: Ensure all features are in a numerical format (integer or float), as Random Forest models require numerical input.

Feature Selection: Identify the most influential features for predicting the price range. This selection process involves analyzing the dataset to determine which attributes significantly impact the target variable.

Dataset Splitting: Employ the train_test_split() function to divide the dataset into four subsets: Training Data, Test Data, Training Data Output, and Test Data Output. This step is crucial for evaluating model performance.

Model Training: Utilize the training data to fit the Random Forest model. During this phase, the algorithm constructs multiple decision trees based on the provided data.

Model Evaluation: Assess the model's performance using the test data. Calculate the accuracy, which represents the proportion of correctly predicted price ranges.
Model Persistence: Save the trained model for future use when new data becomes available. This ensures consistency and efficiency in predictive tasks.

Following this structured methodology, a Random Forest model achieved an accuracy of 92.75%, indicating its efficacy in predicting price ranges. This accuracy metric serves as a reliable indicator of the model's predictive capability.

By adhering to these professional practices, the Random Forest algorithm proves to be a robust solution for supervised classification tasks, such as price range prediction.

ENDPOINTS

Prerequisites

For Creating endpoints I use Django Framework in which I use the default database which is sqlite.

To use end points you first follow these rules:

1. Make sure you have the model file present with name rna
2. You must install all packages that are present inside requirements.txt
    a. python -r install requirements.txt
3. Run command to create db and device table
    a. python manage.py makemigrations
    b. python manage.py migrate
4. To run server
    a. python manage.py runserver

Endpoints and its Purposes:

1. http://localhost:8000/devices/

    a. This endpoint will get list of all devices with pagination
2. http://localhost:8000/devices/1/
    a. This will get specific device which id we pass
3. http://localhost:8000/devices/add/
    a. This will add device record and predict the price range value and save it to db as well
4. http://localhost:8000/devices/bulk_add/
    a. This will add multiple records at once you just need to pass csv file which contains device records
5. http://localhost:8000/devices/predict/1/
    a. This will predict the price range of device which id is passed
    b. It also checks if it matches the previous save of the price range.
    c. If not same it will update the value of price range

To set up endpoints in Django utilizing the default SQLite database, follow these procedural guidelines:

1. **Model Configuration:**
    - Ensure that the model file named `rna` is present, defining the structure of the device table.
2. **Package Installation:**
    - Install all required packages listed in the `requirements.txt` file using the command:
        i. pip install -r requirements.txt

3. **Database Migration:**
    - Execute commands to create the necessary database schema and device table:
        i. `python manage.py makemigrations`
        ii. `python manage.py migrate`

4. **Server Initialization:**
    - Launch the Django server using the command:
        i. python manage.py runserver

Below are the endpoints along with their respective purposes:

- **List of Devices with Pagination:**
    - Endpoint: `http://localhost:8000/devices/`
    - Purpose: Retrieve a paginated list of all devices.
- **Specific Device Retrieval:**
    - Endpoint: `http://localhost:8000/devices/1/`
    - Purpose: Fetch details of a specific device by its ID.
- **Device Addition and Price Range Prediction:**
    - Endpoint: `http://localhost:8000/devices/add/`
    - Purpose: Add a new device record to the database and predict its price range. The predicted value is saved to the database.
- **Bulk Device Addition:**
    - Endpoint: `http://localhost:8000/devices/bulk_add/`
    - Purpose: Add multiple device records at once by providing a CSV file containing device details.
- **Price Range Prediction and Update:**
    - Endpoint: `http://localhost:8000/devices/predict/1/`
    - Purpose: Predict the price range of a device specified by its ID. Additionally, check if the predicted price range matches the previously saved value. If it differs, update the price range value in the database.

These endpoints facilitate various functionalities related to device management, prediction, and database interaction, enhancing the flexibility and usability of the Django application.