Code Explanation for Bike Redistribution Optimization

Introduction

The code is designed to solve the bike redistribution optimization problem using linear programming. It utilizes the `PuLP` library to formulate and solve the optimization model.

Code Components

Importing Libraries

The code begins by importing the necessary libraries:

- `pandas` for data manipulation.
- `numpy` for numerical calculations.
- `networkx` for graph representation and calculations.
- `pulp` for linear programming.

Distance Calculation

A function named `haversine` is defined to calculate the great-circle distance between two points on the Earth's surface based on their latitude and longitude. The formula used is:

$$d = 2R \cdot \arctan2(\sqrt{a}, \sqrt{1-a})$$

where R is the radius of the Earth in kilometers.

Loading Data

The code loads bike station data from a CSV file using `pandas`. The relevant columns are:

- `Station`: Name of the bike station.

- `Latitude` and `Longitude`: Geographic coordinates of the station.
- `CurNumberOfBikes`: Current number of bikes at the station.
- `MaxNumberOfBikes`: Maximum capacity of bikes at the station.

Graph Representation

A complete graph is created using `networkx`, where each station is a node, and edges represent distances between stations. The distances are calculated using the `haversine` function and are stored as weights on the edges.

Setting Up the Optimization Problem

The optimization problem is defined using `PuLP`:

- The objective is to minimize the total number of bike transfers between stations.
- Decision variables T_{ij} represent the number of bikes transferred from station i to station j and are defined as non-negative integers.

Objective Function

The objective function to minimize is defined as:

$$Z = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} T_{ij}$$

This function captures the total number of bike transfers across all stations.

Constraints

The following constraints are applied to the optimization problem:

- **Supply Constraints:** Ensure that no station transfers more bikes than it has:
$$\sum_{j=1, j \neq i}^{N} T_{ij} \leq S_i \quad \forall i$$
- **Demand Constraints:** Ensure that each station receives enough bikes to meet the average target:
$$\sum_{j=1, j \neq i}^{N} T_{ji} + S_i \geq A \quad \forall i$$

- **Non-Negativity Constraints:** Ensure all transfers are non-negative:

$$T_{ij} \geq 0 \quad \forall i, j$$

- **Integer Constraints:** Specify that the transfer variables must be integers:

$$T_{ij} \in \mathbb{Z} \quad \forall i, j$$

Solving the Model

The model is solved using `model.solve()`, and the status of the solution is printed. If an optimal solution is found, the code outputs the optimized bike transfers between stations.

Output

The results include:

- Transfers between stations in the format: "Transfer x bikes from Station i to Station j."
- The total transportation cost, which can be derived from the distance and number of bikes transferred.

Conclusion

This code effectively formulates and solves the bike redistribution problem, ensuring a balanced distribution of bikes across stations while minimizing the number of bike transfers.