Notebook 01: Understanding the ACE Intelligence Pipeline

The main goal of this notebook is to figure out if the MTA's Automated Camera Enforcement (ACE) system is actually helping buses move faster, or if we're seeing an **enforcement paradox** (lots of tickets, no speed improvement).

1. Setting the Stage and Defining the Problem

We start by loading necessary Python libraries like pandas, numpy, and matplotlib, getting the environment ready for data processing and visualization.

The notebook explicitly addresses the following key questions that guide the project:

• **The Paradox:** How exactly do we calculate the **Paradox Score**?

• **Enforcement Metrics:** What does the **Enforcement Intensity** metric measure?

• **CUNY Focus:** How does the **CUNY Proximity Analysis** use buffer zones to identify relevant stops?

2. Phase 1A: Processing Violation Data

This step takes the raw violations data and turns it into actionable metrics aggregated by *route and hour*.

• **Aggregation:** Individual violation records are grouped by `route_id` and `violation_hour`.

• **Metric Calculation:** This aggregation calculates four essential metrics:

  ◦ **`violation_count`**: The total number of violations recorded.

  ◦ **`ticketed_violations`**: The sum of violations that resulted in a ticket (to differentiate actual enforcement action).

  ◦ **`technical_issues`**: The sum of records flagged as technical issues.

  ◦ **`unique_vehicles`**: The count of different vehicles involved in violations during that route-hour.

3. Phase 1B: Incorporating Speed Data

To determine if enforcement is actually effective, we must compare violation rates with changes in speed.

• **Speed Data:** Historical and current speed data is loaded and analyzed.

• **Effectiveness Measurement:** The core calculation here is the **pre/post-ACE speed change**, which is explicitly called "crucial for measuring enforcement effectiveness". For example, sample speed data records show routes like BX1 and BX10 from 2015.

4. Phase 1C: Adding the CUNY Dimension

This phase ensures that bus routes serving critical educational infrastructure—CUNY campuses—are prioritized, tying into the Datathon's Question 1 and the **ClearLane** narrative.

• **Spatial Technique:** The analysis uses a **CUNY Proximity Analysis**.

• **Buffer Zone:** It uses the **Haversine formula** to calculate distances, identifying violations that fall within a defined **500-meter buffer zone** around CUNY campuses.

• **Purpose:** This identifies which routes are serving educational institutions, guaranteeing that "student transportation gets proper attention".

5. Phase 2 & 3: Calculating the Paradox

This is where the customized analytical metrics are created, transitioning from simple counts to deep measures of effectiveness.

A. Enforcement Intensity Score

This metric is designed to show how **concentrated** the enforcement effort is, normalizing the data beyond simple volume.

• **What it measures:** How "intense" enforcement is relative to the number of vehicles on the road.

• **Formula Logic:** It gives tickets (ticketed violations) a heavier weight (0.6) than general violation counts (0.4), and then normalizes this by the number of unique vehicles involved (plus one to prevent division by zero).

• **Interpretation:** A high score means many tickets are issued, but they are concentrated among fewer unique vehicles.

B. Paradox Score

This is the core diagnostic tool that reveals enforcement failure, directly addressing the project's central paradox.

• **Speed Improvement Factor:** First, a factor is calculated. If a route's speed improved (positive percentage change), that improvement plus one is used. If speed got worse (negative percentage change), a factor of one (no improvement) is used. This prevents dividing by zero and rewards successful speed changes.

• **Core Formula:** The Paradox Score is calculated as: **(violation count × enforcement intensity) / speed improvement factor**.

• **Interpretation:** A **HIGH** score is the paradox itself: it means there are lots of violations and high enforcement intensity, but **NO** speed improvement (the enforcement is not working). A **LOW** score means that violations *are* leading to speed improvements.

C. Overall Paradox Ranking

To create a single, definitive measure of failure, the analysis combines several aspects into a single rank.

• **Weighting:** The overall rank is weighted as follows:
  ◦ **50%** weight on the Paradox Score (violations versus speed change).
  ◦ **30%** weight on Enforcement Efficiency (how well enforcement works).
  ◦ **20%** weight on Temporal Volatility (consistency of performance).

• **Interpretation:** A **HIGHER rank** means the route is **MORE paradoxical** (the enforcement system is failing most severely there). For instance, a sample ranking shows route BX36 with a rank of 0.876, indicating it is highly paradoxical.

6. Visualizations and Next Steps

The notebook demonstrates how visualizations help quickly grasp the calculated paradox.

• **The Visualization:** A scatter plot shows `Violation Count` versus `Speed Change (%)`, with points colored by the `Paradox Score`.

• **Visualization Insight:** This plot includes a red dashed line at zero speed improvement. Routes below that line show that speed got worse after ACE implementation, despite enforcement efforts. Higher paradox scores visually confirm that enforcement is ineffective.

• **Conclusion:** Notebook 01 proves that the core idea works. The "paradox score" is a "clear, data-driven way to calculate it on the real datasets". The next step in the project (Notebook 02) is to run this pipeline on the full 3.78 million record dataset to identify the true patterns. This ultimately transitions the project from looking backward to predicting the future.