Deliverable 2 (Revised): ML-Powered Emission Analysis and Airline Ranking Dashboard

This deliverable details a comprehensive machine learning and data pipeline solution for predicting CO emissions and ranking airline efficiency. It has evolved from a basic script into a full-fledged, interactive Streamlit application. The system now dynamically collects data from multiple sources, trains a suite of advanced ML models, and provides in-depth analysis through a user-friendly interface.

0.1 Data Collection and Preparation

1. **Dynamic Data Sourcing**: Instead of loading a static CSV, the system, via the `EFlightCarbonCalculator` class, collects data from multiple sources to create a rich, comprehensive dataset.
   - **AviationStack API**: Fetches real-world historical and scheduled flight data.
   - **OpenSky Network**: Gathers real-time flight state vectors for live tracking insights.
   - **Synthetic Data Generation**: Creates a large volume of realistic flight routes to ensure model robustness and broad coverage.
2. **Advanced Feature Engineering**: Raw data is processed to create meaningful features for the ML models.
   - **Sophisticated CO Calculation**: Emissions are not a simple target variable but are calculated using a detailed formula incorporating aircraft type, load factor, route distance, flight type (domestic/international), and penalties for takeoff/landing cycles.
   - **Engineered Features (X)**: The feature set is significantly expanded to capture non-linear relationships.
     - **Core Features**: `distance_km`, `passenger_capacity`, `load_factor`.
     - **Polynomial & Transformed Features**: `distance_squared`, `distance_log`.
     - **Interaction Features**: `distance_capacity_interaction`, `distance_load_interaction`.
     - **Categorical Features**: One-hot encoded `aircraft_category` (e.g., narrow-body, wide-body) and a binary `is_domestic_encoded` flag.
   - **Target (y)**: The primary prediction target remains `co2_per_passenger_kg`, which is calculated during the data preparation phase.

0.2 Training Advanced ML Models for Prediction

1. **Multi-Model Approach**: The system moves beyond a single OLS model to train and compare a suite of powerful regression algorithms from `scikit-learn` and `statsmodels`. This allows for a more accurate and nuanced prediction by capturing both linear and complex non-linear patterns. The models include:
   - Linear Regression

- Random Forest Regressor
- Gradient Boosting Regressor
- Support Vector Regressor (SVR)
- Ordinary Least Squares (OLS) (for baseline comparison)

2. **Model Evaluation and Comparison**:
   - The dataset is properly split into training and testing sets using `train_test_split` for robust validation.
   - Performance is evaluated using metrics like **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R^2)**.
   - Results are displayed in a comparison table and a multi-model scatter plot of actual vs. predicted emissions.

3. **Interactive Prediction**: The Streamlit interface allows a user to input flight details (origin, destination, aircraft type). The application then provides emission predictions from **all trained ML models** alongside the formula-based calculation, offering a comprehensive view.

4. **Visualization and Interpretability**:
   - **Actual vs. Predicted Plot**: A scatter plot visualizes the performance of all models against a "perfect prediction" line.
   - **Feature Importance**: A bar chart shows the most influential features as determined by the Random Forest model, providing insight into the key drivers of emissions.
   - **OLS Summary**: The detailed statistical summary from the `statsmodels` OLS model is provided for in-depth analysis of coefficients and p-values.

0.3 Ranking Airlines by Efficiency

1. **Robust Aggregation**:
   - The `compare_airlines_efficiency` logic is enhanced for reliability. It first filters the data to include only airlines with a minimum of **10 flights** in the dataset, preventing skewed results from insufficient data.
   - It then groups the data by `airline_name` and aggregates key metrics, including the mean `co2_per_passenger_per_km`.

2. **Enhanced Visualization**: The ranking is displayed using two separate Plotly bar charts: "Top 10 Most Efficient Airlines" and "Top 10 Least Efficient Airlines," providing a clear and immediate understanding of performance.

3. **Benchmark Against Industry Reports**: The application programmatically includes sustainability data for major airlines. A comparative bar chart directly visualizes the **calculated efficiency** from the dataset against the **self-reported emissions intensity** from airline ESG reports, offering a valuable validation and accountability check.

0.4 Integrated Streamlit Application Pipeline

The entire workflow is encapsulated in a sophisticated, multi-tab Streamlit application, replacing the simple pipeline script.

1. **On-Demand Pipeline Execution**: The pipeline is triggered via a button in the "Data Collection" tab. A user can configure the desired dataset size, and the app executes the full data collection, processing, calculation, and cleaning workflow, showing progress with a spinner.
2. **Multi-Tab Dashboard**: The application is organized into logical tabs for a structured user experience:
   - **Data Collection**: Configure and run the data gathering pipeline.
   - **Dataset Analysis**: Explore the generated dataset with summary statistics and visualizations (histograms, correlation matrices, box plots).
   - **ML Models**: Train the suite of ML models and view detailed performance comparisons.
   - **Predictions**: Use the interactive tool to predict emissions for a specific flight route.
   - **Airline Comparison**: View the efficiency rankings and benchmark against sustainability reports.
3. **Session State Management**: The collected DataFrame and trained models are stored in Streamlit's `session_state`, ensuring data persistence across tabs and preventing redundant computations.

0.5 Application Structure

The previous two-notebook structure (`emission_prediction.ipynb`, `airline_ranking.ipynb`) is now obsolete. All functionalities—data collection, analysis, model training, prediction, and ranking—are integrated into a **single, unified Python script** that runs the Streamlit application. This creates a cohesive and maintainable codebase.

0.6 Documentation and Limitations

- **Models**: The prediction engine uses a multi-model framework. The relationship is no longer a simple linear assumption but a complex function learned by algorithms capable of capturing non-linear interactions:

$$CO_2 \sim f(\text{distance, capacity, load\_factor, aircraft\_category}, \dots)$$

- **Ranking**: The ranking methodology is based on the mean CO per passenger-kilometer. Its reliability is improved by a flight count filter ($n \geq 10$).
- **Pipeline**: The pipeline is an on-demand, user-triggered workflow within an interactive Streamlit dashboard. It can be scheduled for automation (e.g., via cron) if deployed on a server environment.
- **Limitations**:
  - **API Dependency**: The system is dependent on external APIs (AviationStack, OpenSky) and requires valid API keys. It is susceptible to rate limits or service changes.
  - **Synthetic Data Quality**: While designed to be realistic, the synthetic data may not capture all the nuances of real-world airline operations.

- **Simplifying Assumptions**: The logic for determining domestic vs. international flights and mapping callsigns to airlines is based on simplified heuristics.
- **Data Granularity**: The model does not yet incorporate hyper-granular data like real-time weather, flight altitude profiles, or specific air traffic control routing, which are known to affect fuel burn.