🔧 Step 1: Data Ingestion and Storage

Notebook: 01\_data\_ingestion.ipynb  
Script: src/ingestion.py  
Tools: PySpark, Pandas, HDFS/S3

Tasks:

* Load 12 monthly Parquet files using PySpark into a unified DataFrame.
* Load latitude/longitude zone mapping CSV using Pandas.
* Merge if needed and save combined DataFrame to data/processed/ as full\_trip\_data.parquet.

This step sets the foundation with scalable, distributed loading.

Step 2: Data Cleaning & Preprocessing

Notebook: 02\_preprocessing.ipynb

Script: src/preprocessing.py  
Tools: PySpark

Tasks:

* Drop nulls and filter invalid entries (e.g., negative or zero lat/lon, impossible durations).
* Extract time features: hour, day, weekday, month from pickup datetime.
* Calculate trip distance using Haversine formula.
* Calculate speed = distance / duration.
* Remove outliers based on thresholds (e.g., speed > 100 mph).

Resulting cleaned data is ready for advanced feature engineering.

Step 3: Feature Engineering & PCA

Notebook: 03\_feature\_engineering\_pca.ipynb  
 Script: src/features.py  
Tools: PySpark, scikit-learn

Tasks:

* Add engineered features: pickup/dropoff lat/lon, duration, speed, time of day, etc.
* Normalize features.
* Apply PCA to reduce dimensions while preserving 95% variance.

Reduces noise and computational burden in downstream models.

Step 4: Clustering (Red/Yellow/Green Zones)

Notebook: 04\_clustering\_zones.ipynb  
Script: src/clustering.py  
Algorithms: KMeans

Tasks:

* Use pickup and dropoff locations to cluster trips.
* Assign each cluster a zone ID.
* Label clusters based on average speed:
  + Red: Congested
  + Yellow: Moderate
  + Green: Free-flowing

Creates spatial segmentation to analyze traffic by zones.

Step 5: Time-Series Prediction (Traffic Speed Over Time)

Notebook: 05\_time\_series\_prediction.ipynb  
Script: src/prediction.py  
Models: XGBoost, LSTM, Prophet

Tasks:

* Feature engineering for temporal patterns:
  + Lag features (1h, 6h, 24h)
  + Rolling means/std
  + Calendar features (weekend, holiday)
  + Spatial context (zone neighbor speed)
* Train models to predict average speed per zone for the next 1–24 hours.
* Use ensemble of XGBoost + LSTM + Prophet with weighted averaging.

Captures both short-term and long-term traffic trends.

Step 6: Visualization on NYC Map

Notebook: 06\_map\_visualization.ipynb  
Script: src/visualization.py  
Tools: Folium, Kepler.gl, Plotly

Tasks:

* Visualize clustered zones on a NYC map (Red/Yellow/Green).
* Animate traffic changes over time using a time slider.
* Export as HTML map for interactive use.

Communicates insights effectively and enables spatial validation.

Step 7: Live Traffic Comparison

Notebook: 07\_live\_api\_comparison.ipynb  
Script: src/live\_api\_comparison.py  
APIs: Google Maps, TomTom, HERE

Tasks:

* Query live traffic speed for zone pairs via external API.
* Compare live with model-predicted.
* Compute accuracy/deviation to monitor drift.

Enables real-time validation and retraining triggers.

Dashboard

File: dashboard/app.py  
Tool: Streamlit

Features:

* Zone-wise visualization
* Interactive filters (time, date, zone)
* Real-time vs predicted trafficcomparison

Ideal for stakeholders or traffic managers.

|  |  |
| --- | --- |
| **Component** | **You Have It** |
| **Scalable Data Processing** | PySpark handles massive NYC trip datasets efficiently |
| **Data Cleaning/Validation** | Preprocessing for outliers, nulls, invalid coordinates |
| **Advanced Feature Engineering** | Time-based, spatial, lag, and external features (e.g., weather, events) |
| **Dimensionality Reduction** | PCA for reducing noise and improving model performance |
| **Clustering & Spatial Analysis** | KMeans for traffic zone discovery and congestion labeling |
| **Predictive Modeling** | XGBoost, LSTM, Prophet — a powerful ensemble of time-series forecasters |
| **Live Data Integration** | Real-time comparison using traffic APIs (Google, HERE, etc.) |
| **Visualization & UI** | Streamlit dashboard + HTML maps via Folium/Kepler.gl |
| **Automation Readiness** | Pipeline can be modularized, scheduled, and monitored |
| **Interpretability** | Visual diagnostics, feature importance, clustering maps |