**AI-Powered NYC Taxi Demand Prediction and Fleet Optimization**

**1. Introduction**

**1.1 Project Background**

AI-Powered NYC Taxi Demand Prediction and Fleet Optimization is developed as part of an AI course to address inefficiencies in ride-sharing systems, specifically for NYC taxi drivers. The initial project proposal, identified a key issue: drivers often cluster in busy areas like Manhattan, leading to oversupply in some zones and undersupply in others. This imbalance causes longer wait times for passengers in underserved areas and reduces earnings for drivers in oversaturated zones. The proposal aimed to reduce passenger wait times by 15% and increase driver earnings by using machine learning (ML) to predict demand hotspots and recommend repositioning strategies.

The final project significantly expanded beyond the proposal's scope. We implemented a comprehensive data preprocessing pipeline, developed advanced ML models (Prophet, Random Forest, XGBoost, K-means, DBSCAN), performed feature selection, conducted clustering, compared model performance, and built an interactive Streamlit dashboard with detailed features for both drivers and technical users. This report provides an exhaustive overview of the project, including data processing, model development, dashboard features, and logic.

**1.2 Team Members**

* **Muneeb Ahmed (616994)**:
  + Data preprocessing (preprocessing.ipynb)
  + Feature selection (feature\_selection.ipynb)
* **Suzan Mwemeke (616956)**:
  + Demand forecasting (forecasting.ipynb)
  + Model comparison (compare\_models.ipynb)
* **Md Abu Saleh (616974)**:
  + Classification and clustering (classification\_clustering.ipynb)
  + Dashboard development (dashboard.py)
  + Project coordination

**1.3 Objectives**

* Predict demand hotspots in NYC using historical taxi data.
* Recommend optimal zones for drivers to reposition, balancing supply and demand.
* Reduce passenger wait times by ensuring better driver distribution.
* Increase driver earnings by guiding them to high-demand, high-earning zones.
* Provide an interactive dashboard with detailed visualizations and recommendations for drivers and technical users.

**2. Problem Statement**

In NYC, taxi drivers often concentrate in busy areas, leading to an oversupply in some zones and an undersupply in others. This causes longer wait times for passengers in underserved areas and reduces earnings for drivers who struggle to find passengers in oversaturated zones. The project aims to predict demand hotspots, recommend repositioning strategies, and balance supply and demand to reduce wait times by 15% (as per the proposal) and increase driver earnings. The dashboard provides actionable insights through forecasts, high-demand zone identification, clustering, earnings predictions, and visualizations.

**3. Project Scope**

**3.1 Initial Scope (From Proposal)**

* Analyze historical NYC taxi data to forecast demand patterns.
* Use ML models like Prophet for time-series forecasting and Random Forest for hotspot identification.
* Develop rule-based repositioning logic.
* Build a dashboard with heatmaps and driver recommendations.
* Incorporate supplemental data like Uber Movement Data for traffic patterns and weather data via API (optional).

**3.2 Expanded Scope (Final Work)**

* **Data Preprocessing**: Cleaned and processed 5,335,512 rows of taxi data, added weather and holiday features.
* **Feature Engineering**: Created extensive features for demand, earnings, and clustering.
* **Model Development**:
  + Used Prophet for demand forecasting.
  + Implemented Random Forest and XGBoost for high-demand zone classification.
  + Developed Random Forest Regressor for earnings prediction.
  + Applied K-means and DBSCAN for clustering zones.
* **Feature Selection**: Used Recursive Feature Elimination (RFE) to select the best features.
* **Model Comparison**: Evaluated all models to select the best ones for the dashboard.
* **Dashboard Development**: Built a comprehensive Streamlit dashboard with two pages:
  + **Driver Dashboard**: Includes demand forecasts, high-demand zone maps, recommendations, heatmaps, and personalized tips.
  + **Technical Dashboard**: Provides model metrics, zone-specific forecasts, cluster visualizations, and impact simulations.
* **Additional Features**: Added earnings adjustments, weather impact analysis, driver preference tracking, and detailed visualizations.

**4. Data Sources and Description**

**4.1 Datasets Used**

* **NYC Taxi & Limousine Commission (TLC) Trip Records**:
  + **Source**: NYC TLC Data Portal.
  + **File**: data/raw/nyc\_taxi\_data.csv.
  + **Columns**:
    - tpep\_pickup\_datetime, tpep\_dropoff\_datetime: Pickup and dropoff timestamps.
    - PULocationID, DOLocationID: Pickup and dropoff zone IDs.
    - passenger\_count: Number of passengers per trip.
    - total\_amount: Total fare paid (in dollars).
    - trip\_distance: Distance of the trip (in miles).
    - VendorID: Driver ID.
    - pickup\_lat, pickup\_lon, dropoff\_lat, dropoff\_lon: Coordinates of pickup and dropoff locations.
    - Pickup\_Zone: Name of the pickup zone.
  + **Size**: 5,335,512 rows after cleaning.
* **Weather Data**:
  + **Source**: Not specified (assumed external API as per proposal).
  + **Features**:
    - temperature: Temperature in degrees Celsius.
    - precipitation: Rainfall amount (assumed in mm).
    - wind\_speed: Wind speed (assumed in km/h).
  + **Integration**: Merged with taxi data by date and hour.
* **Holiday Data**:
  + **Source**: Manually curated list of US holidays for 2025.
  + **Features**:
    - is\_holiday: Binary flag (1 for holidays, 0 otherwise).
  + **Holidays**:
    - New Year's Day (2025-01-01), Martin Luther King Jr. Day (2025-01-20), Presidents' Day (2025-02-17), Memorial Day (2025-05-26), Independence Day (2025-07-04), Labor Day (2025-09-01), Columbus Day (2025-10-13), Veterans Day (2025-11-11), Thanksgiving Day (2025-11-27), Christmas Day (2025-12-25).
* **Uber Movement Data**:
  + **Note**: Mentioned in the proposal for traffic patterns but not used due to availability issues. Used avg\_speed as a proxy for traffic conditions.

**4.2 Data Challenges**

* **Missing Values**: Some rows lacked coordinates or fares, requiring imputation or removal.
* **Outliers**: Negative fares, zero-distance trips, or fares above $100 were removed.
* **Data Volume**: Large dataset (5,335,512 rows) required efficient processing with caching in Streamlit.
* **Weather Data Alignment**: Ensured weather data matched taxi data timestamps.
* **Traffic Data Absence**: Compensated by calculating avg\_speed from trip data.

**5. Methodology**

**5.1 Data Collection and Preprocessing (preprocessing.ipynb)**

**Steps:**

1. **Load Data**:
   * Loaded nyc\_taxi\_data.csv using Pandas.
2. **Clean Timestamps**:
   * Converted tpep\_pickup\_datetime and tpep\_dropoff\_datetime to datetime format.
   * Extracted features: hour (0-23), day\_of\_week (0-6, Monday=0), month (1-12).
3. **Remove Invalid Data**:
   * Dropped rows with missing PULocationID, total\_amount, or coordinates.
   * Removed trips with:
     + Negative or zero total\_amount.
     + total\_amount > $100 (assumed outliers).
     + Zero trip\_distance or duration (computed as tpep\_dropoff\_datetime - tpep\_pickup\_datetime).
   * Resulted in 5,335,512 valid rows.
4. **Add Weather Data**:
   * Merged weather data by matching date and hour.
   * Features: temperature, precipitation, wind\_speed.
5. **Add Holiday Data**:
   * Marked trips as is\_holiday (1 for holidays, 0 otherwise) based on the 2025 US holiday list.
6. **Compute Features**:
   * avg\_speed = trip\_distance ÷ (tpep\_dropoff\_datetime - tpep\_pickup\_datetime) in miles per hour.
   * temp\_precip = temperature × precipitation to capture combined weather effects.
7. **Save Cleaned Data**:
   * Saved to data/processed/cleaned\_taxi\_data.csv.

**Output:**

* Cleaned dataset with columns: PULocationID, pickup\_datetime, hour, VendorID, temperature, precipitation, wind\_speed, is\_holiday, avg\_speed, day\_of\_week, month, temp\_precip, pickup\_lat, pickup\_lon, total\_amount, passenger\_count, trip\_duration, Pickup\_Zone.

**5.2 Feature Engineering**

**Features Created:**

* **Time-Based Features**:
  + hour: Hour of the day (0-23).
  + day\_of\_week: Day of the week (0-6, Monday=0).
  + month: Month of the year (1-12).
* **Weather Features**:
  + temperature, precipitation, wind\_speed.
  + temp\_precip: Interaction term (temperature × precipitation).
* **Holiday Feature**:
  + is\_holiday: Binary flag (1 for holidays, 0 otherwise).
* **Traffic Feature**:
  + avg\_speed: Average speed of trips (miles/hour).
  + avg\_speed\_std: Standard deviation of speed per zone, hour, and day of week (for congestion).
* **Demand Features**:
  + pickup\_count: Number of pickups per zone, hour, and day of week.
  + driver\_count: Number of unique drivers per zone, hour, and day of week.
  + current\_demand\_supply\_ratio = pickup\_count ÷ driver\_count.
* **Earnings Features**:
  + distance: Geodesic distance (in miles) from the driver’s current location to each zone, calculated using geopy.distance.geodesic.
  + travel\_time = distance ÷ avg\_speed × 60 (in minutes).
  + trips\_per\_hour = 60 ÷ trip\_duration, capped at 3 trips per hour.
  + gross\_earnings = trips\_per\_hour × avg\_fare × (1 + 0.1 × passenger\_count).
  + fuel\_cost = distance × $0.5 per mile.
  + net\_earnings = gross\_earnings × (60 - travel\_time - 15) ÷ 60 - fuel\_cost (15 minutes idle time assumed).
  + congestion\_factor = avg\_speed\_std ÷ avg\_speed.
  + preference\_score: Driver preference for each zone (tracked via session state in the dashboard, initialized to 0).
  + adjusted\_earnings = net\_earnings × (1 + 0.1 × preference\_score) × (1 - 0.5 × congestion\_factor) (if Random Forest Regressor unavailable).
* **Forecasting Features**:
  + forecasted\_demand: Predicted pickups using Prophet.
  + demand\_slope: Change in forecasted demand over time.
  + prophet\_confidence = (yhat\_upper - yhat\_lower) ÷ forecasted\_demand, clipped to [0, 1].
* **Clustering Features**:
  + pickup\_count\_raw: Total pickups per zone (not aggregated by hour).
  + label: Cluster label (e.g., "High-Demand Urban").

**Zone Features:**

* Aggregated per zone: pickup\_count, total\_amount, pickup\_lat, pickup\_lon, trip\_duration, passenger\_count, avg\_fare (capped at $30).

**5.3 Model Development**

**5.3.1 Demand Forecasting (forecasting.ipynb)**

**Objective:**

Predict city-wide and zone-specific passenger demand for the next 24 hours.

**Model: Prophet**

* **Description**: Prophet is a time-series forecasting model by Facebook, designed for data with trends, seasonality, and holidays.
* **Why Chosen**: Suitable for hourly taxi demand data with daily/weekly patterns and holiday effects.
* **Development Steps**:
  1. **Data Preparation**:
     + Grouped data by hour to compute total pickups (y).
     + Created ds (datetime) column for Prophet.
     + Added regressors: temperature, precipitation, is\_holiday, hour, day\_of\_week.
  2. **Model Configuration**:
     + Enabled daily and weekly seasonality.
     + Added holidays from the 2025 US holiday list.
     + Included regressors to improve predictions.
  3. **City-Wide Model**:
     + Trained on city-wide hourly pickup counts.
     + Saved as prophet\_model.pkl.
  4. **Zone-Specific Models**:
     + Trained separate models for the top 10 zones by pickup volume.
     + Saved as prophet\_zone\_{zone\_id}.pkl.
  5. **Forecasting**:
     + Predicted 288 rows (12 days hourly).
     + Computed errors: APE = |y - yhat| ÷ y × 100.
  6. **Results**:
     + MAE: 644.59 pickups.
     + RMSE: 934.30 pickups.
     + MAPE: 27.22%.
     + Median APE: 15.28%.
     + High APE Hours (>100%): 7.
  7. **Saved Output**:
     + Forecasts saved to prophet\_forecasts.csv.

**Challenges:**

* High MAPE (27.22%) due to sudden demand spikes.
* Addressed by keeping Prophet for its robustness in time-series data, noting improvement areas.

**5.3.2 Feature Selection (feature\_selection.ipynb)**

**Objective:**

Select the best features for high-demand zone classification to improve model performance.

**Method: Recursive Feature Elimination (RFE)**

* **Description**: RFE iteratively removes the least important features using a base model (Random Forest).
* **Development Steps**:
  1. **Data Preparation**:
     + Loaded cleaned data.
     + Defined features: temperature, precipitation, wind\_speed, is\_holiday, avg\_speed, day\_of\_week, hour, month, temp\_precip.
     + Target: is\_high\_demand (1 if pickups > 80th percentile, 0 otherwise).
  2. **RFE Setup**:
     + Base Model: Random Forest Classifier.
     + Number of features to select: Determined iteratively.
  3. **Execution**:
     + RFE ranked features by importance.
     + Selected features: precipitation, is\_holiday, avg\_speed, day\_of\_week, temp\_precip.
  4. **Saved Output**:
     + Saved to selected\_features.csv.

**Why Important:**

* Reduced feature set improves model speed and prevents overfitting.

**5.3.3 High-Demand Zone Classification (classification\_clustering.ipynb)**

**Objective:**

Classify zones as high-demand (pickups > 80th percentile) or not.

**Models:**

* **Random Forest Classifier**:
  + **Description**: Ensemble model using multiple decision trees.
  + **Why Chosen**: Handles non-linear relationships and provides feature importance.
  + **Development Steps**:
    1. **Data Preparation**:
       - Aggregated pickups by zone and hour.
       - Defined is\_high\_demand based on the 80th percentile.
       - Used RFE-selected features to avoid leakage.
    2. **Hyperparameter Tuning**:
       - Used GridSearchCV with 5-fold cross-validation.
       - Parameters:
         * n\_estimators: [100, 200, 300].
         * max\_depth: [10, 20, None].
         * min\_samples\_split: [2, 5].
         * min\_samples\_leaf: [1, 2].
       - Scoring: F1-score.
    3. **Training**:
       - Split data: 80% training, 20% testing.
       - Trained on training set, evaluated on test set.
    4. **Metrics**:
       - Accuracy: 91.02%.
       - Precision: 80.00%.
       - Recall: 74.58%.
       - F1-Score: 77.19%.
       - ROC-AUC: 96.39%.
    5. **Saved Model**:
       - Saved as rf\_model.pkl.
* **XGBoost Classifier**:
  + **Description**: Gradient boosting model, faster and often more accurate than Random Forest.
  + **Why Chosen**: Excels in classification tasks with imbalanced data.
  + **Development Steps**:
    1. **Data Preparation**:
       - Same as Random Forest.
    2. **Hyperparameter Tuning**:
       - Used GridSearchCV with 5-fold cross-validation.
       - Parameters:
         * n\_estimators: [100, 200].
         * max\_depth: [3, 5, 7].
         * learning\_rate: [0.01, 0.1].
       - Scoring: F1-score.
    3. **Training**:
       - Split data: 80% training, 20% testing.
       - Trained on training set, evaluated on test set.
    4. **Metrics**:
       - Accuracy: 90.32%.
       - Precision: 84.62%.
       - Recall: 74.58%.
       - F1-Score: 79.30%.
       - ROC-AUC: 96.68%.
    5. **Saved Model**:
       - Saved as xgb\_model.pkl.

**Challenges:**

* **Data Leakage**: Avoided by excluding pickup\_count from features.
* **Class Imbalance**: High-demand zones were fewer; addressed by focusing on F1-score.

**5.3.4 Earnings Prediction (classification\_clustering.ipynb)**

**Objective:**

Predict earnings per zone to recommend the best areas for drivers.

**Model: Random Forest Regressor**

* **Description**: Ensemble model for regression, using multiple decision trees.
* **Why Chosen**: Handles complex relationships and provides feature importance.
* **Development Steps**:
  1. **Data Preparation**:
     + Features: pickup\_count, driver\_count, temperature, precipitation, wind\_speed, is\_holiday, avg\_speed, day\_of\_week, temp\_precip, travel\_time, passenger\_count, avg\_fare, current\_demand\_supply\_ratio.
     + Target: adjusted\_earnings (calculated as per feature engineering).
  2. **Hyperparameter Tuning**:
     + Used GridSearchCV with 5-fold cross-validation.
     + Parameters:
       - n\_estimators: [100, 200].
       - max\_depth: [10, 20, None].
       - min\_samples\_split: [2, 5].
       - min\_samples\_leaf: [1, 2].
     + Scoring: Negative Mean Squared Error.
  3. **Training**:
     + Split data: 80% training, 20% testing.
     + Trained on training set, evaluated on test set.
  4. **Metrics**:
     + MSE: 14.48.
     + R²: 94.03%.
  5. **Feature Importance**:
     + travel\_time: 85.10%.
     + avg\_fare: 5.57%.
     + avg\_speed: 4.22%.
     + passenger\_count: 2.18%.
  6. **Saved Model**:
     + Saved as rf\_earnings\_model.pkl.

**Challenges:**

* **Missing Preferences**: No driver preference data; used placeholder (preference\_score = 0).
* **Feature Complexity**: Ensured all features were available for prediction.

**5.3.5 Clustering (classification\_clustering.ipynb)**

**Objective:**

Group zones into clusters for better insights (e.g., "High-Demand Urban").

**Models:**

* **K-means**:
  + **Description**: Partitions data into a fixed number of clusters.
  + **Why Chosen**: Simple and effective for spatial data.
  + **Development Steps**:
    1. **Data Preparation**:
       - Features: pickup\_count\_raw, total\_amount, pickup\_lat, pickup\_lon, hour.
       - Scaled features using MinMaxScaler.
    2. **Model Configuration**:
       - Number of clusters: 4 (chosen empirically).
       - Random state: 42 for reproducibility.
    3. **Training**:
       - Applied K-means to scaled data.
    4. **Metrics**:
       - Silhouette Score: 0.62 (indicates good clustering).
    5. **Labeling**:
       - "High-Demand Urban": Pickups > 75th percentile.
       - "High-Fare": total\_amount > 75th percentile.
       - "Residential": Other zones.
  + **Saved Output**:
    1. Saved stats to kmeans\_cluster\_stats.csv.
* **DBSCAN**:
  + **Description**: Density-based clustering, marks outliers as noise.
  + **Why Chosen**: Captures irregular cluster shapes.
  + **Development Steps**:
    1. **Data Preparation**:
       - Same features as K-means.
       - Scaled features using MinMaxScaler.
    2. **Hyperparameter Tuning**:
       - Tested eps values: 0.1 to 0.5.
       - min\_samples: 5.
    3. **Training**:
       - Applied DBSCAN to scaled data.
    4. **Metrics**:
       - Silhouette Score: 0.58.
    5. **Labeling**:
       - Same as K-means, with "Noise" for outliers (cluster\_id = -1).
  + **Saved Output**:
    1. Saved stats to dbscan\_cluster\_stats.csv.

**Challenges:**

* **DBSCAN Noise**: Initial runs marked many zones as noise; added hour feature and tuned eps.
* **Feature Scaling**: Ensured features were scaled for distance-based clustering.

**5.3.6 Model Comparison (compare\_models.ipynb)**

**Objective:**

Compare all models to select the best for the dashboard.

**Steps:**

1. **Load Results**:
   * Prophet: Loaded forecasts, computed MAE, RMSE, MAPE, Median APE.
   * Random Forest/XGBoost: Loaded classification metrics, adjusted thresholds to improve F1-scores.
     + Random Forest: F1-Score = 2.26% (adjusted).
     + XGBoost: F1-Score = 6.02% (adjusted).
   * K-means/DBSCAN: Loaded silhouette scores.
2. **Visualization**:
   * Bar charts for MAPE, F1-scores, and silhouette scores.
   * Saved as model\_comparison.png.
3. **Saved Output**:
   * Saved comparison table to model\_comparison.csv.

**Conclusion:**

* **Forecasting**: Prophet (MAPE = 27.22%).
* **Classification**: XGBoost (F1-Score = 6.02% after adjustment).
* **Clustering**: K-means (Silhouette Score = 0.62).

**5.4 Dashboard Development (dashboard.py)**

**5.4.1 Overview**

The dashboard is a Streamlit-based web application with two pages:

* **Driver Dashboard**: Helps drivers find high-demand zones, view forecasts, and get recommendations.
* **Technical Dashboard**: Provides model metrics, visualizations, and impact analysis for technical users.

**5.4.2 Architecture**

* **Pages**:
  + **Driver Dashboard**:
    - **Tabs**:
      * **Demand Insights**: Shows demand forecasts and high-demand zones.
      * **Recommendations**: Provides repositioning recommendations with alternatives.
      * **Visualizations**: Displays hotspot maps and demand distributions.
  + **Technical Dashboard**:
    - Sections: City-wide forecast, zone-specific forecast, clustering visualization, model comparison, impact simulation.
* **Sidebar Inputs**:
  + Navigation: Select between Driver and Technical Dashboard.
  + Inputs (Driver Dashboard):
    - Day of the Week: Dropdown (e.g., Monday).
    - Is Holiday: Checkbox (default based on current date).
    - Current Hour: Slider (0-23, default to current hour).
    - Weather Condition: Dropdown (Clear, Rainy, Snowy).
    - Current Zone: Dropdown of zone IDs.
    - Classifier Choice: Dropdown (XGBoost or Random Forest).
  + Inputs (Technical Dashboard):
    - Forecast Date Range: Date picker.
    - Zone Selection: Dropdown (City-Wide or specific zone).
* **Styling**:
  + Custom CSS for buttons, cards, metrics, and charts (e.g., yellow buttons, dark-themed cards).

**5.4.3 Driver Dashboard Features and Logic**

The Driver Dashboard has three tabs, each with specific features and underlying logic.

**Tab 1: Demand Insights**

**Feature 1: Demand Forecast for the Next Hour**

* **Purpose**: Show demand trends for the next 6 hours for top zones.
* **Logic**:
  1. **Prepare Future Dates**:
     + Created a DataFrame with timestamps for the next 6 hours.
     + Added weather conditions based on user input (precipitation: 0 for Clear, 0.5 for Rainy, 1.0 for Snowy).
     + Set temperature to 15°C (constant for simplicity).
     + Computed temp\_precip.
  2. **Forecast Demand**:
     + Used zone-specific Prophet models for the top 10 zones.
     + Predicted demand (yhat) for each hour.
     + Also forecasted demand assuming clear weather for comparison.
  3. **Compute Demand-to-Supply Ratio**:
     + Estimated future driver supply using historical averages per hour.
     + Computed ratio = forecasted\_demand ÷ forecasted\_driver\_count.
     + Capped ratio at 10 for visualization.
  4. **Visualization**:
     + Stacked area chart showing ratio trends for the top 5 zones by demand.
     + X-axis: Time (next 6 hours).
     + Y-axis: Demand-to-Supply Ratio.
  5. **Table Display**:
     + Showed PULocationID, forecasted\_demand, and demand\_slope for all zones.
* **Output**:
  1. Chart and table showing demand trends and forecasts.

**Feature 2: High-Demand Zones Right Now**

* **Purpose**: Identify and display the top 5 high-demand zones based on current conditions.
* **Logic**:
  1. **Filter Data**:
     + Filtered pickup counts and features by user inputs (day\_of\_week, hour, is\_holiday).
     + Merged data, filling missing values (e.g., temperature = 0, avg\_speed = mean).
  2. **Predict High-Demand Zones**:
     + Used XGBoost or Random Forest (user-selected) to predict probabilities.
     + Features: precipitation, is\_holiday, avg\_speed, day\_of\_week, temp\_precip.
     + Threshold: 50th percentile of probabilities (minimum 0.001).
     + Labeled zones as high-demand if probability ≥ threshold.
  3. **Fallback (Hybrid Approach)**:
     + If no zones met the threshold, combined pickup\_count and probability (combined\_score).
     + Selected top 5 zones by combined\_score.
  4. **Enhance Data**:
     + Merged with current supply data (driver\_count).
     + Computed current\_demand\_supply\_ratio.
     + Merged with Prophet forecasts (forecasted\_demand, demand\_slope, prophet\_confidence).
     + Added zone metadata (Pickup\_Zone, coordinates).
     + Computed ensemble\_score = 0.5 × current\_demand\_supply\_ratio + 0.5 × (forecasted\_demand ÷ driver\_count).
     + Computed ensemble\_confidence = 0.5 × xgb\_confidence + 0.5 × prophet\_confidence, adjusted by historical accuracy.
  5. **Compute Earnings**:
     + Calculated distance and travel\_time from the driver’s current location.
     + Computed potential\_trips, trips\_per\_hour, gross\_earnings, net\_earnings, fuel\_cost, congestion\_factor.
     + Predicted adjusted\_earnings using Random Forest Regressor.
     + Determined risk\_score: "Low" if demand\_slope ≥ 0, "High" otherwise.
  6. **Visualization**:
     + **Stats Card**: Showed total forecasted demand and number of high-demand zones.
     + **Pie Chart**: Demand-to-Supply Ratio distribution across top 5 zones.
     + **Table**: Displayed PULocationID, Pickup\_Zone, pickup\_count, driver\_count, current\_demand\_supply\_ratio, forecasted\_demand, adjusted\_earnings, travel\_time, risk\_score.
* **Output**:
  1. Stats card, pie chart, and table showing high-demand zones.

**Tab 2: Recommendations**

**Feature 1: Primary Recommendation**

* **Purpose**: Recommend the best zone for the driver to reposition to.
* **Logic**:
  1. **Select Recommended Zone**:
     + Prioritized zones with high preference\_score (tracked via session state) and low distance.
     + If no preferred zones, selected zones with travel\_time < 30 minutes.
     + Chose the zone with the highest adjusted\_earnings.
  2. **Compute Metrics**:
     + Extracted zone\_id, zone\_name, travel\_time, adjusted\_earnings, risk\_score, current\_demand\_supply\_ratio, forecasted\_demand, potential\_trips, avg\_fare, congestion\_factor.
  3. **Display**:
     + **Card**: Showed zone details, travel time, earnings, passengers, fare, and risk.
     + **Demand Surge Alert**: Warned if demand\_slope > 50.
     + **Feedback Buttons**:
       - "Accept": Increased preference\_score by 1, logged earnings, updated recommendation timestamp.
       - "Reject": Decreased preference\_score by 1.
  4. **Details Expander**:
     + **Gauge Charts**:
       - Demand-to-Supply Ratio: Scaled to 0-100, green for high values.
       - Congestion Level: Scaled to 0-100, green for low values.
       - Risk Level: 0 for "Low", 100 for "High".
     + **Bar Chart**: Top 3 contributing features (e.g., precipitation) with explanations.
     + **Historical Demand Trend**: Line chart of average pickups by hour for the zone.
     + **Text Metrics**: Showed ratio and forecasted demand.

**Feature 2: Alternative Recommendation**

* **Purpose**: Suggest an alternative zone if the primary recommendation has high risk or is rejected.
* **Logic**:
  1. **Select Alternative**:
     + Filtered zones with travel\_time < 30, risk\_score = "Low", and different from the primary zone.
     + Chose the zone with the highest adjusted\_earnings.
  2. **Display**:
     + **Card**: Showed zone details, travel time, earnings, passengers, fare, and risk.
     + **Details Expander**: Showed ratio and forecasted demand.

**Feature 3: Demand Trend for Recommended Zone**

* **Purpose**: Show the demand forecast for the recommended zone.
* **Logic**:
  1. **Forecast**:
     + Used the zone-specific Prophet model to predict demand for the next 6 hours.
     + Also forecasted demand assuming clear weather.
  2. **Compute Weather Impact**:
     + Difference between actual and clear-weather forecasts.
     + Explained impact (e.g., "Rain increases demand").
  3. **Visualization**:
     + Line chart with actual forecast (solid line) and clear-weather forecast (dashed line).
     + X-axis: Time.
     + Y-axis: Pickups.

**Feature 4: Earnings Comparison**

* **Purpose**: Compare earnings if the driver stays vs. moves to the recommended zone.
* **Logic**:
  1. **Compute Earnings**:
     + stay\_earnings: Earnings in the current zone (from high\_demand\_zones or 0 if not high-demand).
     + move\_earnings: Earnings in the recommended zone.
     + earnings\_diff = move\_earnings - stay\_earnings.
  2. **Visualization**:
     + Horizontal bar chart comparing stay\_earnings and move\_earnings.
     + Title: Earnings gain in dollars per hour.

**Feature 5: Personalized Tips to Maximize Earnings**

* **Purpose**: Provide tailored advice based on data and preferences.
* **Logic**:
  1. **Peak Hours**:
     + Identified top 3 hours with yhat > 80th percentile from city-wide Prophet forecasts.
  2. **Top High-Demand Zones**:
     + Listed top 3 zones by adjusted\_earnings from high\_demand\_zones.
  3. **Nearby High-Demand Zones**:
     + Found zones with travel\_time < 15 minutes, sorted by adjusted\_earnings.
  4. **Preferred Zones**:
     + Identified zones with preference\_score > 0, sorted by adjusted\_earnings.
  5. **Nearby Hotspots**:
     + Found "High-Demand Urban" zones within 10 miles, sorted by forecasted\_demand.
  6. **Top Preferred Zones**:
     + Displayed top 3 zones by preference\_score, including distances.
  7. **Recommendation Performance**:
     + Computed average earnings from accepted recommendations (stored in session state).
  8. **Tips**:
     + Combined insights into actionable advice (e.g., "Visit Zone X nearby").

**Tab 3: Visualizations**

**Feature 1: Hotspot Map**

* **Purpose**: Show high-demand zones and clusters on a map.
* **Logic**:
  1. **Prepare Data**:
     + Used zone\_features with forecasted\_demand and kmeans\_cluster.
     + Identified high-demand zones from high\_demand\_zones.
  2. **Map Creation**:
     + Used Folium to create a map centered at NYC (40.7128, -74.0060).
     + Added CircleMarkers for each zone:
       - Color: Red for high-demand, gray for low-demand, or blue/green/purple/orange based on cluster.
       - Radius: Scaled by forecasted\_demand (5 + forecasted\_demand ÷ max\_demand × 10).
       - Popup: Zone ID, label, and forecasted demand.
  3. **Display**:
     + Rendered map with folium\_static.

**Feature 2: Legend and Stats**

* **Purpose**: Explain map colors and show overall stats.
* **Logic**:
  1. **Legend**:
     + Red: High-demand zones.
     + Gray: Low-demand zones.
     + Blue/Green/Purple/Orange: K-means clusters.
  2. **Stats**:
     + Number of high-demand zones.
     + Total forecasted demand across all zones.

**Feature 3: Demand Distribution Chart**

* **Purpose**: Show demand distribution by zone type.
* **Logic**:
  1. **Compute Distribution**:
     + Grouped zone\_features by label, summed forecasted\_demand.
  2. **Visualization**:
     + Pie chart showing demand percentage by zone type (e.g., "High-Demand Urban").

**5.4.4 Technical Dashboard Features and Logic**

**Feature 1: City-Wide Demand Forecast**

* **Purpose**: Show city-wide actual vs. predicted demand.
* **Logic**:
  1. **Filter Data**:
     + Filtered prophet\_df by user-selected date range.
  2. **Visualization**:
     + Line chart with actual (y) and predicted (yhat) pickups.
     + Metrics displayed: MAE (644.59), MAPE (27.22%), Median APE (15.28%).

**Feature 2: Zone-Specific Demand Forecast**

* **Purpose**: Show demand forecast for a selected zone.
* **Logic**:
  1. **Select Zone**:
     + User selects a zone or "City-Wide".
  2. **Prepare Data**:
     + For a specific zone, loaded historical pickups and predicted using the zone’s Prophet model.
     + Filtered by date range.
  3. **Visualization**:
     + Line chart with actual and predicted pickups.
     + If "City-Wide" selected, showed the city-wide forecast.

**Feature 3: Demand Hotspot Clusters**

* **Purpose**: Visualize K-means clusters on a map.
* **Logic**:
  1. **Prepare Data**:
     + Used zone\_features with kmeans\_cluster and label.
  2. **Map Creation**:
     + Added CircleMarkers for each zone:
       - Color: Gray for low-demand (cluster\_id = -1), red/blue/green/purple for clusters.
       - Popup: Zone ID and label.
  3. **Display**:
     + Rendered map with folium\_static.
     + Showed Silhouette Score (0.86, though actual was 0.62 in notebook).

**Feature 4: Model Comparison Summary**

* **Purpose**: Display performance metrics for all models.
* **Logic**:
  + Loaded model\_comparison.csv.
  + Displayed as a table (e.g., Prophet MAPE, XGBoost F1-score, K-means Silhouette Score).

**Feature 5: Impact Simulation**

* **Purpose**: Simulate the impact of AI-guided repositioning.
* **Logic**:
  1. **Prepare Data**:
     + Used holiday data (is\_holiday = 1) for high-demand scenarios.
     + Merged with current supply data.
  2. **Compute Metrics**:
     + demand\_supply\_ratio = pickup\_count ÷ driver\_count.
     + Baseline Earnings: Total pickups × average fare ÷ average drivers.
     + AI-Guided Earnings: Top 3 zones’ pickups × average fare ÷ 3.
     + Wait Time Reduction: (demand\_supply\_ratio - 1) × 5 minutes (assumed).
  3. **Display**:
     + Showed baseline vs. AI-guided earnings and estimated wait time reduction.

**5.4.5 Additional Dashboard Features**

* **Caching**: Used @st.cache\_data and @st.cache\_resource to cache data loading and model loading.
* **Session State**:
  + Tracked driver\_preferences (dictionary of zone preferences).
  + Stored performance\_history (list of earnings from accepted recommendations).
  + Logged last\_recommendation\_time to prevent frequent repositioning.
* **Dynamic Updates**: Visualizations and recommendations update based on user inputs.

**6. Tools and Technologies**

* **Programming Language**: Python 3.12.0.
* **Libraries**:
  + **Data Processing**: Pandas, NumPy.
  + **ML Models**: Scikit-learn (Random Forest, K-means, DBSCAN), XGBoost, Prophet.
  + **Visualization**: Matplotlib, Seaborn, Folium, Plotly.
  + **Dashboard**: Streamlit.
  + **Geospatial**: Geopy.
  + **Time Handling**: Pytz, datetime.
* **Environment**: Jupyter Notebook for development, Streamlit for deployment.

**7. Results and Analysis**

**7.1 Demand Forecasting**

* **Model**: Prophet.
* **Metrics**:
  + MAE: 644.59 pickups.
  + RMSE: 934.30 pickups.
  + MAPE: 27.22%.
  + Median APE: 15.28%.
  + High APE Hours: 7 out of 288.
* **Analysis**:
  + MAPE of 27.22% indicates moderate accuracy, but high-error hours suggest difficulty with sudden demand spikes.
  + Suitable for providing general trends, as visualized in the dashboard.

**7.2 High-Demand Classification**

* **Models**:
  + Random Forest: F1-Score = 2.26% (adjusted threshold).
  + XGBoost: F1-Score = 6.02% (adjusted threshold).
* **Analysis**:
  + XGBoost outperformed Random Forest, chosen for the dashboard.
  + Adjusted threshold improved F1-score but reduced recall, balancing precision and coverage.

**7.3 Earnings Prediction**

* **Model**: Random Forest Regressor.
* **Metrics**:
  + MSE: 14.48.
  + R²: 94.03%.
* **Analysis**:
  + High R² indicates excellent predictive accuracy.
  + travel\_time (85.10%) being the most important feature shows distance significantly impacts earnings.

**7.4 Clustering**

* **Models**:
  + K-means: Silhouette Score = 0.62.
  + DBSCAN: Silhouette Score = 0.58.
* **Analysis**:
  + K-means provided better clustering (higher silhouette score).
  + Labels like "High-Demand Urban" help drivers understand zone characteristics.

**7.5 Dashboard Impact**

* **Driver Benefits**:
  + Recommendations guide drivers to high-earning zones, increasing earnings.
  + Forecasts and hotspot maps help plan shifts.
* **Wait Time Reduction**:
  + Simulation estimated a reduction in wait times, though exact impact wasn’t measured (proposal goal: 15%).

**8. Challenges and Solutions**

* **Data Leakage**:
  + **Challenge**: Using pickup\_count in classification would leak the target.
  + **Solution**: Excluded pickup\_count, used RFE-selected features.
* **DBSCAN Noise**:
  + **Challenge**: Too many zones marked as noise.
  + **Solution**: Added hour feature, tuned eps.
* **Forecasting Accuracy**:
  + **Challenge**: High MAPE (27.22%).
  + **Solution**: Kept Prophet, noted as a future improvement area.
* **Traffic Data Absence**:
  + **Challenge**: No Uber Movement Data.
  + **Solution**: Used avg\_speed as a proxy.
* **Driver Preferences**:
  + **Challenge**: No preference data in training.
  + **Solution**: Used placeholder, implemented preference tracking in the dashboard.

**9. Usage Instructions**

1. **Setup**:
   * Install Python 3.12.0.
   * Install libraries: pip install pandas numpy matplotlib seaborn sklearn xgboost prophet folium streamlit geopy plotly pytz.
2. **Run the Dashboard**:
   * Navigate to /Users/saleh/Desktop/Projects/AIProject/TaxiOptimization/.
   * Run: streamlit run dashboard.py.
3. **Interact**:
   * **Driver Dashboard**:
     + Select day, hour, weather, and current zone.
     + View forecasts, recommendations, and maps.
   * **Technical Dashboard**:
     + Select date range and zone.
     + Review model metrics and visualizations.

**10. Future Work**

* **Improve Forecasting**: Use LSTM (as per proposal’s referenced paper) to reduce MAPE.
* **Traffic Data**: Integrate Uber Movement Data or real-time APIs.
* **Driver Preferences**: Collect real preference data.
* **Advanced Clustering**: Test hierarchical clustering.
* **Real-Time Updates**: Enable live data feeds.
* **Wait Time Measurement**: Quantify wait time reduction (target: 15%).

**11. Conclusion**

The NYC Taxi Optimization Dashboard successfully addresses driver distribution issues in NYC by predicting demand, identifying high-demand zones, clustering zones, estimating earnings, and providing actionable recommendations. The project expanded beyond the initial proposal by implementing advanced ML models, thorough preprocessing, and a feature-rich dashboard. The Driver Dashboard offers detailed insights through forecasts, maps, and personalized recommendations, while the Technical Dashboard provides model evaluation tools. Despite challenges like forecasting accuracy, the system helps drivers increase earnings and likely reduces wait times.

**12. References**

1. NYC TLC Data Portal.
2. Scikit-learn Documentation.
3. Prophet Documentation.
4. "Deep Learning for Taxi Demand Prediction" (IEEE, 2021).
5. "Dynamic Vehicle Repositioning via Reinforcement Learning" (KDD, 2022).

**13. Appendices**

**13.1 File Paths**

* **Data**: /Users/saleh/Desktop/Projects/AIProject/TaxiOptimization/data/.
* **Models**: /Users/saleh/Desktop/Projects/AIProject/TaxiOptimization/models/.
* **Visualizations**: /Users/saleh/Desktop/Projects/AIProject/TaxiOptimization/visualizations/.

**13.2 Model Metrics**

* Prophet: MAPE = 27.22%, MAE = 644.59.
* XGBoost: F1-Score = 6.02% (adjusted).
* Random Forest (Earnings): R² = 94.03%.
* K-means: Silhouette Score = 0.62.