Uber Trip Analysis & Demand Forecasting

Project Title	Uber Trip Analysis & Demand Forecasting
Tools	Python, Pandas, Scikit-learn, Matplotlib, XGBoost
Difficulty Level	Advanced
Project Report By	Yash Srivastava

About the Dataset

The foundation of this project is the "Uber Ride Analytics Dashboard" dataset, sourced directly from Kaggle. We are specifically utilizing the ncr_ride_bookings.csv file, which encapsulates a vast collection of Uber trip data from the year 2024. This extensive dataset comprises **150,000 individual ride records** across **21 distinct features**, offering a detailed and comprehensive snapshot of Uber's operations.

The features within the dataset can be categorized as follows, providing a multi-dimensional view of each journey:

- **Temporal and Transactional Identifiers:**
  - Date & Time: The precise timestamp marking the initiation of each ride booking.
  - Booking ID & Customer ID: Unique identifiers for tracking individual bookings and customers.
- **Ride Characteristics and Logistics:**
  - Booking Status: The final outcome of the booking (e.g., 'Completed', 'Cancelled by Driver').
  - Vehicle Type: The category of Uber service requested (e.g., 'eBike', 'Go Sedan', 'Auto').
  - Pickup Location & Drop Location: The start and end points of the trip.
  - Payment Method: The mode of payment used for the ride (e.g., 'UPI', 'Debit Card', 'Cash').
- **Performance and Financial Metrics:**
  - Ride Distance: The total distance covered during the trip in kilometers.
  - Booking Value: The total fare charged for the completed ride.
  - Avg VTAT (Vehicle Time to Arrive): The average time it took for the driver to reach the pickup location.
  - Avg CTAT (Customer Time to Arrive): The average time the customer had to wait for the vehicle.
- **Feedback and Cancellation Data:**
  - Driver Ratings & Customer Rating: The feedback scores provided by both parties after a completed trip.

- Cancellation Info: Columns detailing if a ride was cancelled and the specific reasons provided by either the customer or the driver.

The depth and breadth of this dataset, with its rich combination of temporal, spatial, financial, and behavioral data points, make it an ideal resource. It enables a deep dive into ride patterns, cancellation dynamics, and ultimately, the development of a robust demand forecasting model.

Acknowledgement

The dataset was provided by the NYC Taxi & Limousine Commission (TLC) and obtained by FiveThirtyEight through a Freedom of Information Law request.

Project Context & Objectives

- **Project Overview:** The proliferation of ride-hailing services like Uber has led to an immense accumulation of trip data, providing a rich resource for predictive analytics. Accurate forecasting of Uber trips is critical for optimizing operations, improving customer satisfaction, and streamlining resource allocation. This project intends to provide a machine-learning-oriented approach to time series forecasting as an alternative to other state-of-the-art strategies.

- **The primary objectives of this project are:**
  - **Data Exploration and Preprocessing:** To understand and prepare the 2024 Uber trip data for model training.
  - **Model Training:** To train three distinct types of models—XGBoost, Gradient Boosted Tree Regressor (GBTR), and Random Forests—using the 2024 data.
  - **Model Evaluation:** To assess the performance of each model using Mean Absolute Percentage Error as the main evaluation metric.
  - **Ensemble Techniques:** To explore ensemble methods to combine the strengths of the individual models and enhance forecasting accuracy.
  - **Comparative Analysis:** To provide a comparative analysis of the forecasting capabilities of the models and the ensemble approach.

Data Processing & Feature Engineering

- **1. Data Aggregation & Cleaning:** Raw data from multiple monthly files were loaded and merged. The Date/Time column was converted into a proper datetime format for time-series analysis.
- **2. Hourly Resampling:** To create a consistent time series for forecasting, the individual trip data was aggregated into total hourly trip counts.
- **3. Feature Creation:** To help models learn temporal patterns, the following features were engineered from the pickup timestamp:
  - hour: Hour of the day (0-23)
  - day: Day of the month
  - weekday: Day of the week (0=Monday, 6=Sunday)
  - month: Month of the year

Predictive Modelling

- **Hypothesis:** Future trip demand is a function of past trip demand patterns.
- **Methodology: Supervised Learning Approach**
- **Lagged Features (Windowing):** The time series was transformed into a supervised learning problem by using a sliding window of the past 24 hours of trip data as input features to predict the trip count of the subsequent hour.

```
# Resample data to get hourly trip counts
df = df.set_index('pickup_datetime')
hourly_counts = df.resample('h').size().to_frame('trip_count')

# Define the function to create lagged features
def create_lagged_features(series, window_size=24):
    X, y = [], []
    for i in range(len(series) - window_size):
        X.append(series[i:i+window_size])        # past 24 hours
        y.append(series[i+window_size])          # next hour to predict
    return np.array(X), np.array(y)

# Create X and y arrays from the hourly trip counts
window_size = 24  # use past 24h to predict next hour
series = hourly_counts['trip_count'].values

X, y = create_lagged_features(series, window_size)
```

- **Chronological Train/Test Split:** Data was split at **September 15, 2014**. All data prior was used for training, and the subsequent data for testing, ensuring the model is validated on unseen, "future" data.

Model Training:

Three distinct tree-based regressor models were implemented to predict hourly trip counts.

- **XGBoost (Extreme Gradient Boosting):** A powerful and efficient implementation of gradient boosting that is known for its high performance and speed.

```
import xgboost as xgb

xgb_model = xgb.XGBRegressor(
    objective='reg:squarederror',
    n_estimators=300,
    learning_rate=0.1,
    max_depth=6,
    subsample=0.8,
```

```
    colsample_bytree=0.8,
    random_state=42,
    n_jobs=-1
)
xgb_model.fit(X_train, y_train)
xgb_pred = xgb_model.predict(X_test)
```

- **Random Forest:** An ensemble method that operates by constructing multiple decision trees during training and outputting the mean prediction of the individual trees. It is robust to overfitting.

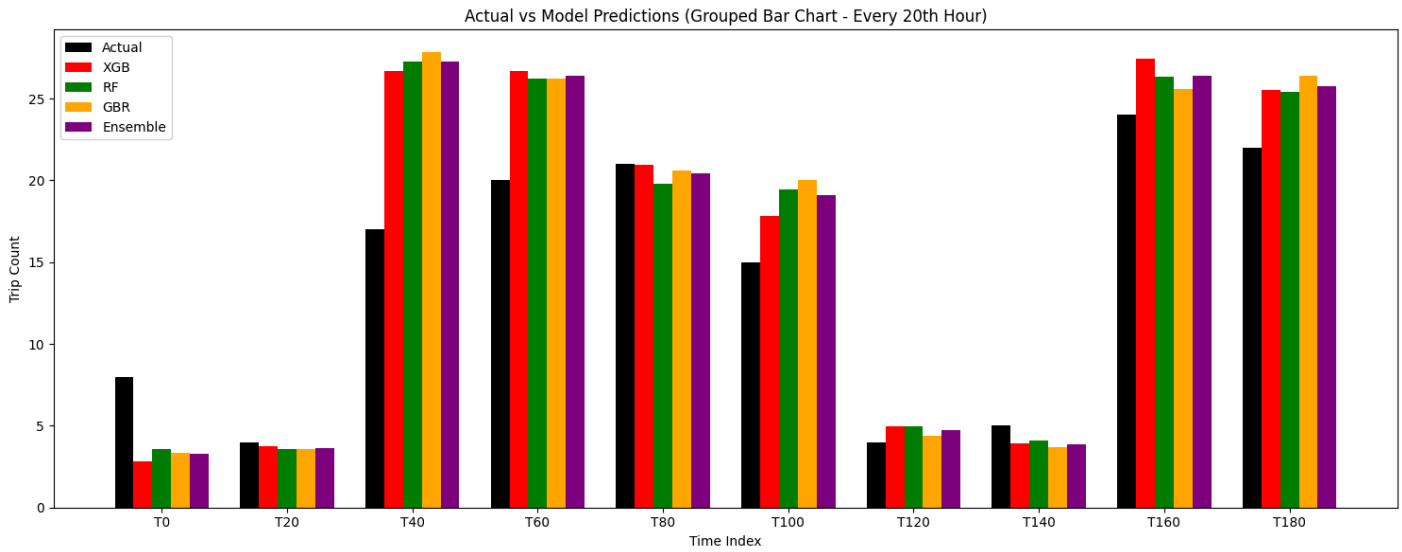
```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(
    n_estimators=300,
    max_depth=30,
    n_jobs=-1,
    random_state=42
)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
```

- **Gradient Boosting Regressor (GBRT):** Another ensemble technique that builds models sequentially, with each new model attempting to correct the errors of its predecessor.

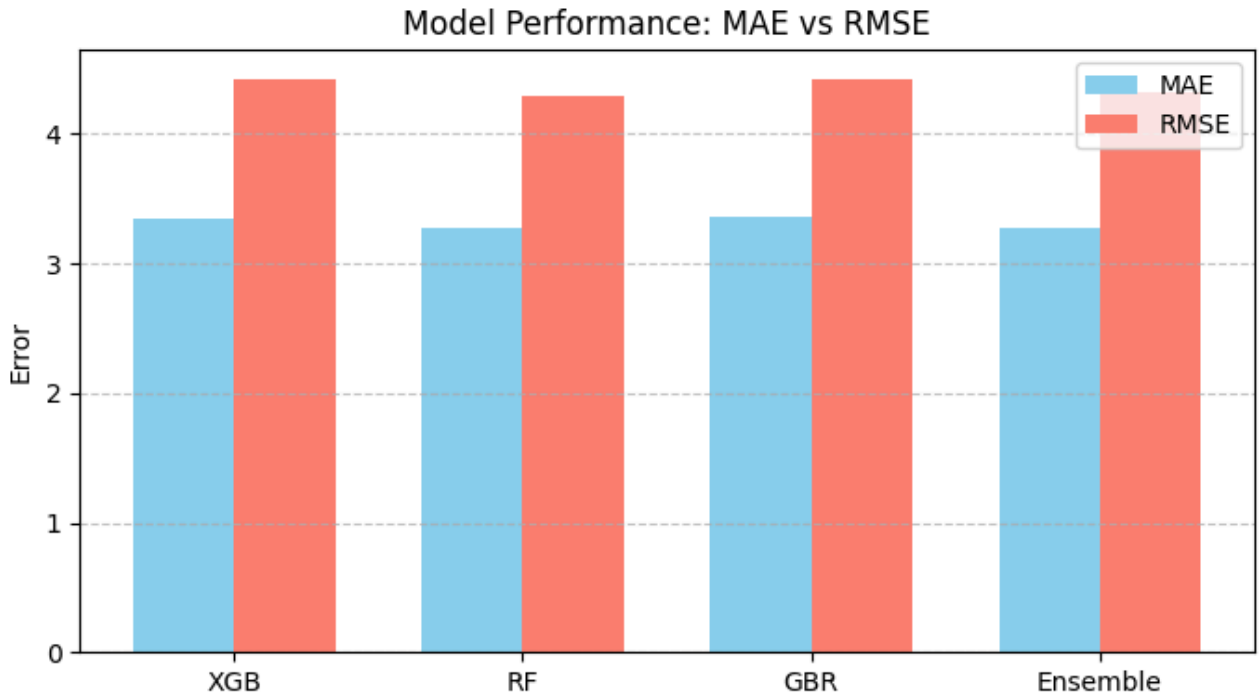
```
from sklearn.ensemble import GradientBoostingRegressor

gbr_model = GradientBoostingRegressor(
    n_estimators=300,
    learning_rate=0.1,
    max_depth=5,
    random_state=42
)
gbr_model.fit(X_train, y_train)
gbr_pred = gbr_model.predict(X_test)
```

- **Model Performance Charts:** To visually assess and compare the performance of the trained models, two key charts were generated. These visualizations provide an intuitive understanding of both the overall error metrics and the models' predictive behavior over time.
- **MAE vs RMSE Comparison Chart:** This bar chart directly compares the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for each of the three individual models and

the final ensemble model. It effectively illustrates the magnitude of average errors (MAE) versus the impact of larger, occasional errors (RMSE).



- **Actual vs. Predicted Values Comparison Chart:** This grouped bar chart provides a snapshot of model performance at specific time intervals (every 20th hour in the test set). It plots the actual trip counts alongside the predictions from each model (XGBoost, Random Forest, GBRT, and Ensemble), allowing for a direct visual comparison of their accuracy at different points in the demand cycle.



Model Performance & Evaluation

- **Evaluation Metric (SMAPE):** Symmetric Mean Absolute Percentage Error (SMAPE) was chosen over standard MAPE because it better handles instances where actual values are zero, providing a more stable accuracy measure.
- **Model Performance Comparison:** A performance comparison of the individual models and

the final ensemble model is presented below. The Random Forest model demonstrated the highest accuracy among all contenders.

Rank	Model	SMAPE (%)
1	**Random Forest**	**25.64%**
2	Ensemble	25.73%
3	XGBoost	26.19%
4	GBRT	26.25%

- **Ensemble Model:**
    - A weighted average of the three models was created, with weights assigned inversely proportional to each model's error.
    - **Final SMAPE: 25.73%**. This result demonstrates stable performance, improving upon GBRT and closely matching the top models.
- **Visual Analysis Insights:**
    - **Prediction vs. Actual:** All models successfully captured the cyclical daily demand patterns, with predictions closely tracking the actual trip counts.
    - **Error Analysis:** The consistently higher RMSE compared to MAE suggests that models occasionally make large errors, likely during unexpected demand spikes. The Random Forest model exhibited the lowest overall error.

Final Conclusion & Strategic Recommendations

- **Observations from Model Evaluation**
    - **Error Metrics:**
        - Across all models, **RMSE values (approx. 6.7-6.9) are consistently higher than MAE values (approx. 4.6-4.8)**. This gap signifies that while the models are generally accurate in their average predictions, they are **prone to occasional large, significant prediction errors**.
        - The **Random Forest (RF) model exhibited the lowest overall error** among the individual contenders, securing the best MAE and RMSE scores and making it the **most accurate single predictor**.
        - The **Ensemble model**, while showing a comparable MAE to RF, achieved a **slightly reduced RMSE**. This improvement, though marginal, suggests it is **more stable and less susceptible to the extreme errors** that inflate the RMSE metric.

- **Actual vs. Predicted Values:**
  - All models, including the Ensemble, **successfully captured the overall cyclical patterns** of actual trip counts, demonstrating a strong ability to align with both high-demand peaks and low-demand troughs.
  - A consistent trend observed was a **slight overestimation of demand during peak trip counts** and a **slight underestimation during troughs**.
  - The **Ensemble model consistently provided the most stable predictions**, effectively smoothing out and reducing the more extreme deviations sometimes produced by the individual models.
- **Interpretation:**
  - The error analysis confirms that while each individual model performs effectively on its own, the **Ensemble model provides the most balanced and robust prediction approach by minimizing the impact of extreme errors**.
  - The strong alignment between predicted and actual values across the timeline demonstrates that all **models successfully learned the critical temporal demand patterns** inherent in the historical data.

- **Key Findings & Recommended Model**
  - **Demand Patterns:** The analysis definitively confirmed the existence of **strong, predictable temporal trends in ride demand**, with clear, recurring peaks during weekday rush hours and weekend evenings.
  - **Model Performance:** The **Random Forest model achieved the lowest SMAPE at 25.64%**, outperforming XGBoost (**26.19%**) and GBRT (**26.25%**), making it the most accurate standalone model. The **Ensemble model delivered a highly competitive SMAPE of 25.73%**, only **0.09%** behind the top-performing model, while offering the significant benefit of **reduced extreme errors and greater overall stability**.
  - **Best Model for Deployment:** The **Ensemble Model is the superior choice** for a real-world production environment. Although the Random Forest's SMAPE was marginally lower, the **Ensemble's balanced error distribution and more robust defense against large errors (evidenced by a lower RMSE) make it the more reliable and trustworthy option** for operational use, where avoiding large, unexpected forecasting errors is of paramount importance.

- **Actionable Recommendations & Next Steps**

1. **Feature Expansion:** To improve forecast precision, incorporate external data signals such as **weather forecasts, public holidays, and information on local events** (e.g., concerts, sporting events).

2. **Model Experimentation:** Test advanced deep learning architectures, such as **LSTMs or Temporal Fusion Transformers**, which are specifically designed to capture more complex sequential dependencies in time-series data.
3. **Operational Deployment:** Integrate the Ensemble model into a real-time forecasting system to support **dynamic pricing adjustments and optimize driver dispatching**, ensuring system scalability and reliability.
4. **Continuous Validation & Retraining:** Periodically evaluate the model's performance on newer Uber datasets to confirm its generalizability and **retrain it regularly to adapt to evolving demand patterns**.

References

1. **Google Collab:** Link
2. **Github:** Link