MSML 605 Computing Systems for Machine Learning

Taxi Demand Predictor

Group Project Proposal

March 2, 2025
Professor: Dr. Samet Ayhan

Group Members:

1. Mariia Vetluzhskikh
2. Thilak Mohan
3. Mengfan Zhang
4. Dennis Reyes

Problem Statement:

We aim to develop an ML-based nowcasting system to predict real-time demand for taxi services in New York City based on real-time weather and traffic data. The system should be deployed in a cloud environment using containers, ensuring scalability, availability, and reliability. It should provide predictions via a REST API, which can be accessed by other services/applications to optimize resources and ensure efficiency.

High-Level Approach, Relevance, and Novelty:

1. **Prediction Objective: Real-time Demand Prediction**

The core task is to predict the number of taxi rides requested in different parts of New York City for the next 30-60 minutes based on real-time weather and traffic data. This prediction will help optimize taxi dispatch systems by better allocating taxis to areas with high demand.

2. **Input Data: Historical and Real-Time Taxi Data**

We will use data related to taxi ride requests: pickup locations, timestamps, ride durations. It will serve as the historical basis for demand patterns. Whereas weather and traffic data (e.g., temperature, rain, traffic congestion levels) will be sourced via APIs like OpenWeather and Google Maps and integrated in real-time for the nowcasting predictions.

3. **Time-Series and Predictive Modeling:**

We will use machine learning models to predict future taxi demand based on time of day, weather conditions, and traffic data. Model performance will be evaluated using standard metrics such as MAE or RMSE.

4. **Cloud Deployment: Containerization and Scalability**

The model and API will be containerized using Docker, so that the system can be easily deployed and scaled in a cloud environment (AWS)

5. **Cloud Deployment: API**

We will develop a RESTful API that exposes the trained model for real-time predictions, which can be queried by any client to get updated demand predictions.

6. **Monitoring and Real-Time Updates:**

Then we will set up data pipelines to stream live weather and traffic data into the system for real-time predictions. To track the health of the system, its performance, and API usage, we will implement logging and monitoring tools.

Planned Contribution:

1. Data Collection & Preprocessing (Mengfan Zhang)

- Data Cleaning: Handle missing values, outliers, and data inconsistencies.
- Feature Engineering: Design and create features like time-based (hour of day, day of week, holiday flag), weather data, past taxi demand, location-based features, etc.
- Data Preprocessing: Apply transformations (normalization, scaling), split data for training/validation, and create time-series windows for modeling

2. Model Development & Training (Thilak Mohan)

- Model Selection: Choose the model(s) to use for prediction (e.g., Random Forest, XGBoost, or LSTM for time-series).
- Model Training: Train models using the preprocessed data. Experiment with hyperparameters to optimize model performance.
- Model Evaluation: Choose the appropriate evaluation metrics (MAE, RMSE, etc.), and evaluate the model's performance on the validation set.
- Model Testing: After training, test the model with out-of-sample data to make sure it generalizes well

3. Cloud & API Deployment (Mariia Vetluzhskikh)

- API Development: Build a RESTful API using Flask or FastAPI. This API will accept features (like time of day, weather, etc) and return a demand prediction.
- Containerization: Use Docker to containerize the API and machine learning model for easy deployment and portability.
- Cloud Deployment: Deploy the Docker container to a cloud provider (AWS). Set up Kubernetes for orchestration, ensuring the system scales based on incoming traffic.
- API Endpoints: Make sure the API endpoints are well-documented and can handle multiple incoming requests (for scalability)

4. Real-Time Data Integration & Monitoring (Dennis Reyes)

- Real-Time Data Handling: Set up integration with live data sources (e.g., weather, traffic data) using APIs like OpenWeather or Google Maps API. This will involve setting up a system that pulls live data and uses it as input for the model.
- Real-Time Data Streaming: If you want to go beyond batch predictions, you could use tools like Apache Kafka or Google Pub/Sub for streaming data into the prediction system.
- Monitoring & Metrics: Implement monitoring tools (e.g., Prometheus and Grafana) to track: API performance (latency, uptime), Model performance (error rates, prediction accuracy), System resource usage (CPU, memory)
- Logging: Set up proper logging and alerting systems to monitor the health of the deployed application in real-time.

Planned Tools:

Data Sources: NYC Yellow Taxi Trip Data, Taxi & Limousine Commission Data, yellow taxi trip records, Global Weather Data, Historical World Weather Data API, Maps JavaScript API, github rep (should help with feature engineering)
Python Libraries: requests, pandas, numpy, scikit-learn, SQLAlchemy, scipy, datetime, geopy, folium, tsfresh, matplotlib, seaborn
Modeling: RandomForestRegressor, GradientBoostingRegressor, CatBoost, XGBoost, LightGBM, Keras,TensorFlow, Optuna
API and Cloud: Flask, FastAPI, Pydantic, AWS, Swagger
Containers and Orchestration: Docker, Kubernetes
Real-Time Data Use and Monitoring: Apache Kafka, ClearML