### **Summary of "Problem Statement and Metrics - Machine Learning System Design"**

#### **1. Overview**

The document outlines the design of a machine learning system to estimate delivery times for a food delivery service. The primary focus is on defining the problem, designing metrics, setting requirements, and ensuring the model's effectiveness through training and inference.

#### **2. Problem Statement**

* **Objective**: Build a model to estimate the total delivery time given order details, market conditions, and traffic status.
* **Scope**: The exercise does not consider batching multiple orders at restaurants.

#### **3. Metrics Design and Requirements**

* **Offline Metrics**:
  + Use Root Mean Squared Error (RMSE) to measure the accuracy of delivery time predictions.
  + RMSE=1n∑k=1n(predict−y)2RMSE=n1​∑k=1n​(predict−y)2
  + ​
  + Where nn is the total number of samples, predictpredict is the estimated wait time, and yy is the actual wait time.
* **Online Metrics**:
  + Use A/B testing to monitor RMSE, customer engagement, and customer retention.

#### **4. Training**

* **High Throughput**: The training pipeline should handle large data volumes efficiently, suggested by organizing data in Parquet files.
* **Frequent Retraining**: Models should be retrained multiple times per day to adapt to dynamic conditions such as traffic and weather. This helps balance overestimation and underestimation of delivery times.
* **Example**: Traffic conditions may worsen on game days, requiring the model to learn and adapt to these changes quickly.

#### **5. Inference**

* **Real-time Estimations**: The system needs to make real-time predictions for each delivery, ideally 30 predictions per delivery.
* **Near Real-time Updates**: Any change in delivery status, like meal preparation or driver start, triggers a new estimate and update to the customer.
* **Low Latency**: The system should capture real-time aggregated statistics with latency ranging from 100ms to 200ms, ensuring quick model scoring and customer updates.

#### **6. Summary**

* **Goals and Metrics**:
  + **Optimize for Low RMSE**: Ensure delivery time estimations are within 10-15 minutes. Overestimation may deter customers from ordering, while underestimation may cause dissatisfaction.
* **Training**:
  + Maintain high throughput and retrain multiple times daily.
* **Inference**:
  + Achieve latency between 100ms to 200ms for near real-time predictions.

This detailed framework ensures the machine learning system for estimating food delivery times is robust, adaptable, and efficient in handling dynamic conditions and large datasets, ultimately providing accurate and timely updates to customers.

### **Detailed Summary of "Estimated Delivery Model - Machine Learning System Design"**

#### **1. Overview**

The document provides a comprehensive guide on building a machine learning model to estimate delivery times for a food delivery application. It covers feature engineering, training data preparation, model selection, and the detailed workings of a Gradient Boosted Decision Tree (GBDT) model.

#### **2. Features Engineering**

The document outlines various features that can be used for training the delivery time estimation model:

* **Order Features**: Include order subtotal and cuisine type.
* **Item Features**: Include the price and type of items in the order.
* **Order Type**: Specify whether the order is a group order or catering.
* **Merchant Details**: Include the store ID and embeddings representing the store.
* **Realtime Features**: Include the number of orders, number of delivery personnel (dashers), traffic conditions, and travel estimates.
* **Time Features**: Include the time of day (e.g., lunch or dinner), day of the week, weekend, or holiday.
* **Historical Aggregates**: Include average delivery times over the past few weeks for different segments such as store, city, market, and time of day.
* **Similarity Features**: Include average parking times and variance in historical times.
* **Geographical Features**: Include latitude and longitude to measure the estimated driving time between the restaurant and delivery location.

#### **3. Training Data**

The training data is derived from historical deliveries over the past six months. This data includes:

* Delivery data and actual total delivery time
* Store data
* Order data
* Customer data
* Location data
* Parking data

#### **4. Model: Gradient Boosted Decision Tree (GBDT)**

The GBDT model is chosen for its effectiveness in predicting continuous outcomes. The document describes the step-by-step process of how GBDTs work:

1. **Initial Prediction**:
   * Calculate the average delivery time from historical data to use as a baseline.
2. **Residual Calculation**:
   * Measure the residuals (errors) between the predicted and actual delivery times.
3. **Building Decision Trees**:
   * Construct decision trees to predict the residuals. Each leaf in the tree represents a prediction for residual values.
4. **Update Predictions**:
   * Use the new predictions to update the estimated delivery time using the formula: EstimatedDeliveryTime=AverageDeliveryTime+learning\_rate×residualsEstimatedDeliveryTime=AverageDeliveryTime+learning\_rate×residuals
5. **Iteration**:
   * Repeat steps 3-5 until reaching the predefined number of iterations (hyperparameter tuning).

#### **5. Model Evaluation**

The document emphasizes the importance of balancing overestimation and underestimation:

* **Model 1**: Overestimates delivery time, which can prevent customers from making orders.
* **Model 2**: Underestimates delivery time, potentially causing customer dissatisfaction.

Both models may have the same RMSE, but the business impact of their errors differs. The decision on which model to deploy should consider customer experience and business outcomes beyond just RMSE.

#### **6. Conclusion**

The document summarizes the goals and metrics for the delivery time estimation model:

* **Metrics**: Optimize for low RMSE, with the aim for estimations to be within 10-15 minutes.
* **Training**: Ensure high throughput and the ability to retrain multiple times per day.
* **Inference**: Achieve latency between 100ms to 200ms for near real-time predictions.

This detailed framework ensures that the machine learning system for estimating food delivery times is accurate, adaptive, and efficient, providing a positive customer experience and meeting business needs.

### **Detailed Summary of "Estimate Food Delivery System Design - Machine Learning System Design"**

#### **1. Overview**

This document provides a detailed guide on designing a machine learning system for estimating food delivery times in a delivery app. It includes sections on assumptions, data size, scaling, system design, and scaling the design, along with follow-up questions and a summary.

#### **2. Calculation & Estimation**

**Assumptions**:

* **User Base**: 2 million monthly active users, 20 million total users, 300k restaurants, and 200k drivers.
* **Deliveries**: On average, there are 20 million deliveries per year.
* **Data Size**: For 1 month, data on 2 million deliveries, each with around 500 bytes of related features, resulting in 1 GB of data.

#### **3. System Design**

**Components**:

* **Feature Store**: Provides fast lookup for low latency. Suggested to use key-value storage like Amazon DynamoDB.
* **Feature Pipeline**: Reads from Kafka, transforms, and aggregates near real-time statistics, then stores them in feature storage.
* **Database**: Stores historical orders and delivery data. Training data is prepped from this database and stored in cloud storage like S3.
* **Services**:
  + **Status Service**: Handles real-time updates of order status.
  + **Notification Service**: Subscribes to message queues (Kafka) and receives real-time order status updates.
  + **Estimate Delivery Time Service**: Uses the latest ML model to predict delivery time and return results to the application server.

**User Flow**:

1. **Consumer/User**:
   * Requests estimated delivery time via the application server.
   * The application server sends the request to the Estimate Delivery Time Service.
   * The service loads the latest ML model, fetches feature values from the Feature Store, predicts delivery time, and returns results to the application server.
2. **Restaurant/Driver**:
   * Updates status (e.g., dish preparation, packaging) sent to the Status Service.
   * Status Service updates order status and queues the event in Kafka.
   * Notification Service receives the updated status in real-time.

**Model Training**:

* The system has a scheduler that handles retraining the model multiple times per day. The retrained model is stored in Model Storage.

#### **4. Scaling the Design**

**Scalability**:

* Services are scaled out to handle large requests per second.
* Load Balancer: Used to balance loads across multiple application servers.
* Kafka: Used for handling notifications and model predictions.

#### **5. Follow-up Questions**

1. **StoreID Embedding Efficiency**:
   * Evaluating if using StoreID embedding efficiently handles new stores.
2. **Model Retraining Frequency**:
   * Retraining frequency depends on online metrics. Infrastructure must monitor metrics, and retraining is triggered if metrics decline.

#### **6. Summary**

* **Formulation**: Estimated delivery times framed as a machine learning problem using Gradient Boosted Decision Trees.
* **Data Utilization**: Collecting and using historical data for model training.
* **Real-time Processing**: Leveraging Kafka for handling logs and model predictions, enabling near real-time predictions.

This comprehensive design framework ensures that the food delivery time estimation system is robust, scalable, and capable of providing accurate, real-time predictions to enhance user experience and operational efficiency.

ChatGPT can make mistakes. Check important info.

ChatGPT