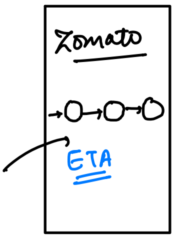


Real-time prediction

Data reception ----> Deployment

Functional Requirements

1. Accurate ETA predictions
   ↳ Real time prediction (optional)
2. Dynamic Updates → travel time, Restaurant prep time, order complexity, rider availability
3. Restaurant & Rider integration

Non-functional Requirements

1. Scalable → peak order processing
2. Low latency
3. High Availability / Reliability
4. Data Security & Privacy

Data Sources & Structures

1. Order data

   ↳ order id, restaurant id, payment info, delivery address, special instruction

   Tabular, Structured

2. Restaurant data

   Dist

   ↳ Restaurant id, name, address, location (lat/long), prep time (Chinese → , Mughlai →), operational hours, rating

3. Rider data

   [ ]

4. Customer data

   [Multiple locations
   - work
   - home]

5. Traffic data
   - Road n/w data
   - speed, congestion
   - historic traffic pattern
   - weather condition

   (Google Maps API)

Storage

Raw Data lake → Amazon S3 → General / Glacier

Structured data → AWS Redshift

Frequently accessed → AWS DynamoDB

Metadata Store → AWS Glue Data Catalog

Data processing & Feature Engineering

ETL (extract, transform, load) → Informatica, Alteryx, Dataiku DSS

Bonus → S3 AWS Lambda AWS Glue

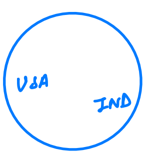
- Data cleaning → ——— ✓
- Data Transformation → ——— ✓
- Feature Engineering

Distances → Haversine formula

Time based

Restaurant features

Ride specific features



Feature encoding

- One hot encoding
- target encoding
- Ordinal encoding

+ Embeddings (Restaurant encoding)
https://blog.zomato.com/food-preparation-time

Feature Scaling → distance, time, Order

Lat / Longitude (Encoding?) → geohashing

Amazon → 5 precision (ETA ↑ & same)
Swiggy → 9 precision

POI → landmarks
Distance

Order instructions
+ TF IDF
+ word2vec, Glove

Model Selection & training phase

AWS → EC2 (Virtual Machines) → tranium

→ Linear Regression
→ XGBOOST, Lightgbm (
→ Neural Networks

Sagemaker (Training, inference, endpoint generation)
Deployed

★ Expose this endpoint as a API

//get prediction

ETA

AWS
Lambda functn

- Preprocess incoming data (encoding, transform)
- invoke sagemaker endpoint
- prediction
- post-processing

① User → Range of possible arrival time (Calculate) Mean

- confidence intervals

② ETA | Arrival (Actual)

Mean of both groups are diff

↳ significant

↳ hypothesis testing

Retuning is needed

↳ Add more feature

A B

Hypothesis Testing:

Model Performance Comparison: When comparing different models (e.g., Model A vs. Model B), use hypothesis testing to statistically determine if one model is significantly better than the other in terms of ETA accuracy (e.g., using paired t-tests or Wilcoxon signed-rank tests on prediction errors).
Null Hypothesis (H0): There is no significant difference in the mean prediction error between Model A and Model B.
Alternative Hypothesis (H1): There is a significant difference in the mean prediction error between Model A and Model B.

Impact of New Features: When introducing new features, use hypothesis testing to assess if the new features significantly improve ETA accuracy. Compare model performance with and without the new features.
H0: Adding the new feature does not significantly improve ETA prediction accuracy.
H1: Adding the new feature significantly improves ETA prediction accuracy.

A/B Testing for System Changes: When rolling out changes to the ETA prediction system (e.g., model updates, feature engineering changes), perform A/B tests to compare the new system version against the old version in a live environment. Monitor metrics like ETA accuracy, customer satisfaction, and rider efficiency. Use hypothesis testing to determine if the new version is significantly better.
H0: There is no significant difference in ETA accuracy or customer satisfaction between the old and new system versions.
H1: The new system version significantly improves ETA accuracy or customer satisfaction.

HW (needed)