The Bengaluru Last Mile Challenge

Submission Guidelines for the Leaderboard

As a testing ground for the code you are developing for the challenge, we encourage all teams to submit their code for evaluation to the leaderboard. The idea is to provide a mechanism to debug any errors that may come up so that the final submission will be smoother. **Therefore, you should do this even though your results may not meet the bar at this point.** The performance of your code (in terms of accuracy) will be evaluated and scored, and the names of the teams, along with the score, will be uploaded on the IEEE DataPort competition page. This will allow you to benchmark yourselves with respect to other teams.

Task 1:

For Task 1, your submission will be tested on a few sequences of live location data belonging to bus trips selected from different times of the day on different days. The selected trips are not part of the three weeks of training data that has been provided to you.

For a given sequence of live location data, which will represent the path of a bus trip for a certain duration, you are required to predict the arrival times at the remaining stops of the trip. The test sequence of live location data need not start from the beginning of the trip. There will be several such sequences on which your leaderboard submission will be scored.

Please note that we will not make public the route or trip information of sequences that are selected for evaluation. Your code needs to fetch the sequences for scoring purposes, process the sequences, and output the predicted arrival times at future stops for each sequence. However, the routes selected for testing do occur in your training data, and will not be new. Your output will be scored based on the ground truth (i.e., the actual arrival times) that are available with us.

Task 1: Scoring method

The submitted code will be run on K live-location sequences belonging to different trips. Let the number of future stops in the j-th sequence be denoted by N_j. For the j-th sequence, let the predicted times of arrival at the N_j future stops be $p^j_1, p^j_2, \ldots, p^j_{Nj}$, and let the actual times of arrival for the N_j future stops be $a^j_1, a^j_2, \ldots, a^j_{Nj}$. The overall Expected Time of Arrival (ETA) prediction accuracy score for the K sequences will be computed as:

$$ETA\ Score = \left(\frac{1}{K}\right) \sum_{j=1}^{K} \left(\frac{1}{N_j}\right) \sum_{i=1}^{N_j} i \times |p^j_i - a^j_i|^2$$

The lower the score, the better. Please note that the farther the stop, the higher the weight associated with its prediction. Thus, bad predictions of ETA of far away stops will be punished more by the formula. Also, if the prediction is missing for a particular future stop, say the i-th

future stop, then a default value of $|p^j_i - a^j_i|$=30 minutes will be assumed. If the prediction is missing for an entire route ID, a score of 10,000 will be assumed.

Task 1: Submission format

Each team needs to package their solution into a single Docker image to be considered for the leaderboard.

The docker image needs to be formulated with the CMD section accepting a JSON file as input (input.json). It should output a JSON file.

The input JSON file will take this form:
Input json
{ '1:'path_to_the_parquet_file', '2:'path_to_the_parquet_file', … }

The output JSON file should take this form:
Output Json - output.json
{
'route_id':{'stop_id':time_stamp, 'stop-id2':time_stamp,..},
'route_id':{'stop_id':time_stamp, 'stop-id2':time_stamp,..},
}.

Please note the time in the output json should be in **UTC (GMT)** format ex. **2025-08-21 11:33:41.**

You must host the docker image in the docker hub or GitHub container registry and send us the link to the image.

We highly encourage you to test your code before submitting. We will run your docker image on a few sequences of live-location data to evaluate your submission. The code will run on a machine of the following specifications:
CPU - Core i9 | GPU - RTX4090 | RAM - 64GB
Please ensure that execution requirements of your code do not exceed the above specs.

The submissions for the leaderboard need to be done on the Discord channel through the Submissions Bot. Please see the next page for more information.

Task 1: Bot Commands for submission on the BLMC Discord Channel

1. Command: `BLMC run`

Purpose: Run a Docker image for a team submission.

Usage

BLMC run <docker_image> --team-name <team_name> --submission-name <submission_name>

Parameters

- **docker_image** *(required)*: Name of the Docker image to run.

  - Invalid characters: `; & | > <` $`

- **--team-name** *(required)*: Your registered team name.

  - Invalid characters: `/ \ .. : * ? " < > |`
  - If you have space in your team name, write the team name in quotes

- **--submission-name** *(required)*: A unique identifier for the submission.

  - Allowed: lowercase letters (`a-z`) only

  - Length: Minimum 1 character, maximum 6 characters

Rules

- Each team is allowed a maximum of **3 submissions per day**.
- A **10-minute cooldown** is enforced between consecutive submissions.
- Submission counts reset every 24 hours.
- Please note the time format example in the output json **2025-08-21 11:33:41**

Example

BLMC run mydocker/image:latest --team-name “Alpha Team” --submission-name test1

2. Command: `BLMC status`

Purpose: Check the execution status of a submission.

Usage

BLMC status <team_name> <submission_name>

Status Indicators

- ✅ Succeeded
- ❌ Failed
- 💥 Error
- 🏃 Running
- ⏳ Pending
- ❓ Unknown

Example

BLMC status AlphaTeam test1

Response:

✅ AlphaTeam-test1: Succeeded

3. Command: `BLMC logs`

Purpose: Retrieve execution logs for a team submission.

Usage

BLMC logs <team_name> <submission_name>

Details

- Returns the most recent **200 lines** of execution logs.
- If logs exceed the message limit, they are shared as a downloadable `.log` file.

Example

BLMC logs AlphaTeam test1

4. Command: `BLMC score`

Purpose: Retrieve evaluation scores for a submission.

Usage

BLMC score <team_name> <submission_name>

Details

- Returns the **error score** of the submission.

Example

BLMC score AlphaTeam test1

Response:

The error for AlphaTeam-test1 is 0.023

Task 2:

For Task 2, your submission will be tested on a few instances of actual autorickshaw trips on Namma Yatri. For each trip, you will be given the time of the day when the trip was requested and the start and end locations (latitude and longitude) of the trip. The trip will be selected from different times of the day on different days. The selected trip dates and times are not part of the two weeks of data that has been provided to you.

For each trip, you are required to predict the time duration of three components: (i) the request acceptance time (duration), i.e., the time it takes to confirm a trip after it is requested, (ii) the pickup time (duration), namely, the time it takes for the autorickshaw to arrive at the pickup point after the pickup is confirmed, and (iii) the trip time (duration), i.e., the time it takes to reach the destination once the trip starts.

Let the number of trip instances be denoted by K. In the j-th trip. Let p^j_1, p^j_2 and p^j_3 represent the predicted request acceptance time (duration), predicted pickup time (duration), and predicted trip time (duration), respectively. Let a^j_1, a^j_2 and a^j_3 represent the actual request acceptance time (duration), actual pickup time (duration), and actual trip time (duration), respectively. The total trip time (TTT) prediction accuracy score for the K sequences will be computed as:

$$TTT\ Score\ =\ \left(\frac{100}{3K}\right)\sum_{j=1}^{K}\sum_{i=1}^{3}\frac{|p^j_i - a^j_i|}{a^j_i},$$

Where the durations a^j_i are assumed to be non-zero.
The lower the score, the better. Please note that the score is related to the percent deviation from the ground truth. Also, if the predicted time is missing for a particular component of a trip, **then a default value of $|p^j_i - a^j_i|$=30 minutes will be assumed.**

Task 2: Submission format

Input: input.csv

rid	ride_request_time	ride_start_location	ride_end_location
int	HH:MM:SS	coordinates	coordinates

Output JSON

```
{
1 :{pa: p1(minutes),pb: p2minutes),pc:p3(minutes)},
2 :{pa: p1(minutes),pb: p2(minutes),pc:p3(minutes)},
}
```

Task 2: Bot Commands for submission on the BLMC Discord Channel

1. Command: `BLMC-AUTO run`

Purpose: Run a Docker image for a team submission.

Usage

BLMC-AUTO run <docker_image> --team-name <team_name> --submission-name <submission_name>

Parameters

- **docker_image** *(required)*: Name of the Docker image to run.
    - Invalid characters: `; & | > <` $`
- **--team-name** *(required)*: Your registered team name.
    - Invalid characters: `/ \ .. : * ? " < > |`
    - If you have space in your team name, write the team name in quotes
- **--submission-name** *(required)*: A unique identifier for the submission.
    - Allowed: lowercase letters (`a-z`) only
    - Length: Minimum 1 character, maximum 6 characters

Rules

- Each team is allowed a maximum of **3 submissions per day**.
- A **10-minute cooldown** is enforced between consecutive submissions.
- Submission counts reset every 24 hours.
- Please note the time format example in the output json **2025-08-21 11:33:41**

Example

BLMC-AUTO run mydocker/image:latest --team-name “Alpha Team” --submission-name test1

2. Command: `BLMC-AUTO status`

Purpose: Check the execution status of a submission.

Usage

BLMC-AUTO status <team_name> <submission_name>

Status Indicators

- ✅ Succeeded
- ❌ Failed
- 💥 Error

- 🏃 Running
- ⏳ Pending
- ❓ Unknown

Example

BLMC-AUTO status AlphaTeam test1

Response:

✅ AlphaTeam-test1: Succeeded

3. Command: `BLMC-AUTO logs`

Purpose: Retrieve execution logs for a team submission.

Usage

BLMC-AUTO logs <team_name> <submission_name>

Details

- Returns the most recent **200 lines** of execution logs.
- If logs exceed the message limit, they are shared as a downloadable `.log` file.

Example

BLMC-AUTO logs AlphaTeam test1

4. Command: `BLMC-AUTO score`

Purpose: Retrieve evaluation scores for a submission.

Usage

BLMC-AUTO score <team_name> <submission_name>

Details

- Returns the **error score** of the submission.

Example

BLMC-AUTO score AlphaTeam test1

Response:

The error for AlphaTeam-test1 is 0.023

Task 3:

For Task 3, your submission will be tested on a few instances of actual multi-modal journeys consisting of a BMTC bus trip flanked on both sides by autorickshaw trips. For each journey, you will be given the starting location and time of the day and when the initial autorickshaw ride was requested, the bus stop where the autorickshaw trip ends, the route number and live-location sequences of all trips of the bus route that would have started before the autorickshaw trip was requested, the bus stop ID where the bus trip ends, and the final destination for the last autorickshaw ride. Please note that there could be several trips (of the specific route that the passenger wishes to take) which pass through the bus stop and the idea is to catch the first bus that shows up after the passenger reaches the bus stop. The passenger is assumed to request an autorickshaw ride as soon as he/she gets off the bus for the final destination.

For each trip, you are required to predict the time duration of seven components: (i) the request acceptance time (duration) for the first autorickshaw ride, (ii) the pickup time (duration), namely, the time it takes for the autorickshaw to arrive at the starting point of the trip after the pickup is confirmed, (iii) the first mile trip time (duration), i.e., the time it takes to reach the bus stop once the autorickshaw trip starts, (iv) the waiting time (duration) at the bus stop, (v) the duration of the bus trip from the embarking stop to the disembarking stop, (vi) the acceptance time (duration) for the autorickshaw ride, (vii) the waiting time (duration) for the autorickshaw pickup at the disembarking stop, and (viii) the time taken by autorickshaw to reach the final destination.

Let the number of journey instances be denoted by K. Let the following notation denote the various time durations in the j-th journey:

Prediction	notation	Actual	notation
Initial auto ride acceptance time (duration)	p^j_1	Initial auto ride acceptance time (duration)	a^j_1
First leg autorickshaw pickup time	p^j_2	First leg autorickshaw pickup time	a^j_2
First leg autorickshaw ride duration	p^j_3	First leg autorickshaw ride duration	a^j_3
Waiting time (duration) at boarding bus stop	p^j_4	Waiting time (duration) at boarding bus stop	a^j_4
Duration of bus journey	p^j_5	Duration of bus journey	a^j_5
Second autorickshaw ride acceptance time (duration)	p^j_6	Second autorickshaw ride acceptance time (duration)	a^j_6
Second autorickshaw pickup time (duration)	p^j_7	Second autorickshaw pickup time (duration)	a^j_7
Second autorickshaw ride duration	p^j_8	Second autorickshaw ride duration	a^j_8

The total multi-modal journey time (MMJ) prediction accuracy score for the *K* sequences will be computed as:

$$TTT\ Score\ =\ (\frac{100}{8K})\sum_{j=1}^{K}\sum_{i=1}^{8}\frac{|p_i^j - a_i^j|}{a_i^j},$$

Where the durations a_i^j are assumed to be non-zero.
The lower the score, the better. Please note that the score is related to the percent deviation from the ground truth. Also, if the predicted time is missing for a particular component of a trip, **then a default value of $|p_i^j - a_i^j|$=30 minutes will be assumed. Also, we are assuming that if the autorickshaw ride request does not get accepted (i.e., no autorickshaw is found) within 3 minutes, then the passenger will try again two more times, i.e., the passenger will make a total of three attempts. If no ride is found after three attempts (i.e., 9 minutes), the algorithm should output a default value of 30 minutes for the predicted acceptance time and "0" (zero) as the predicted durations for the rest of the components of the journey.**

Task 3: Submission format

To be updated**.**

General Rules and Guidelines

1. Both **team name** and **submission name** must be provided where required.
2. Team names are unique and confidential. Do not share your team name with other participants.
3. Docker images must be valid and free of any malicious commands or unsafe configurations.
4. Submission limits and cooldown times are strictly enforced by the system.
5. Please use the bot in personal messages to hide your team name
6. Please use your **exact team names**. If there is a space in your team name then use it in **" ".**

7. If multiple teams have the same score, they will be ranked based on date and time of submission, i.e., the team that has submitted earlier will be ranked higher.

Thank you for your participation!

P.S.*: *Teams that do not follow these instructions will not be evaluated.