OLA Bike - Rides Request Demand Forecast
Project Overview

Business Problem
Ola Bikes are suffering losses and losing out from their competition due to their inability to fulfill the ride requests of many users. To tackle this problem you are asked to predict demand for rides in a certain region and a given future time window. This would help them allocate drivers more intelligently to meet the ride requests from users.

Goal
You have to predict ride requests (demand forecast) for a particular latitude and longitude for a requested future time window/duration.

Data Description
Raw Data contains a `number` (unique for every user), ride request DateTime (IST time),
pickup and drop location latitude, and longitude.

Data Fields
1. number: unique id for every user
2. ts: DateTime of booking ride (IST time)
3. pick_lat: ride request pickup latitude
4. pick_lng: ride request pickup longitude
5. drop_lat: ride request drop latitude
6. drop_lng: ride request drop longitude

Defining a Good Ride Request

Ola Management knows the task is not easy and very important for their business to grow.
Hence, their business team has provided you some guidelines to follow.

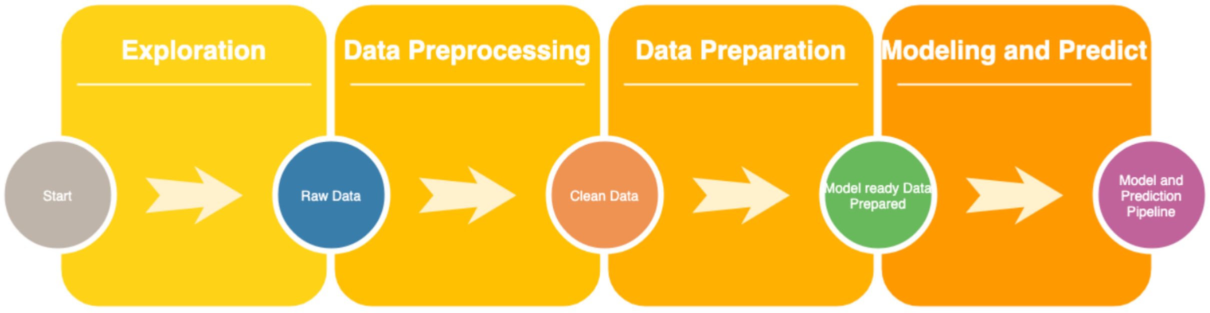
1. Count only 1 ride request by a user, if there are multiple bookings from the same latitude and longitude within 1hour of the last booking time.
2. If there are ride requests within 8mins of the last booking time consider only 1 ride
   request from a user (latitude and longitude may or may not be the same).
3. If the geodesic distance from pickup and drop point is less than 50meters consider that ride request as a fraud ride request.

4. Consider all ride requests where pick up or drop location is outside India bounding box: ['6.2325274', '35.6745457', '68.1113787', '97.395561'] as system error.
5. Karnataka is our prime city where we have a lot of drivers and ride requests to fulfill. We would not love to serve rides that are outside Karnataka and have pickup and drop geodesic distance > 500kms. Karnataka bounding box: ['11.5945587', '18.4767308','74.0543908', '78.588083']

Predict Task to test your model

After model development, Ola has requested us to build a prediction pipeline for the deployment of the model. To test our prediction pipeline they have provided us clean data (
filtered rides requests data based on good ride definition conditions) of 2021-03-26 based
on which they have requested us to predict/show initial hours rides request demand forecast for 2021-03-27.

Project Structure



Modular code structure

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. output
4. lib
5. requirements.txt - list of all the packages used in the project module.

1. The input folder contains all the data that we have for analysis.
2. The src folder is the heart of the project. This folder contains all the modularized code for all the above steps in a modularized manner. It further contains the following.
    a. ML_pipeline
    b. engine.py

The ML_pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the engine.py file

3. The output folder contains all the models that we trained for this data saved. These models can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
4. The lib folder is a reference folder. It contains two folders namely
    a. Data - Contains all the input data that you see in the videos
    b. Learning Resources - Contains all the learning resources
    c. Notebooks - Contain the jupyter notebook that you see in the videos.
    d. Scripts - The code in the notebook is divided into python scripts
    e. Models - The trained models are stored here

```
input
  |__clean_data.csv
  |__cleaned_test_booking_data.csv
  |__Data_prepared.csv
  |__raw data.csv
src
  |__engine.py
  |__ML_pipeline
           |__add_time_features.py
           |__advanced_cleanup.py
           |__clustering.py
           |__data_prep_advanced.py
           |__data_prep_basic.py
           |__data_prep_geospatial.py
           |__model_training.py
           |__regressor_evaluate.py
           |__shift_time.py
           |__shift_with_lag_and_rollingmean.py
           |__train_test_data_prep.py
           |__utils.py
           |__xgb_model.py
lib
  |__Data
        |__data_checkpoint
        |__test_dataset
        |__test_dataset_prediction_output
        |__clean_data.csv
        |__Data_prepared.csv
        |__raw data.csv
  |__Learning Resources
        |__1. Multistep Time Series.pdf
        |__2. TimeSeries Types.pdf
        |__3. Geodesic distance algorithm.pdf
        |__4. MiniBatchKmeans.pdf
        |__5. Autocorrelation.pdf
  |__Models
  |__Notebooks
  |__Scripts

output
  |__*contains all output models saved as joblib files*
requirements.txt
```

Project Takeaways

1. What is Multi Step time series forecast?
2. Advance Data Cleanup based on Business definition of a good ride
3. Feature Engineering - Time Feature Addition
4. Geospatial Feature Engineering
5. Understanding Mini-Batch K Means clustering
6. ACF and PACF
7. What is lead and lag?
8. What is Rolling Mean?