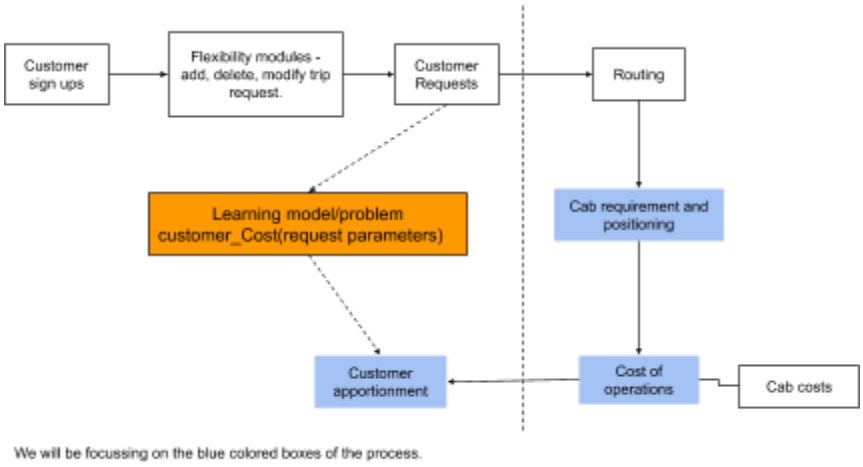
Mid term project report

Problem and Objective

We currently have the data sourced from a B2B Home-Office-Home commute service provider, MoveInSync. The service enables an employee of a company (that is subscribed to MoveInSync) to commute between their home and office on a daily basis.

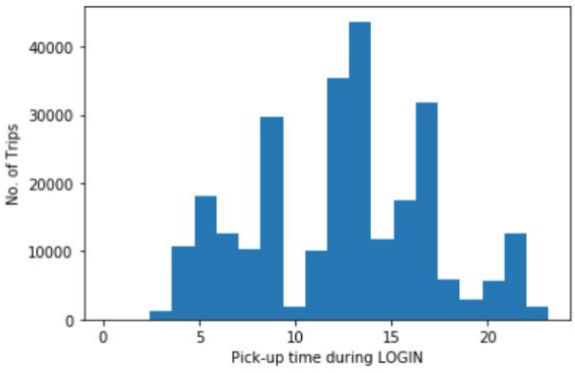
Banking on the fact that they can do better optimal shared routing (as they have travel requests in advance), the company is trying to attain price leader position in the market. We want to learn optimization, cost and attribution functions into a single function f. Such that Y = f(X), closely approximates steps done through routing, cab planning and cost apportionment to requests. This will help the company to show estimates for subscription requests in real time. (**similar to how uber/lyft show estimated price)**

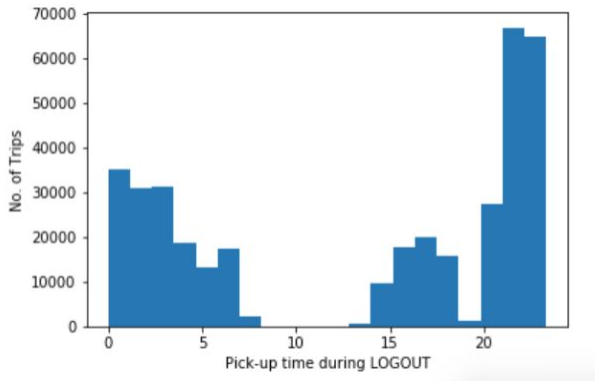


Exploratory data analysis

The dataset has 896047 rows and 22 columns, each row corresponds to the details of a single trip taken by an employee of a company over the months May, June and July 2019, each column provides details about the logistics of the trip, the details of the cab used and the employee.

- Self explanatory features: Shift_time, date, office_id, office_location, employee_id, Trip_id (int64), Vehicle_number, Vendor_id, Gender, planned_km, traveled_km, Num_employees_in_trip, cab_type, planned_pickup_time.
- Trip_type (LOGIN - The trip taken to the office, LOGOUT - The trip taken from the office)
- Planned_escort - TRUE if an escort was required to be accompanied for the trip. Needed in about 30% of trips.
- Employee_order - The order of pickup(for LOGIN)/drop(for LOGOUT) of the employee
- pickup_geo/drop_geo: Geolocation of the pick-up/drop

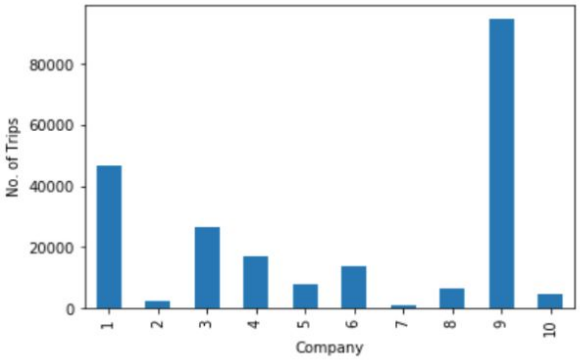


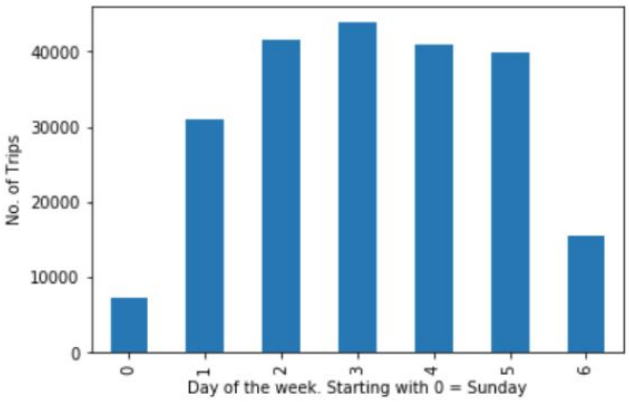


Key Data Points

We plotted a few histograms to confirm the sanity of the dataset. We analyzed how the shift timings were distributed between a trip to the office and out of the office. It can be seen from the histograms that the timings are complementary.

We analyzed the distribution of the number of trips taken by employees over the week, which were low during the weekends as expected.

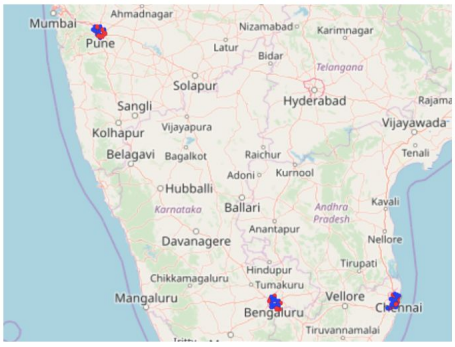


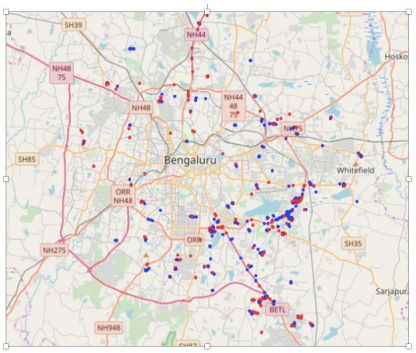


We further noticed that about 70% of the data corresponds to one single company (but we multiple office locations). We believe this would not affect our analysis, as we would want to develop an algorithm irrespective of the client's company.

Data cleaning

Most of the details provided for the trip were mostly available for every row in the dataset. One of the key features pertaining to cost are the location of the pick up and drop locations. Hence, we **decided to drop all the rows with no information of pick up and drop off locations**.





On further analyzing the pickup and drop off geolocations, we found that our trips were not focused in a particular city (Bangalore) as we had initially expected. Using simple thresholds on latitude and longitudes we were able to **filter out the corrupt data (data of other cities)** from the dataset.

We further found cases of **missing actual drop times** and **missing shift times** (ad hoc requests). Ad Hoc requests we treated by making a separate category and extracting approximate times (by comparing other people in the same trip id). We had to drop rows where shift times and actual times were all missing. Our final cut of the dataset contained **637,351 rows (viz a viz initial 896,047 rows).**

Feature transformations

We have knowingly **removed the other features which had obvious high correlation with these selected features**. For example, "Shift type" will have a high correlation with features "Hour of the day", "Quarter of the Hour" and hence "Shift type" is not considered in the feature set. Categorical features like "Hour of the day", "Weekday", "Month", "Office" are transformed using the **one-hot transformation**.

Further we have created **two features** - "Customer geohash density class" and "Number of traffic points on the way". "Customer geohash density class" is calculated by sorting number of requests arising from

the customer geo hash (approximately - 150 met*150 met) grid size. Second feature, “Number of traffic points on the way” is an integer output calculated by checking the proximity of pre-defined traffic points from the line connecting customer location and office location. We believe that these features will provide information about possible utilization (higher density area, higher cab utilization) and possible traffic delays.

Other attributes like - **features that are not available while request is being made. Hence, these attributes are not considered** in the input space of problem, e.g. Traveled_km, employee_order, Employee_count etc., planned pick up time (all these depend on eventual routing)

Feature selection

We had to remove certain features columns which had either all 0 or all 1 values. This was done to ensure **gram matrix remains invertible**. An example of such features is ‘month’ which had data only for 3 months and was zero for the remaining months.

```
In [640]: X = X_final
          y = per_person_cost
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
          regressor = LinearRegression()
          regressor.fit(X_train, y_train) #training the algorithm
          print(regressor.intercept_)
          print(regressor.coef_)

          [-1.36325324]
          [[ 0.44213592  0.41370553 -1.13813509  0.28590416 -0.91836063 -1.3897368
             1.11605638  0.83277661  1.02086962 -0.6652157  -0.73131516  0.16200575
            -0.53196218  0.26195398  0.41055995  1.1293867  -0.70062904 -0.30758602
             0.30758602  1.8009403   2.2267951   0.36919295  3.6724378   0.31693375
            -1.82243663  3.28942644  0.5165297 ]]
```

With remaining features, we ran a linear regression model to understand importance for each selected feature. As seen from screenshot below, weights for linear regression are quite balanced indicating that feature set is balanced (no one feature dominates).

Preliminary model results

Our preliminary model (a linear regression model) indicates high error rates (about one-third of mean value) across training and testing sets - **indicating a possibility of underfitting**. We will be adding more features to overcome this issue.

```
In [645]: y_pred_test = regressor.predict(X_test)
          y_pred_train = regressor.predict(X_train)
          print ("y_mean = ", np.mean(y))
          print('Absolute Mean Squared Error for training data:',
          print('Absolute Mean Squared Error for test data:', met

          y_mean =  9.420258389615231
          Absolute Mean Squared Error for training data: 2.648420
          Absolute Mean Squared Error for test data: 3.0490562580
```

Next Steps

1. Adding more features to remove underfitting (approx 1 week of effort)
2. We will be working on following model classes: (approx 1 week of effort)
    a. Quadratic loss with no regularizer.
    b. Quadratic loss with L1 regularizer. (Make sense as out input feature space is sparse owing to the
    c. Quadratic loss with L2 regularizer. (Ridge regression)
    d. Huber loss with no regularizer.
    e. Huber loss with L1 regularizer.
    f. Huber loss with L2 regularizer.
3. Model selection: Split all data randomly into 80%-20% training-testing set. Cross validation with 60%-20%. (approx 0.5 week of effort)
4. Numerical algorithm to find optimal number of cabs required.(approx 0.5 week of effort)