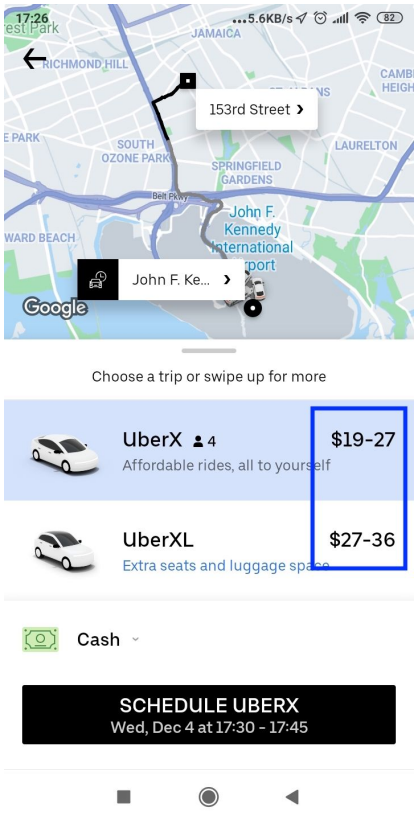
ORIE 4741
Final project report

Submitted by: Aman jain (aj644), Vineel Yellapantula (vy45)

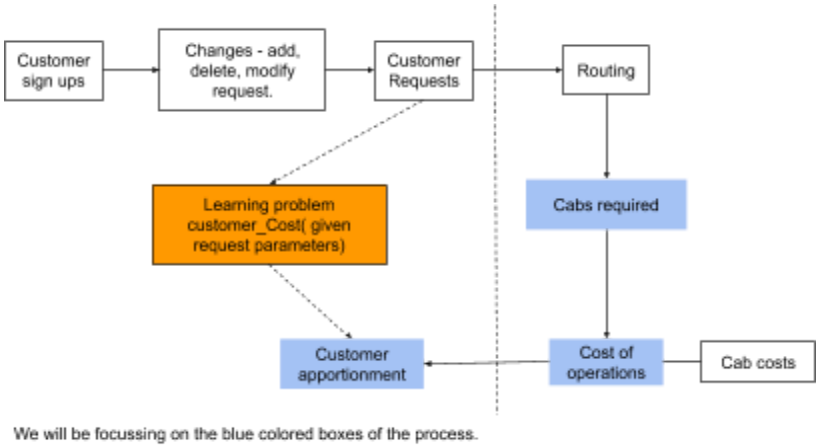
Describe the problem

We have the data sourced from a B2B Home-Office-Home commute service provider, XYZ. The service enables a passenger of a company to commute between their home and office on a daily basis - without having to make a booking daily. Banking on the fact that they can do optimal routing (as they have travel requests in advance), the company is trying to attain price leader position in the market. This leads them to cost based pricing of service.



The company needs to show estimates for subscription requests in real time while the customer is on the booking flow. (Similar to how uber/lyft show estimated price, refer screenshot on right).In order to arrive at the cost of providing this repeat service, three key components are important to understand. These are:

- Routing and cab utilizations.
- Number of cabs required and cost of cabs.
- Cost attribution to each passenger.



We will be focussing on the blue colored boxes of the process.

However, in order to show expected price to customer while booking, it is not really possible to complete information about three components discussed above. Hence, we want to learn these three components into a single function f. Such that Y = f(X), closely approximates steps done through routing, cab planning and cost apportionment to requests.

Data set description

In order to comply with the information security policies of the company, we have avoided to publish any explicit data points here.

- **Raw data contains of 896,047 rows** with following columns.
- Each raw is one request (One passenger travelling from one office to their home (or vice versa) on a given day). This data represents 3 months requests / travel data for city of Bangalore, India.

Datag group	Field	Datatype	Description
Company attribute	bunit_id	Text	This is client id (company of passenger).
	profile_office	Category	Office id to/from passenger travels.
Passenger attribute	Gender, passenger_id	Category, Int	Gender of passenger. passenger id. Used as unique key
Locations	Pickup_geo, drop_geo	float	Latitude and longitudes. Upto 6-7 digits of decimal.
Time attribute	shift_time_type	Text	Target Time to reach office or target time to depart from office.
Trip attribute	trip_id	Int	ID for the trip.
	passenger_order	Int	Order of boarding/deboarding the vehicle.
	trip_state_text	Category	Completed / Cancelled / Planned.
	need_security guard	Binary	If a security guard is required as per govt law.
	trip_type	Binary	Login (coming to office)/ Logout (leaving from office)
	planned_km	Float	Road distance between pick and drop locations.
	planned_pickup_time	Text	Planned pick up time for each passenger in the trip.
Vehicle attribute	vendor_id	Text	Vendor ID, who provides vehicle for a particular trip.
	actual_cab_registration	Text	Registration number of Vehicle doing the trip.
	actual_vendor_cab_id	Text	Cab id in company's system.
	cab_type	Category	Category of vehicle used - 4 wheeler, 6 Wheeler etc.

Data Imputation

As discussed in the coursework, we looked at the following imputation techniques for various missing data points. We were able to recover part of this using mentioned imputed methods.

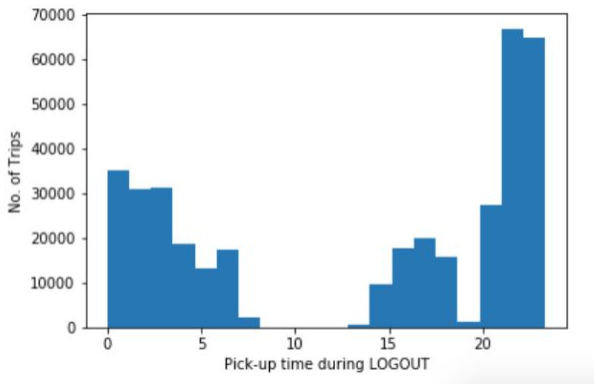
Missing data points	Volume	Imputation method	Volume recovered
"Duplicate" data	74920	Dropped	NA
"Out of city".	20691	Dropped. The trips belonged to cities other than Bangalore.	NA
"no lat long", "ad hoc" requests Pick/drop locations missing	119990	- Take all lat longs for Pick/Drop for same passenger ID travelling in same trip direction (login or logout). - Trim to 3 digits post decimals (within 100 met accuracy). - Take mode of these truncated lat longs.	15%

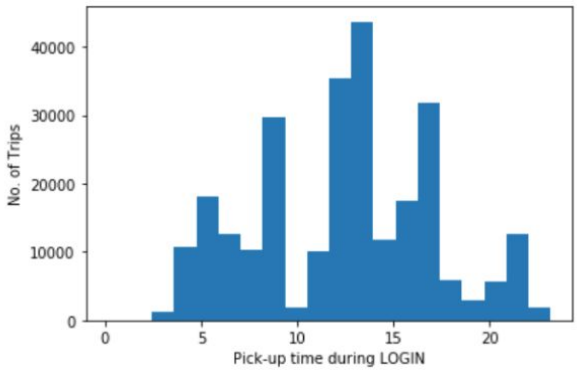
"non shift" requests - Pick/drop locations and time stamps missing.	118033	- Latlong recovered as per above mechanism. - Get time for nearest trip id mapped to given office location. Adjust this with distance between home and office (Assuming a constant distance).	10%
Removing 0 distance	1181	- Latlong recovered as per above mechanism. - Distances then computed as aerial distance*1.4. Aerial distances are basically geometric distances, taking into account curvature of earth.	30%
“Gender”/ “ID”missing	2028	- Dropped the columns with no passenger ID information.	80%
“Planned time” missing	110207	- Group the trips based on passenger ID information and trip_type(Login or Logout)	40%

After cleaning and imputation, our final cut of the dataset contained 510,778 rows.

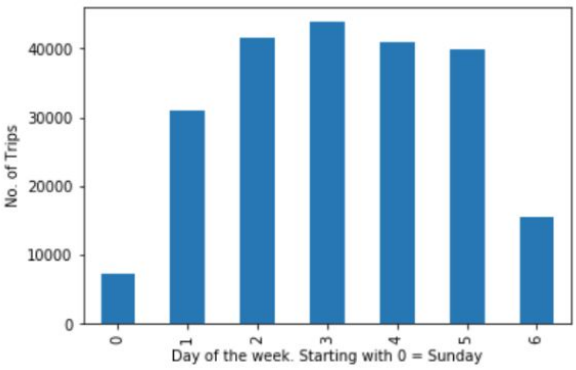
Exploratory Data Analysis

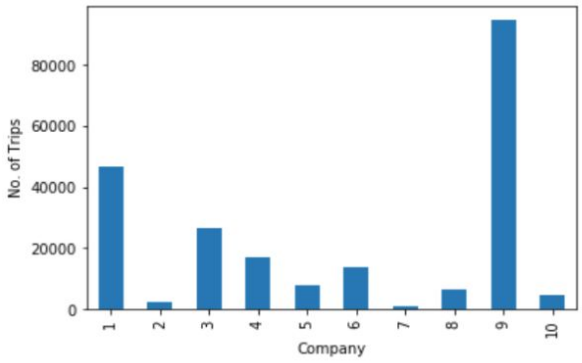
- Planned_security guard - Needed in about 30% of trips.





- 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.

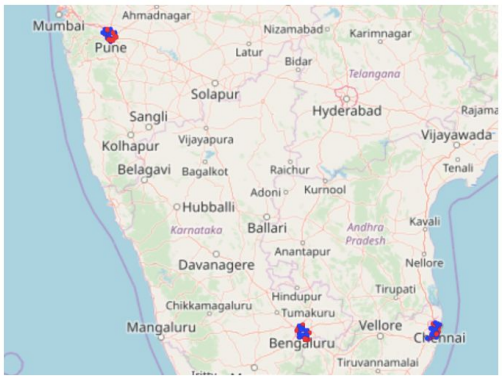


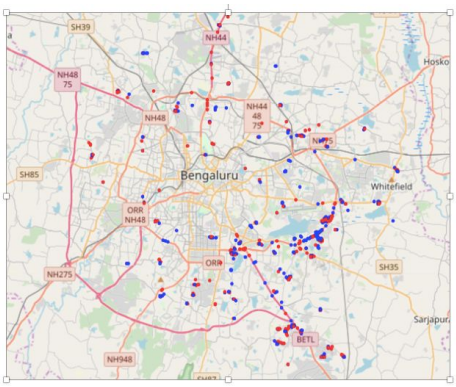


- We analyzed the distribution of the number of trips taken by passengers over the week, which were low during the weekends as expected.
- We further noticed that about 40% of the trips taken belonged to a single company. We believe this would not affect our analysis, given that the price of the trip is also a function demand, the learning model should be able to

adjust the cost of a new request based on an area's demand appropriately.

- 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. Red dots correspond to LOGIN and blue correspond to LOGOUT - which looks to be equally distributed.





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

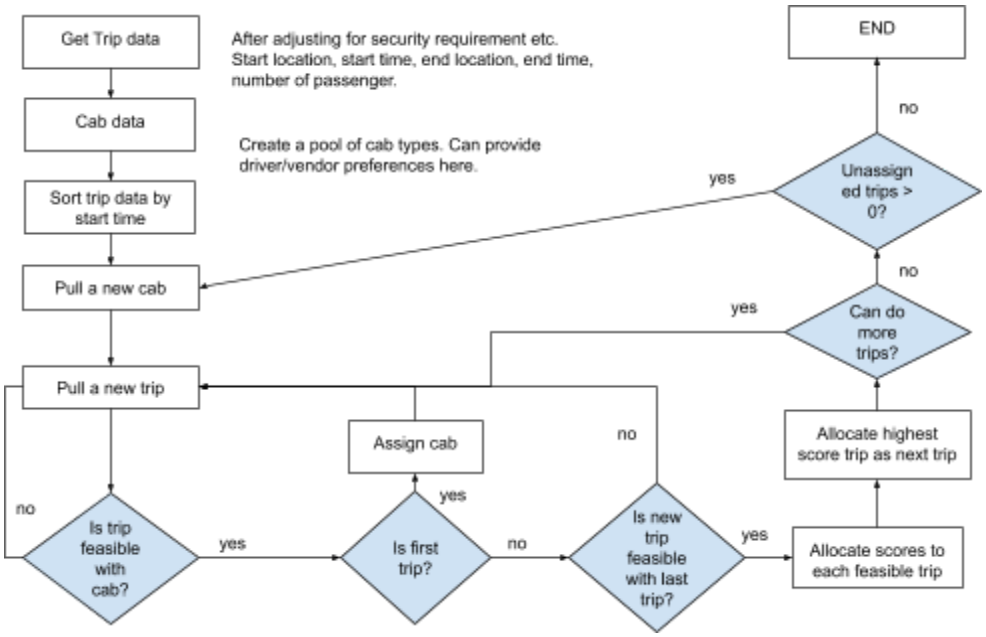
Approach

As discussed in the problem description, we divided problem in 2 parts:

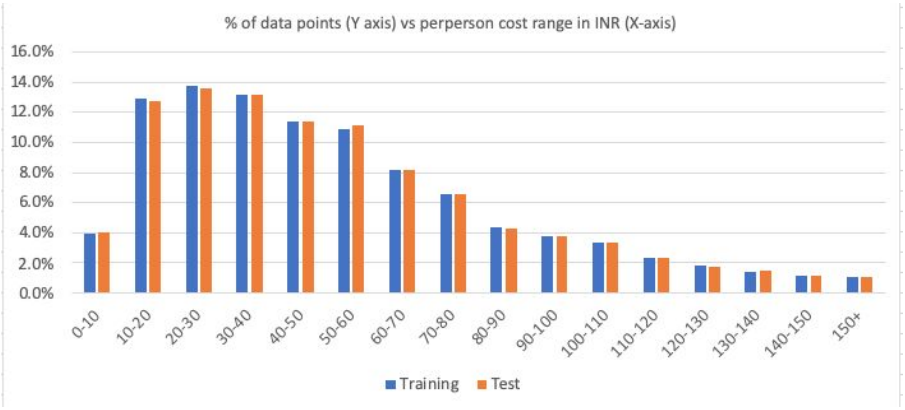
- Part 1: Calculating cost per passenger cost.
- Part 2: Setting up learning problem to predict this cost while customer is on the booking flow.

Part 1: Calculating cost per passenger cost (Y).

- Routing and cab utilizations - leveraged ‘passenger order’ data available in raw data to reconstruct routes.
- Number of cabs required and cost of cabs - used following workflow to develop a customized algorithm to come up with number of cabs required. Used approximate monthly contracts with vendors to arrive at total cost.
- Cost attribution to each passenger - Total cost of operation is divided among different trips based on distance traveled and if escort is required. Each trip’s cost is then attributed among passengers based on distance and time by each passenger.



Overlapping histograms (On the right panel) shows the distribution of calculated per passenger cost for both training (Blue) and test data set (Orange).



Part 2: Setting up learning model

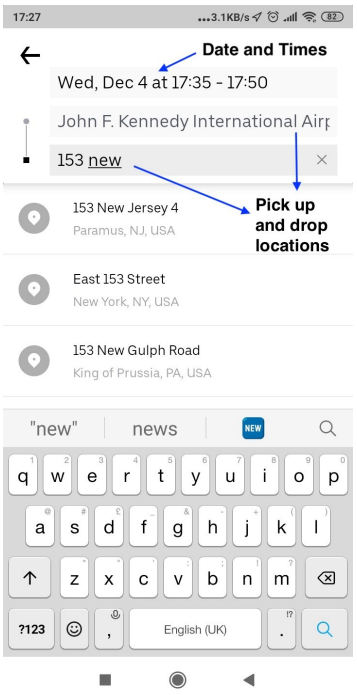
Features: While deciding on the features to work with, we wanted to ensure that these features (or underlying information) are available while the user is in the booking flow, or could be derived from the information available at that stage. Consequently, most of these features are based on location and time inputs.

Model Classes and selection: We looked at primarily linear and decision tree model. Intuition behind linear model is continuous output. Decision tree model class made sense as quite a few features are binary/categorical - having step effect on output. In addition to this we suspect mild interaction between features (e.g. particular office having higher requests for an early morning or night shift). Hence the idea is to have learning models break down dataset into smaller subsets (like a decision tree) and learn on these subsets.

Error metrics: As discussed in the problem description, we are looking at a range type of output. However, we would want to keep range as narrow as possible. Further, customer experience might constraint as common width of range (to reduce cognitive load of customers). Keeping this context in mind, we chose to look at mean absolute error as key error metrics. Smaller this is, narrower is the range of prediction.

Training budget: As model can be trained offline, training time is not really a constraint. We can afford training time of few hours.

Response times: As we are looking to deploy model in a real time system (booking flow), response time (time to make a prediction for a single request) might be a deal breaker. Keeping network latencies in mind, we try to keep this below 20 milliseconds



Feature transformations

While at the booking screen, we have the following information at our disposal: Date of booking, Time of booking, Pick up location and Drop location.

One hot feature transformation is used for ***Hour of the day, Quarter of the day, Weekday, Month, Profile_office, Cab_type features.*** ***Trip_type*** is the binary features used. ***Planned_km*** is a continuous float value feature. Polynomial (degree = 3) transformation has been done for it.

Further we have derived two key 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.
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. 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, passenger_order, passenger_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.

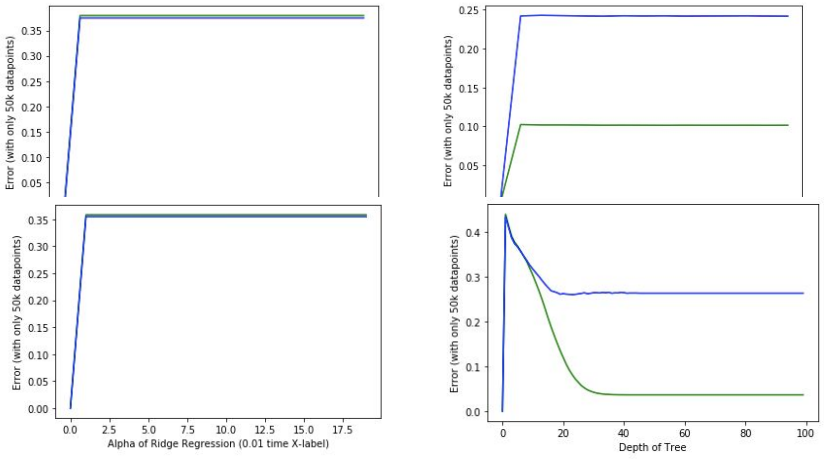
With remaining features, we ran a linear regression model to understand importance for each selected feature. As seen from the screenshot below, weights for linear regression are quite balanced indicating that feature set is balanced (no one feature dominates).

Algorithms used

- **Linear Regression:** Quadratic loss with no regularizer.
- **Huber Regression:** Huber loss with no regularizer.
- **Ridge Regression:** Quadratic loss with L2 regularizer. (alpha = 0.1, 0.2, 0.3…...0.9)
- **LASSO Regression:** Quadratic loss with L1 regularizer. (alpha = 0.1, 0.2, 0.3…...0.9)
- **LARS Lasso Regression:** Slightly modification to LASSO. Time complexity to calculate all possible Lasso estimates for a given problem is better than normal LASSO models. (alpha = 0.1, 0.2, 0.3…...0.9)
- **Bayesian Regression:** The response, y, is not estimated as a single value, but is assumed to be distributed normally with mean equal to wX and variance as sigma.
- **DT regressor:** Decision tree regressors basically divide large data sets into smaller data sets based on attributes (what a decision tree does), then regress on individual small parts and eventually combine them.
- **RandomForestRegressor:** Same as DT regressor. Different in way DTs are generated.

Hyperparameter fine tuning (Grid Search)

Hyper parameters were fine tuned using grid search methods. Following graphs shows error vs configurations for certain models above.
Top left: Alphas for LASSO
Bottom left: Alphas for ridge regression.
Top right: Number of estimators for RandomForest.
Bottom right: Depth of decision tree DT regressor.

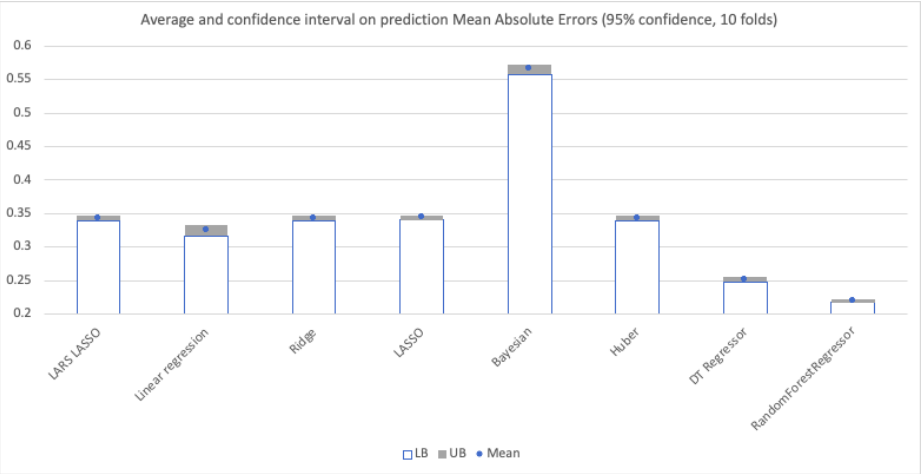


Grid search for fine tuning parameters for selected model:

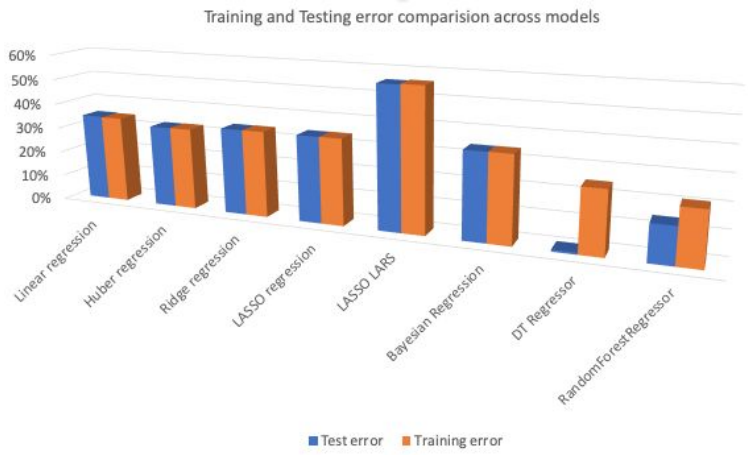
Parameter	'n_estimators'	'max_features	'max_depth'	'min_samples_split'	'min_samples_leaf'	'bootstrap'
Parameters Space	[500,750,1000,1250,1500]	['auto', 'sqrt']	[5, 10, 15, 20, 25, 30, 35,40]	[2, 5, 10]	[1, 2, 4]	[True, False]
Best	1000	'auto'	35	10	1	TRUE

K-fold Cross validation for Model Selection

We have done 10 fold cross validation for all the candidate models (after fine tuned hyper parameters). As shown on the graph on the right, **RandomForestRegressor has lowest error and lowest variance as well**, consequently we selected it.



We further looked at the test and training errors. For RandomForest regressor test and training errors were found to be comparable.

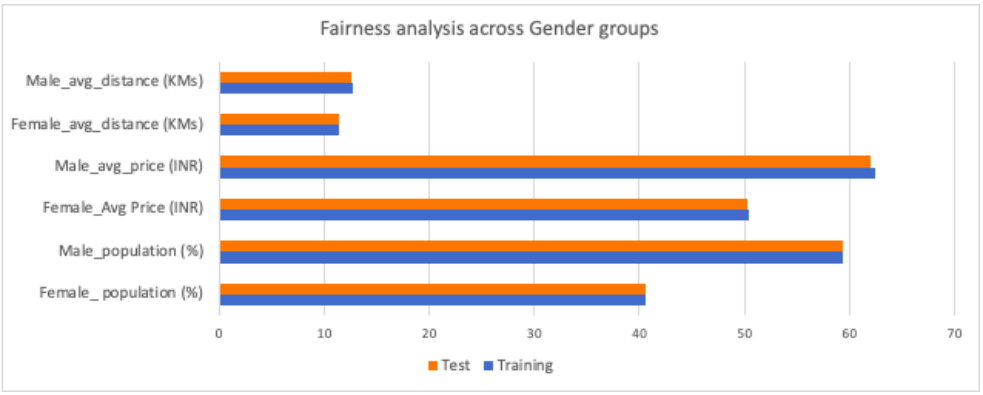


We found training time for all models with 2 hour (which will allow for once a day or once a week training). Response time on a single test request was also less than 10 milliseconds for all models - allowing for real time implementation.

Limitations, Risks and Fairness

Although, predictive model has some business risks in terms of financial impact on company's operations, however, **we do not believe this predictive model to be a weapon for mass destruction**. Our reasons to believe so are:

- Although outcome is not directly measurable, but some directional tests are possible in real time.
- Negative consequences are limited to the company itself, it does not have any wider social impact as such.
- There is no feedback loops, costs of company operations are independent of revenue.



Fairness is important in terms predicted price to be independent of the gender of person making request. In this regard, cost of security guard (which is mandated by government laws for women's safety) should be equally divided among all trips - not trips with women. To ensure this, we **masked the gender feature**. Secondly, we looked at the **demographic parity metric** of distribution to training and predicted costs for each gender and found that to be comparable. Slightly higher price for male passengers can be apportioned to larger distances for them.

Results and Conclusion

Based on Cross validation results we select RandomForestRegressor as final model. We also fine tuned hyperparameters of the selected model. This model yields mean absolute error of about 22% at a 95% confidence interval.

Further model training and response time (keeping real time application in mind) are within acceptable limits. This gives us confidence to put model in production to drive company's pricing decision decisions. However, to understand real life financial impact of model, we suggest to run it in real life for about 2 weeks as a trial in small area/city before full roll out. Also, having historical trip data going back a couple of years might help the model in understanding customer flow based on public holidays and the weather.

We believe, Uber can be used as an external validation to understand goodness of measure. Ubers shows a range of $19-$27 (refer screenshot on first page). It can be interpreted as mean $23 and +-17% range width. Corresponding width for our model will be 22% of the mean value. Although, we should look at more data points, at more locations and time points, to draw anything conclusively - however, this ballpark numbers gives us confidence in the model, given the limited data and time on hand.

Finally to conclude, we believe we have made good first production model for given problem statement. Having said that, there is definitely scope of future improvement in terms of improving accuracy of features using google api data, building speed into model etc.

Appendix

Models used:
http://statweb.stanford.edu/~tibs/ftp/lars.pdf
https://towardsdatascience.com/introduction-to-bayesian-linear-regression-e66e60791ea7
https://www.saedsayad.com/decision_tree_reg.htm
https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/decisionForests_MSR_TR_2011_114.pdf

Others:
https://github.com/JuliaPlots/ExamplePlots.jl/blob/master/notebooks/scratch/hist.ipynb
https://stackoverflow.com/questions/
https://pandas.pydata.org/pandas-docs/stable/reference/api/
https://www.oracle.com/assets/realtime-responses-big-data-wp-2524527.pdf
https://www.kdnuggets.com/2019/01/monitor-machine-learning-real-time.html
https://towardsdatascience.com/architecting-a-machine-learning-pipeline-a847f094d1c7
https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/decisionForests_MSR_TR_2011_114.pdf