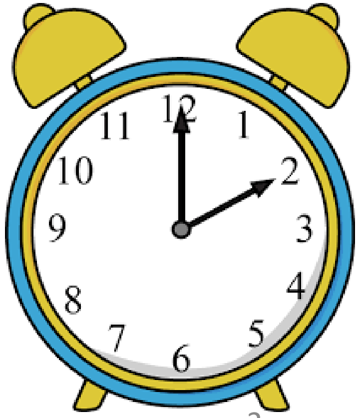
Next Location Prediction Challenge using

mobility traces of taxi cabs in San Francisco

Why the next location problem for taxi cabs is important

- Optimize efficiency of movement that can reduce traffic jams
- Reduce taxi idle times (time spent looking around for passengers)
- Reduce customer wait time



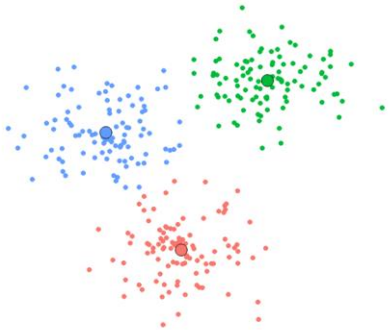
Outline of approach

Data cleaning

- Converting the data into a tidy format
- Each observation is a ride
- Based on consecutive pattern recognition

Exploratory Data Analysis

- Summary statistics
- Distribution of rides, cabs
- Feature associations
- Outlier identification

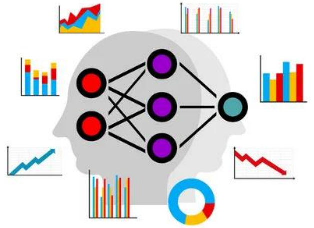


Region Detection

- Kmeans clustering
- Elbow method
- 11 clusters
- Weekdays, hour of the day, previous week's demand, week, minute of the day

Convert to ML problem

- Binary classification problem
- Predict pickup for each region individually
- Class up sampling for class imbalance problems
- One hot encoding of weekday variables



Model Training and Evaluation

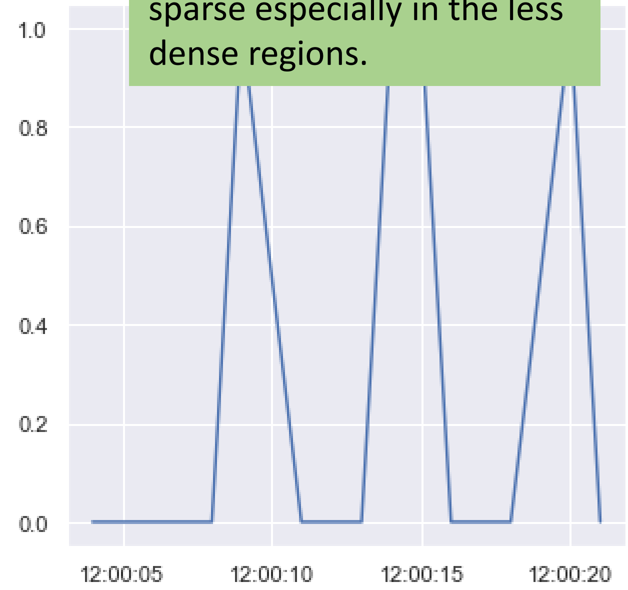
- Random guess of Bernoulli trials (baseline model)
- Logistic regression
- XGBoost
- LSTM
- ROC, AUC model evaluation

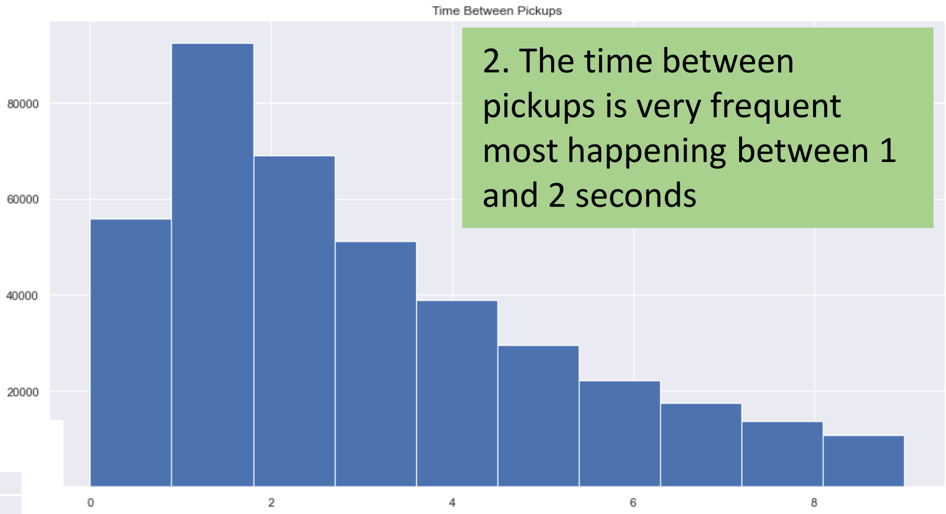
Feature engineering

What do we know about the data?

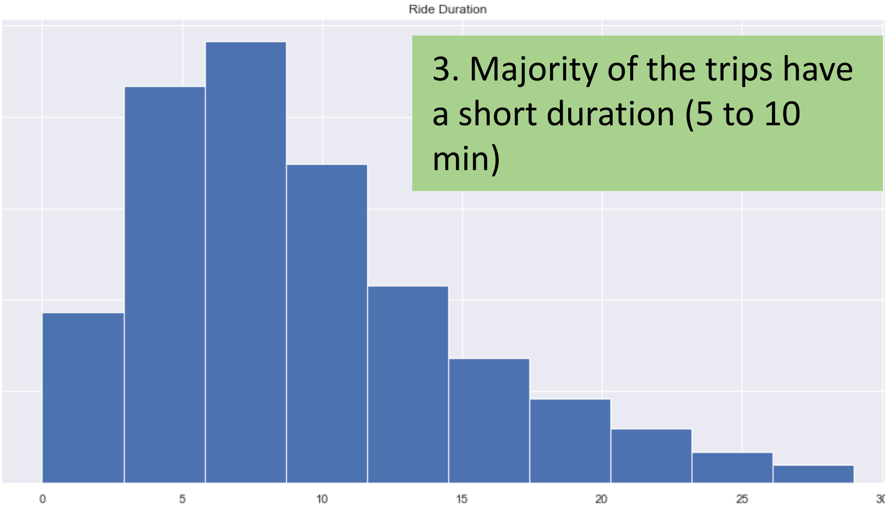
pickup_time \ cluster	0	1	2	3	4	5	6	7	8	9	10
2008-05-17 12:00:04	0	0	0	0	0	1	1	0	0	0	0
2008-05-17 12:00:05	0	0	0	0	0	0	0	0	0	0	1
2008-05-17 12:00:07	1	0	0	0	0	1	0	1	0	3	0
2008-05-17 12:00:08	0	0	0	0	0	1	0	0	0	1	0
2008-05-17 12:00:09	0	0	1	1	0	0	0	0	0	1	0

4. At the region level the time between rides is more sparse especially in the less dense regions.

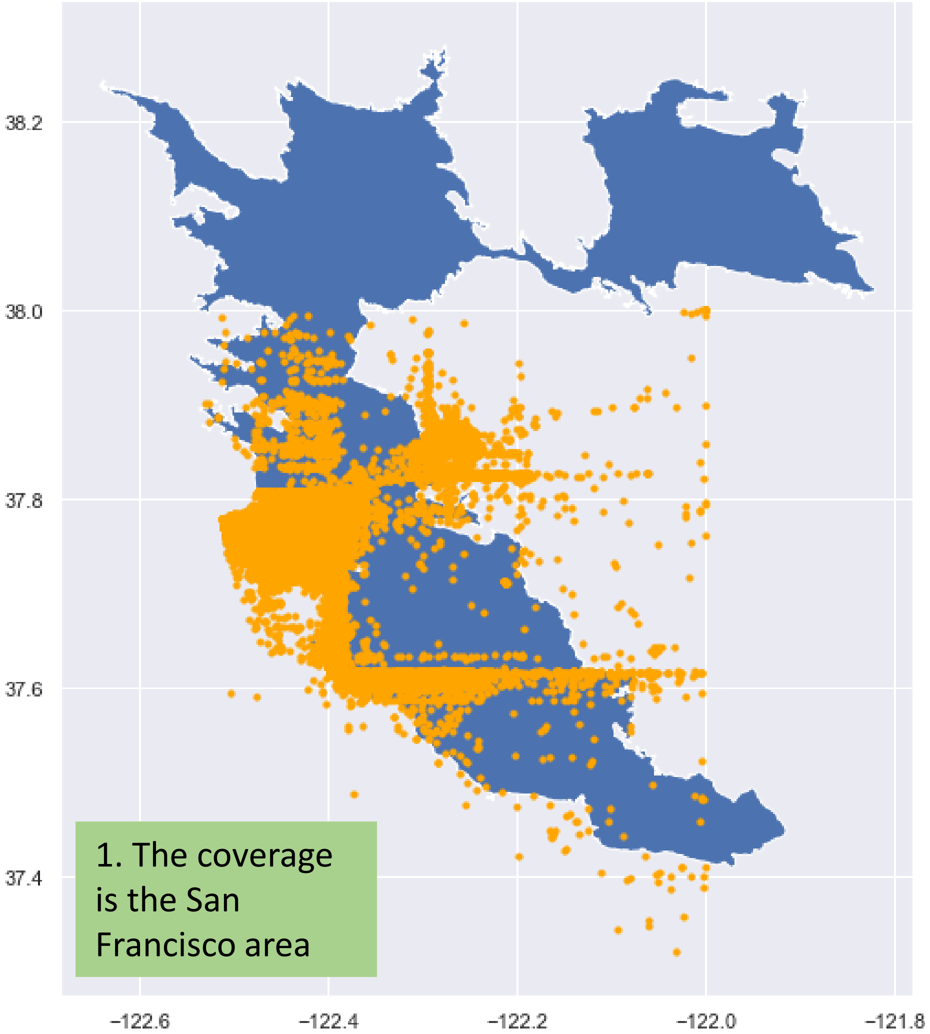




2. The time between pickups is very frequent most happening between 1 and 2 seconds

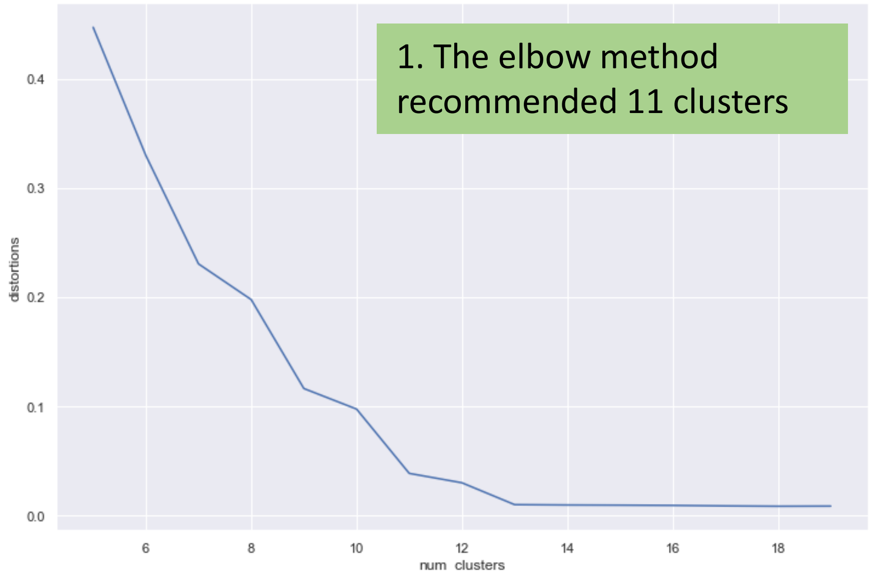


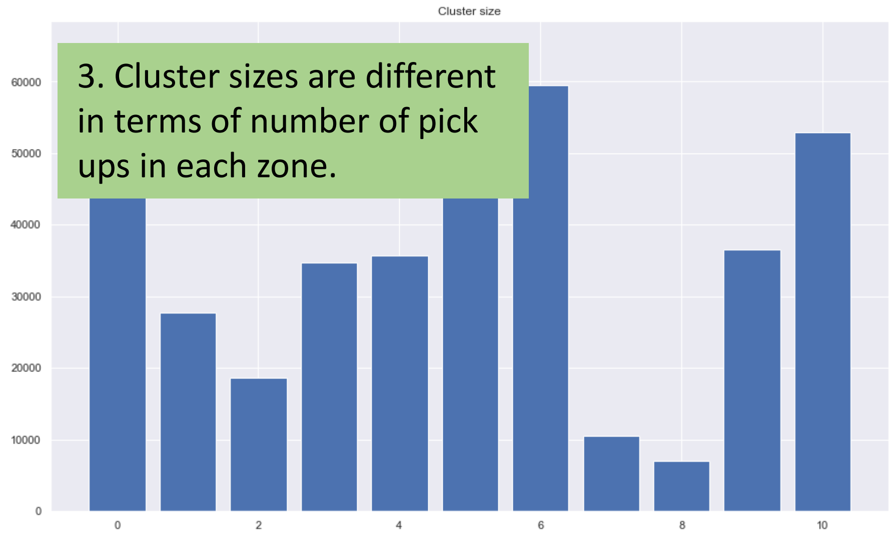
3. Majority of the trips have a short duration (5 to 10 min)



1. The coverage is the San Francisco area

The Clusters

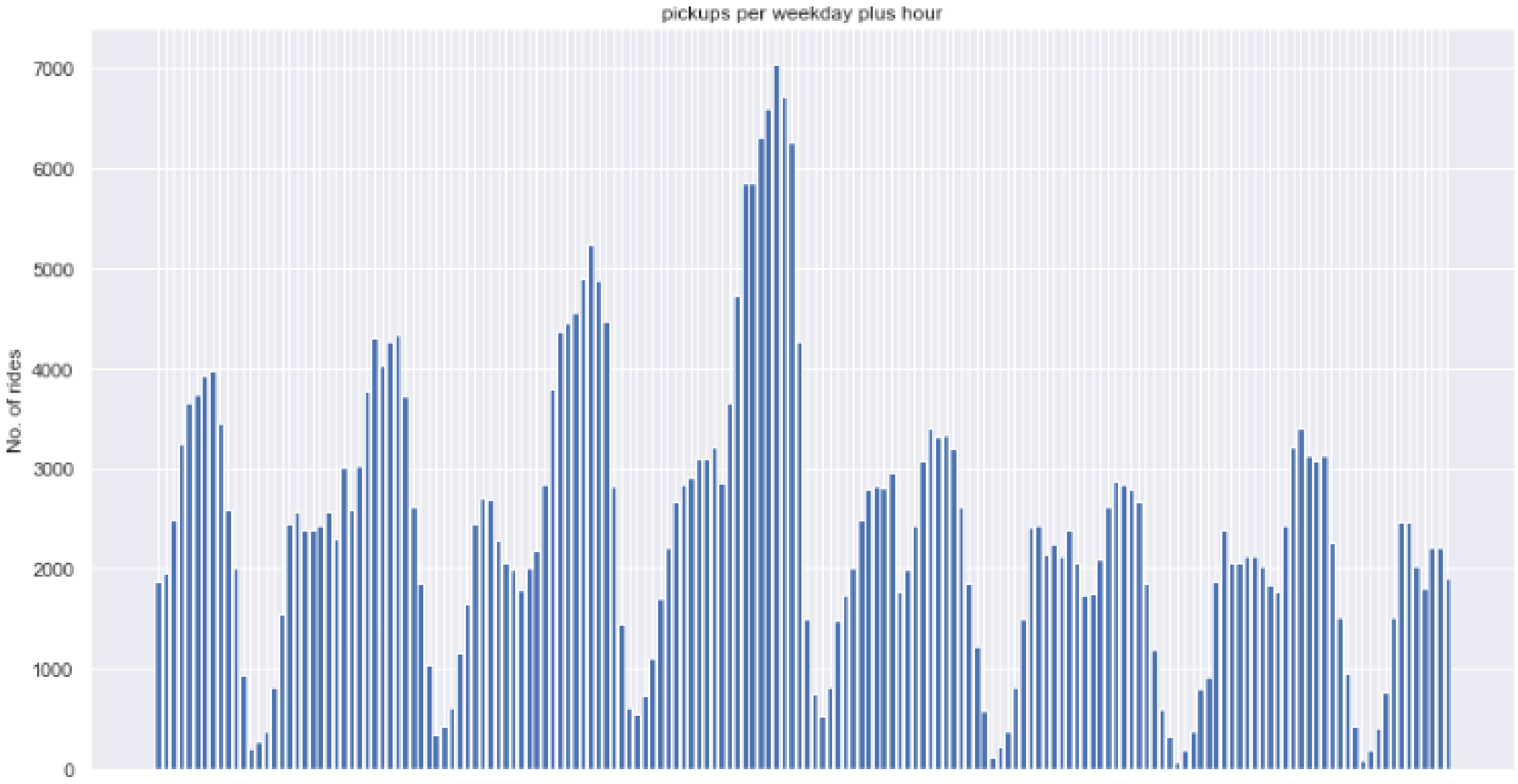




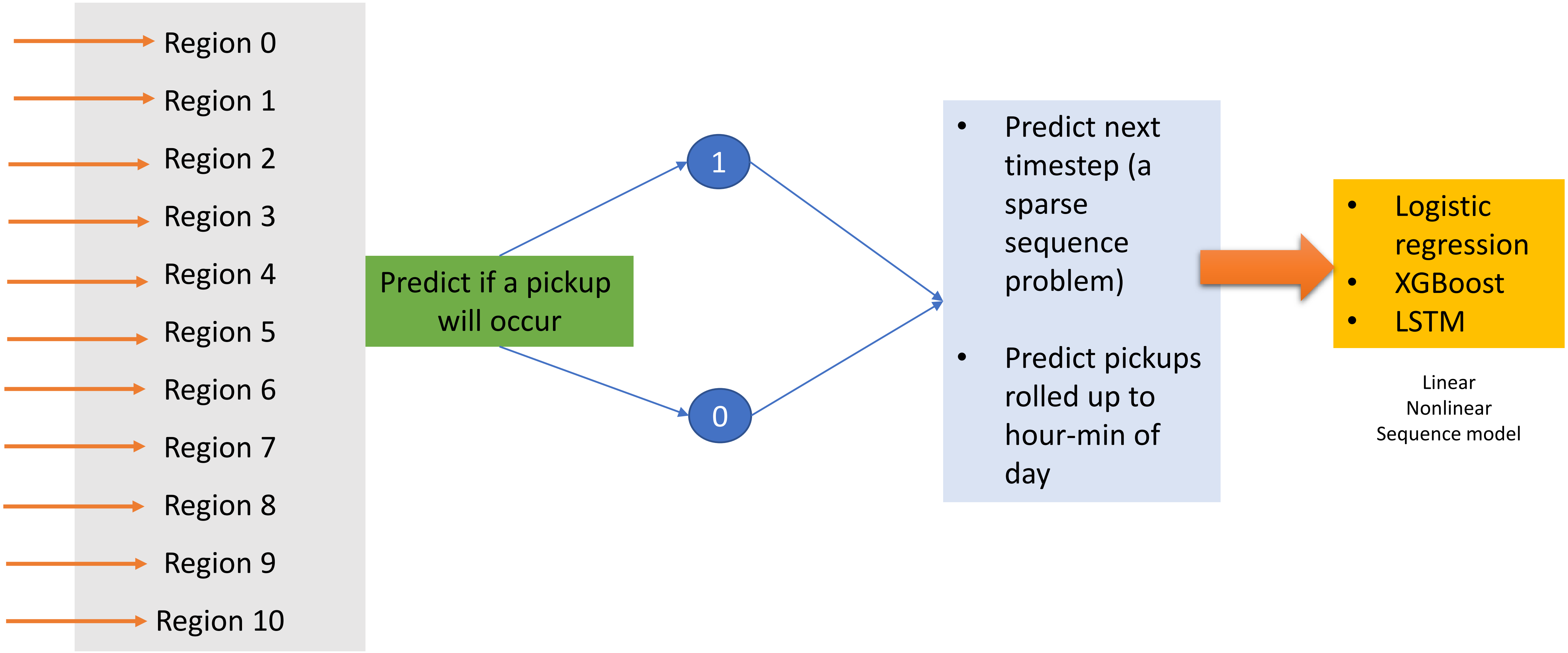
Feature Selection

The EDA revealed that there is much variability in:

- Count of rides each hour (e.g. Not many rides at midnight)
- Count of daily pickups day of the week (e.g. Saturday was most busy)
- Variability within zones
- Weekly dependencies
- No. of pickups in the previous timestep (for the LSTM model)

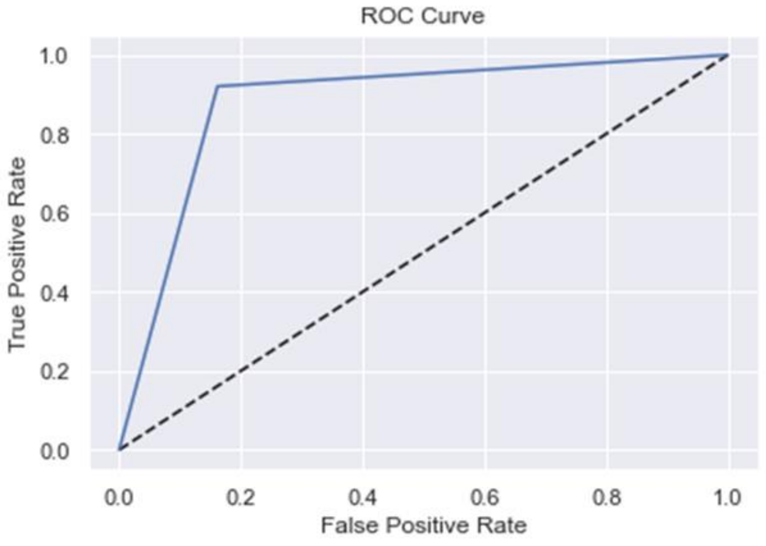


Machine Learning Model Formulation

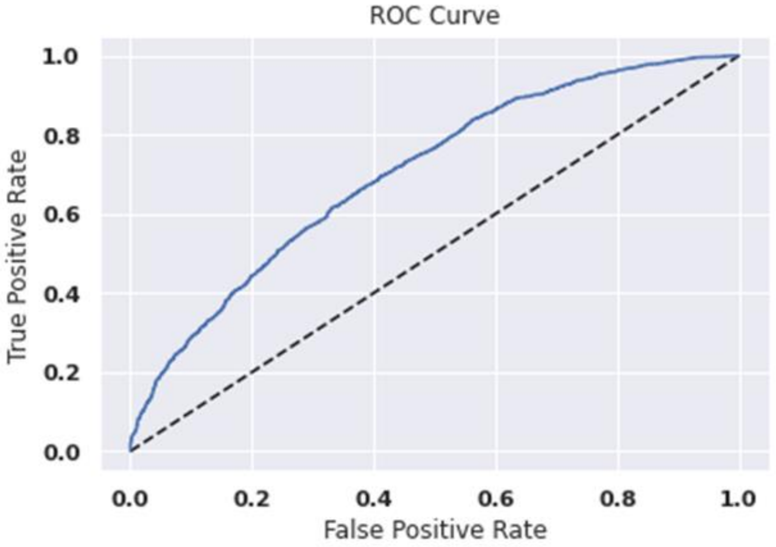


Model Evaluation Metric

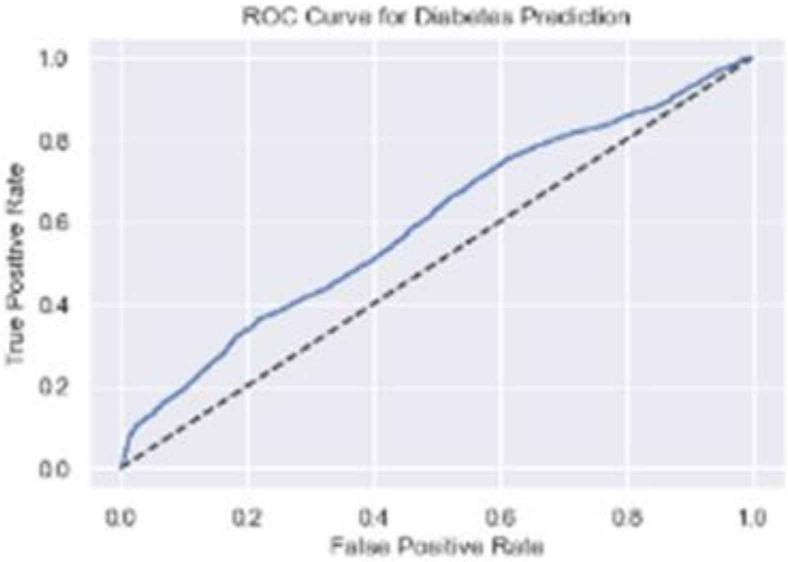
- AUC (Area under the curve) was used to evaluate model performance.
- This measures how well a model can distinguish between classes.
- We also used accuracy for the xgboost model, this measures how many classifications the model predicted correctly.



Xgboost

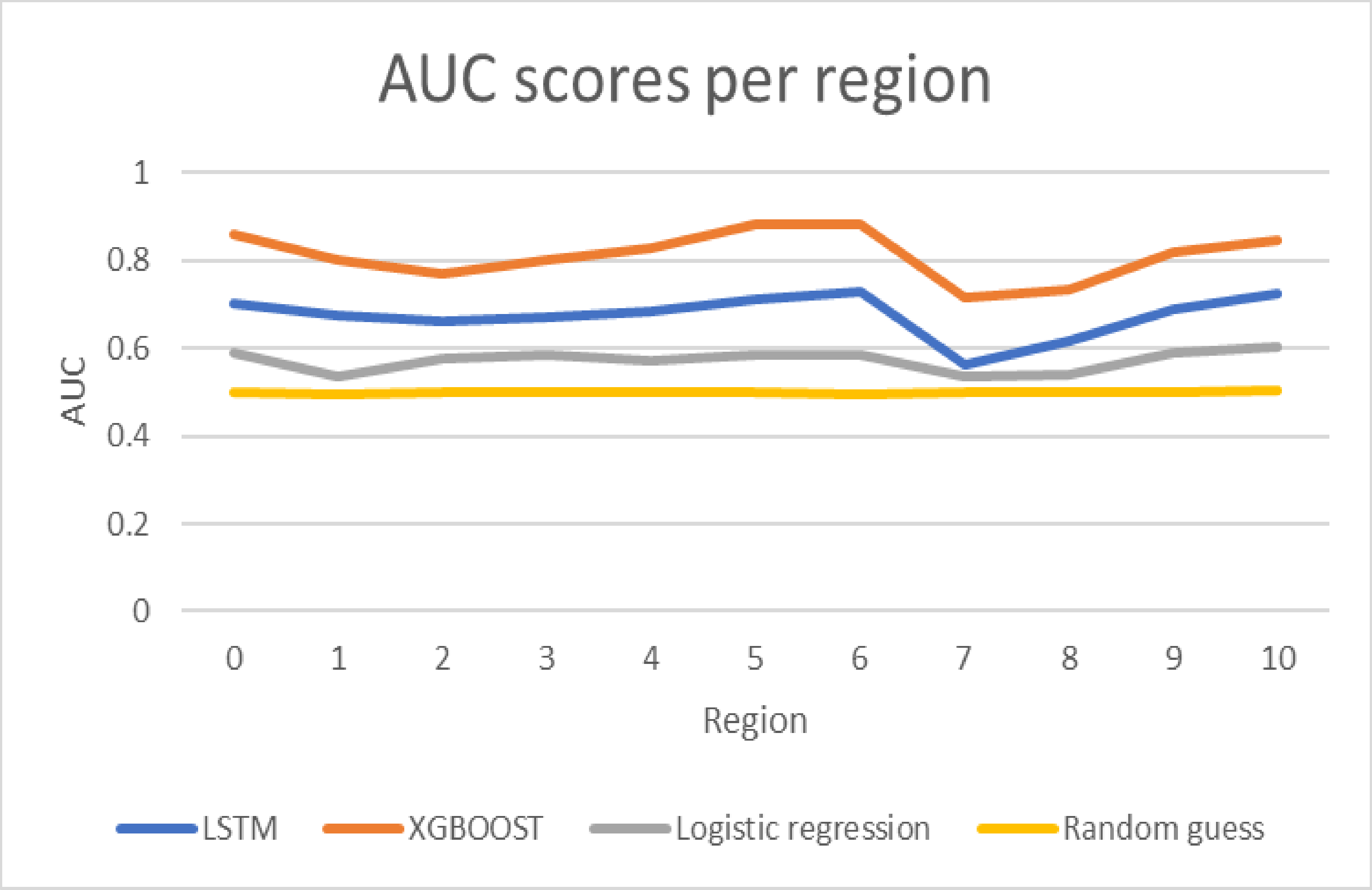


LSTM



Logistic regression

Model Evaluation



AUC	Category
0.9 – 1	excellent
0.8 – 0.9	good
0.7 – 0.8	fair
0.6– 0.7	poor
0.5-0.6	failed

- All models performed better than a random guess
- Xgboost model had the best performance
- LSTM was least sensitive to class imbalance and did not require any intervention. This is because the model relies on sequence and class imbalance techniques throw off the sequence.
- The timestep prediction problem with no aggregations (not shown here), the LSTM performed significantly better as it can handle sparse data much better than other models.
- Logistic regression is an over simplification of the problem as the data was not normally distributed.

Model Selection and Conclusion

- XGBoost provided better results.
- However LSTM was not optimized. It did not require data pre-processing and was more powerful for lower level predictions.
- The model selection will depend on what results the business finds more useful.
- Ex… is it better to know which location to be at during a specific hour of the day?
- What location to go to based on information of where the last pick up happened

		AUC		Accuracy	AUC	
Cluster	**%class imbalance**	**LSTM**	**XGBOOST**	**XGBOOST**	**Logistic regression**	**Random guess**
0	0.444583333	0.703071	0.860167	85.52%	0.590991	0.497117695
1	0.326329365	0.673788	0.801313	76.57%	0.53574	0.495104344
2	0.261884921	0.659683	0.768891	69.56%	0.577181	0.500756709
3	0.375992063	0.667801	0.800175	77.10%	0.584551	0.500672586
4	0.377083333	0.682989	0.827769	80.83%	0.571713	0.499889758
5	0.481686508	0.711069	0.879048	87.76%	0.584597	0.501157056
6	0.459861111	0.726677	0.879488	87.62%	0.584597	0.495433947
7	0.170972222	0.561313	0.716254	59.44%	0.532811	0.498483858
8	0.122757937	0.614024	0.733306	59.31%	0.541368	0.500282174
9	0.382539683	0.689411	0.818973	79.64%	0.590174	0.500778661
10	0.434305556	0.724347	0.845249	83.45%	0.603384	0.50307699

Part two: how much CO2 can we improve?

- Filtered the data to only get the trips with zero occupancy
- Used the python h3 API to estimate the distance travelled while waiting for a customer
- Calculated the total number of trips for the year (next month = current month*1.15)
- Estimated the total distance travelled calculating the sum of n samples (total number of empty trips for the that) sampled from an exponential distribution with lambda equal to the mean of the existing data set.
- The total CO2 emissions was calculates as (total distance travelled in miles * 404 grams (grams of CO2 emitted per mile)
- This gave us a value of **7646** grams.