

# ***Taxi Pickup Location Recommendation System***

Using MATLAB and  
Python

**Presented by:**

Saloni Shukla (42320902817)

Shwetank Dwivedi (41820902817)

Aakif Hussain Naqvi (41620902817)



# ***Objective***



- 1** Our recommendation system will serve as a helping hand for the ***small-scale taxi services***
- 2** We will be helping the companies to ***derive wiser decision using our product***
- 3** We will ***recommend the best pick-up locations*** within the cities using historical dataset.
- 4** Our product is ***cost-effective*** yet most promising to ***generate revenue***.

# ***Motivation***

## **Perspective 1**

**– Would you like to tell us the name of a company which provide taxi services around you?**

None of us is aware of taxi services in India except Uber and Ola.

–We, as the common people are underestimating the fact, that they are building the monopoly in the taxi service market.



## **Perspective 2**

In India and many more countries like India does have approx 70% of the towns and cities which does not have public transport services as smooth as the other metro cities within a country.

# *Lesson from Past*



Hailo was an Uber-like taxi booking app based in the UK founded in 2011.

In 2013, Hailo tried to enter the New York's taxi booking market with a massive funding of \$100M from investors but failed to generate profits.

**Why Failed?**

**Poor algorithm selection** for taxi fleet allocation in New York City by Hailo's Engineers

**Hailo thought that drivers in New York won't be any different than that of London.** Which ultimately turned out to be a horrendous assumption.

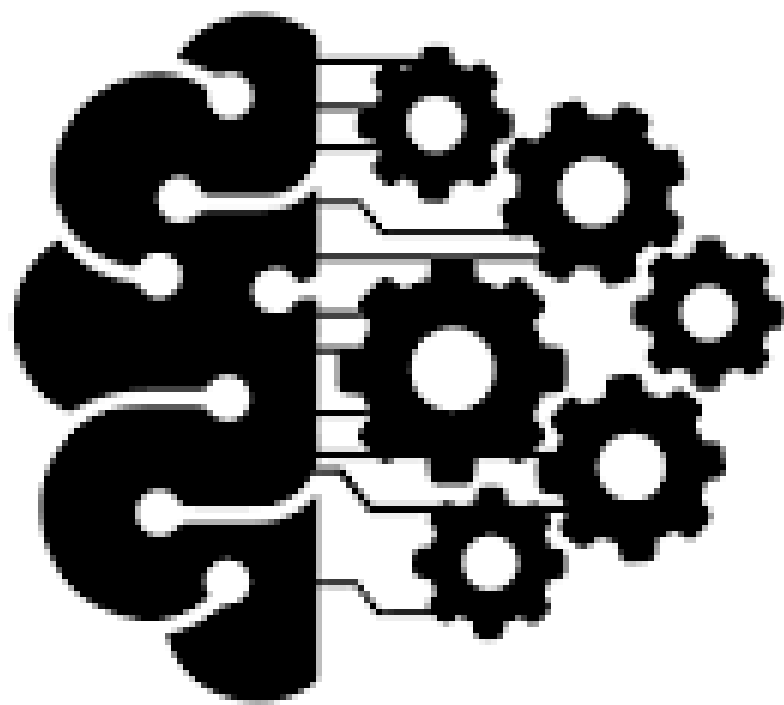




# **Working of the project**

# ***A Quick Talk About Streamlit***

Streamlit is an open-source Python library that makes it easy to build beautiful apps for machine learning. You can easily install it via `pip` in your terminal and then start writing your web app in Python.



# ***Why we used Streamlit only?***



*Streamlit is open-source, which makes it the best fit for our public domain portal. e.g. Best business decision at no cost.*

*Sit back and use our public domain free at home.*



START

***Now, we will walk you through  
the live demo of our application.***



## — What Now?

**Travelling Town is a concept based on the dataset of New York City only.**

**What will be the further steps?  
If the idea got an acceptance, and we as an individual will be allowed to collect the data then we can translate the same concept into reality for INDIA**



# ***Why NYC data only?***

1

Reliable and Trust Worthy Data, available on U.S govt. website

2

Comes with all the required attributes such as Location, Trip counts, latitude, longitude etc.

3

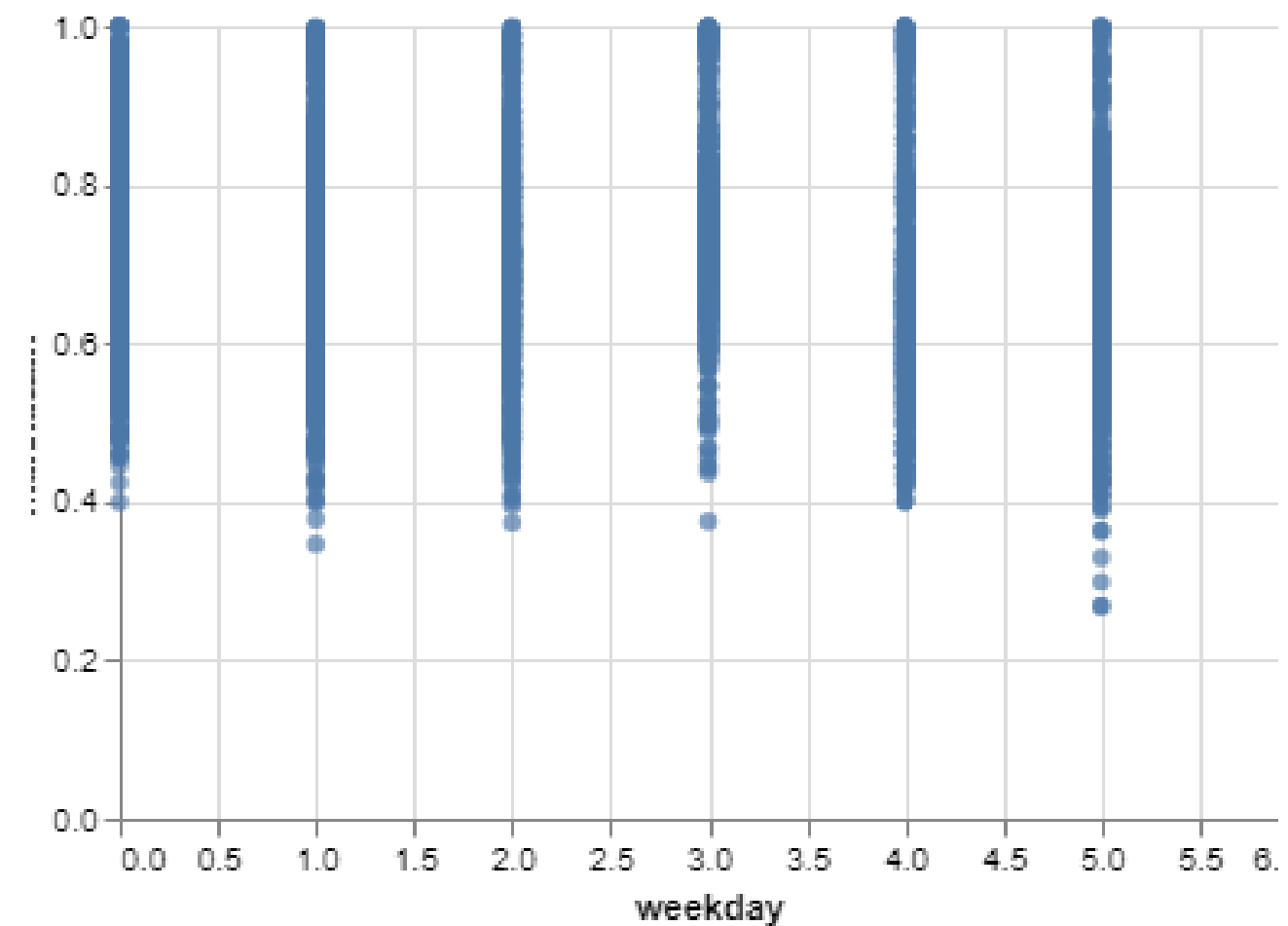
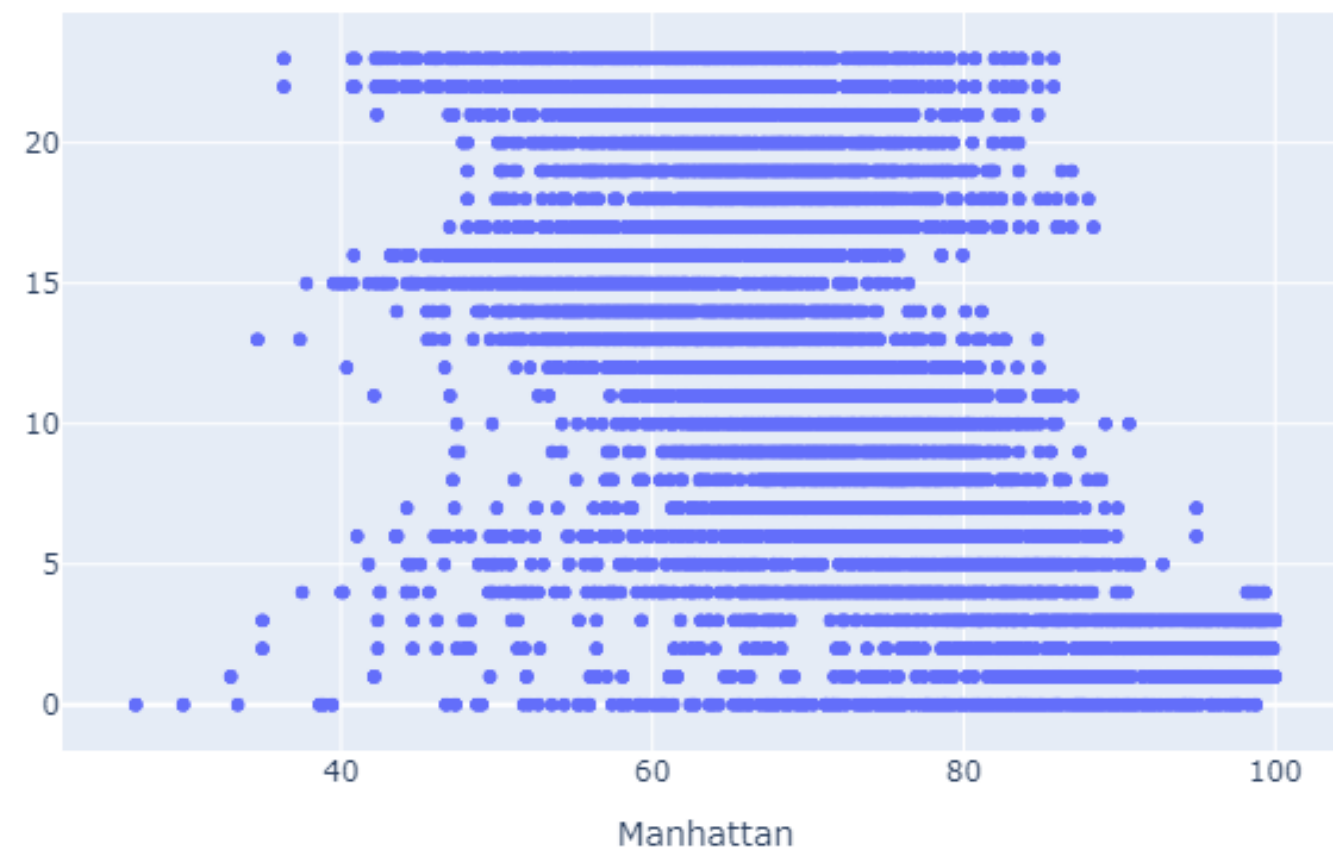
The Data set also contains the pickup time and drop off time which was very helpful in training the ML model to give the best possible results



[Click here for the dataset](#)

# ***Importance of Scatter & Altair plot***


*These plots primary uses are **to observe and show relationships between two numeric variables**. The dots in plots not only report the values of individual data points but also **patterns when the data are taken as a whole**.*



# ***Anaconda Prompt Command to start application***

Streamlit

localhost:8501



*Travelling town*  
A helping hand for taxi services

Anaconda Prompt (anaconda3) - streamlit run .\Web\_App.py

```
(base) C:\Users\Sanidhya>conda activate minor
(minor) C:\Users\Sanidhya>cd "C:\Users\Sanidhya\Downloads\Minor"
(minor) C:\Users\Sanidhya\Downloads\Minor>streamlit run .\Web_App.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.29.54:8501
```

# Raw Data

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Vendor	PickupTime	DropoffTime	Passenger	Distance	PickupLon	PickupLat	RateCode	HeldFlag	DropoffLo	DropoffLa	PayType	Fare	ExtraCharg	Tax	Tip	Tolls	ImpSurcha	TotalCharge	
2	15-01-2019 14:00	15-01-2019 14:13	1	3	-73.9643	40.77302	Standard	N	-73.9658	40.80467	Cash	12	0	0.5	0	0	0.3	12.8	
2	15-01-2019 14:00	15-01-2019 14:05	1	0.67	-73.9709	40.79592	Standard	N	-73.9702	40.78912	Credit card	5	0	0.5	1	0	0.3	6.8	
2	07-01-2019 14:58	07-01-2019 15:06	1	0.98	-73.9487	40.77778	Standard	N	-73.9553	40.78687	Credit card	7	0	0.5	1.4	0	0.3	9.2	
2	07-01-2019 14:58	07-01-2019 15:12	3	4.39	-73.9887	40.72271	Standard	N	-73.9872	40.69441	Cash	15.5	0	0.5	0	0	0.3	16.3	
1	20-01-2019 23:08	20-01-2019 23:23	1	3.9	-73.975	40.76028	Standard	N	-74.0085	40.71147	Credit card	15	0.5	0.5	3.25	0	0.3	19.55	
2	18-01-2019 19:49	18-01-2019 20:08	6	4	-73.971	40.78316	Standard	N	-73.9883	40.73729	Credit card	16	0	0.5	3.2	0	0.3	20	
2	01-01-2019 01:08	01-01-2019 01:35	1	5.78	-74.0078	40.71486	Standard	N	-73.9397	40.67459	Credit card	23	0.5	0.5	4.7	0	0.3	29	
2	01-01-2019 01:08	01-01-2019 01:14	4	0.88	-73.9642	40.76782	Standard	N	-73.9622	40.75893	Cash	6	0.5	0.5	0	0	0.3	7.3	
1	28-01-2019 10:50	28-01-2019 10:55	1	0.6	-73.9664	40.75332	Standard	N	-73.9677	40.75905	Cash	5.5	0	0.5	0	0	0.3	6.3	
1	23-01-2019 16:51	23-01-2019 17:50	1	9.3	-74.0067	40.71335	Standard	N	-73.9102	40.75486	Cash	39	1	0.5	0	0	0.3	40.8	
1	07-01-2019 20:09	07-01-2019 20:32	1	6.9	-73.9901	40.71411	Standard	N	-74.0042	40.65437	Credit card	23	0.5	0.5	4.85	0	0.3	29.15	
1	10-01-2019 19:37	10-01-2019 19:43	1	1	-73.9785	40.78333	Standard	N	-73.9822	40.77154	Credit card	6	0	0.5	1.36	0	0.3	8.16	
1	10-01-2019 19:37	10-01-2019 19:47	1	1.1	-74.0016	40.74668	Standard	N	-73.9839	40.7436	Credit card	8	0	0.5	1	0	0.3	9.8	
2	25-01-2019 17:45	25-01-2019 17:45	1	0	-73.9757	40.76132	JFK	N	0	0	Credit card	52	0	0.5	20	0	0.3	72.8	



# Processed Data

	A	B	C	D	E	F	G	H	I	
1	PickupTime	PickupLon	PickupLat	Location	TripCount	TimeOfDa	DayOfWee	DayOfMor	DayOfYear	
2	17-04-2019 15:00	-73.8746	40.77393	LaGuardia	2	0	Tuesday	1	1	
3	23-04-2019 19:00	-73.9863	40.75187	Manhattar	10	1	Tuesday	1	1	
4	10-04-2019 07:00	-73.9901	40.75663	Manhattar	0	1	Tuesday	1	1	
5	28-04-2019 23:00	-73.9899	40.75709	Manhattar	14	2	Tuesday	1	1	
6	08-04-2019 20:00	-73.8728	40.7742	LaGuardia	0	2	Tuesday	1	1	
7	29-04-2019 00:00	-73.7905	40.64397	JFK	0	2	Tuesday	1	1	
8	05-04-2019 10:00	-73.9941	40.75122	Manhattar	0	3	Tuesday	1	1	
9	10-04-2019 07:00	-73.9913	40.7503	Manhattar	10	4	Tuesday	1	1	
10	06-04-2019 15:00	-73.9873	40.75518	Manhattar	1	4	Tuesday	1	1	
11	23-04-2019 20:00	-73.9802	40.75085	Manhattar	0	4	Tuesday	1	1	
12	12-04-2019 12:00	-73.9919	40.74978	Manhattar	0	5	Tuesday	1	1	
13	06-04-2019 18:00	-73.873	40.77403	LaGuardia	1	5	Tuesday	1	1	
14	23-04-2019 21:00	-73.98	40.74873	Manhattar	5	6	Tuesday	1	1	
15	06-04-2019 00:00	-73.9902	40.75686	Manhattar	0	6	Tuesday	1	1	
16	10-04-2019 15:00	-73.7899	40.64692	JFK	2	6	Tuesday	1	1	

# ***About Our Model and its performance***

1

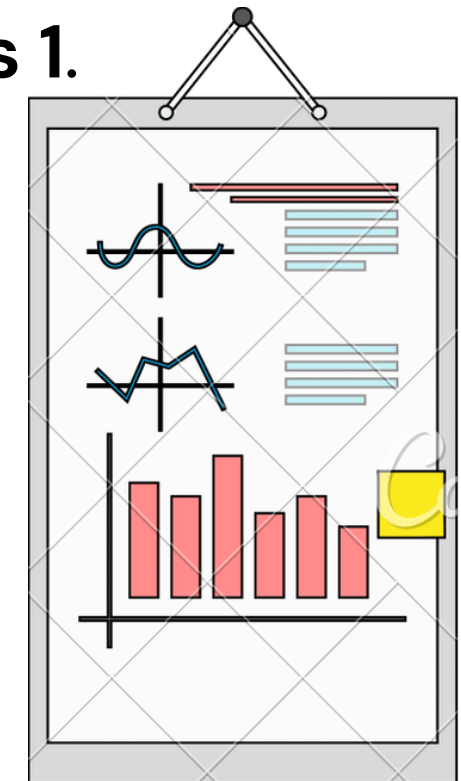
Our model is a subpart of the **regression tree** model named **Fine Tree**.

2

The performance was judged on the basis of Validation root mean square error which is **3.4456** of our trained model and the ideal **RMSE is 1**.

3

The model is not overfitting the dataset as the test data is also providing the same Validation RMSE as of training data..



# Training The Model

## Model Training

```
[modelStruct, validationRMSE] = trainRegressionModel(taxiPickupstrain)
```

```
modelStruct = struct with fields:
    predictFcn: @(x)treePredictFcn(predictorExtractionFcn(x))
    RequiredVariables: {'DayOfMonth' 'DayOfWeek' 'DayOfYear' 'Location' 'TimeOfDay'}
    RegressionTree: [1x1 RegressionTree]
    About: 'This struct is a trained model exported from Regression Learner R2020a.'
    HowToPredict: 'To make predictions on a new table, T, use: yfit = c.predictFcn(T) replacing 'c'
validationRMSE = 3.4592
```

```
myModel = modelStruct.RegressionTree
```

```
myModel =
    RegressionTree
        PredictorNames: {'Location' 'TimeOfDay' 'DayOfWeek' 'DayOfMonth' 'DayOfYear'}
        ResponseName: 'Y'
        CategoricalPredictors: [1 3]
        ResponseTransform: 'none'
        NumObservations: 20974
```

Properties, Methods

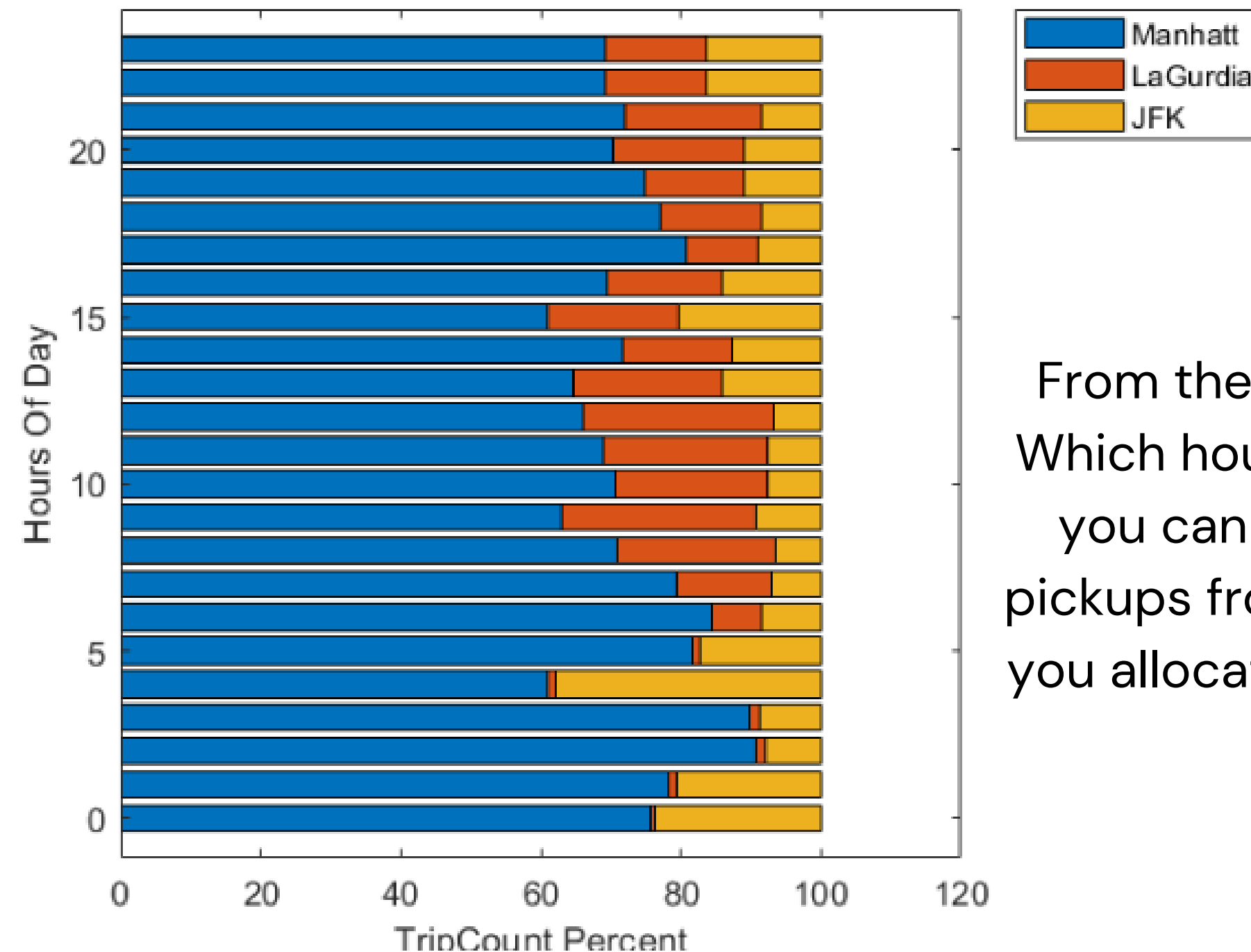
# *Testing The Model*

## Testing

```
taxiPickupstest = taxiPickups(test(Partition),:);  
taxiPickupstest = providedPreprocessing(taxiPickupstest);  
[modelStruct, validationRMSE] = trainRegressionModel(taxiPickupstest)
```

```
modelStruct = struct with fields:  
    predictFcn: @(x)treePredictFcn(predictorExtractionFcn(x))  
    RequiredVariables: {'DayOfMonth' 'DayOfWeek' 'DayOfYear' 'Location' 'TimeOfDay'}  
    RegressionTree: [1x1 RegressionTree]  
    About: 'This struct is a trained model exported from Regression Learner R2020a.'  
    HowToPredict: 'To make predictions on a new table, T, use: yfit = c.predictFcn(T)'.  
validationRMSE = 3.6615
```


# Conclusion



From the above stacked barplot, we can easily visualize which hour of the day is preferable for which location, as you can see most of the hours have a high percent of pickups from Manhattan, so it would be a great decision if you allocate most of your fleet for taxis in Manhattan, and rest taxis at other locations.



# Thank you



Have a great  
day ahead.