

Taxi Deployment

Table of Contents

Data Extraction and Introduction	1
Data Exploration and Partitioning	2
Visualization and Analysis.....	2
Separate Test Data	4
Models Training and Validation	4
Preprocessing	4
Model Training.....	4
Model Testing and Evaluation	4
Testing.....	5
Evaluation.....	5
Model Application, Results, and Analysis.....	6
Apply Model	6
Use the Results to Allocate Fleet.....	7

Data Extraction and Introduction

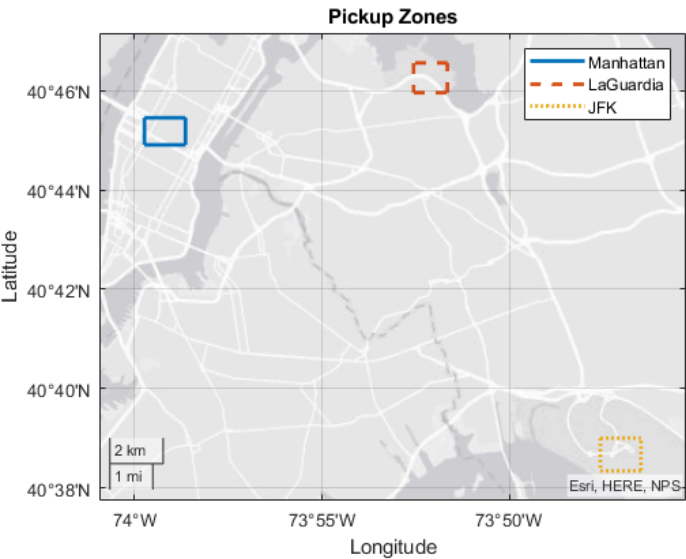
To get started, navigate to the Taxi Project folder, and run the script **generateTaxiPickupTable.mlx**.

This will create (and save) two tables: pickupLocations and taxiPickups. A preview and description of each table is given below.

pickupLocations = 3x5 table

	Names	Lat1	Lat2	Lon1	Lon2
1	"Manhattan"	40.7485	40.7576	-73.9955	-73.9773
2	"LaGuardia"	40.7660	40.7760	-73.8760	-73.8610
3	"JFK"	40.6390	40.6500	-73.7930	-73.7750

The zones are shown below for reference.



taxiPickups = 26226×3 table

	PickupTime	Location	TripCount
1	01-Jan-2015 00:00:00	Manhattan	22
2	01-Jan-2015 00:00:00	LaGuardia	2
3	01-Jan-2015 00:00:00	JFK	2
4	01-Jan-2015 01:00:00	Manhattan	10
5	01-Jan-2015 01:00:00	LaGuardia	0
6	01-Jan-2015 01:00:00	JFK	2
7	01-Jan-2015 02:00:00	Manhattan	14
8	01-Jan-2015 02:00:00	LaGuardia	0
9	01-Jan-2015 02:00:00	JFK	0

The table taxiPickups represents the number of pickups in your data over one hour intervals of 2015 in the zones defined in pickupLocations.

Data Exploration and Partitioning

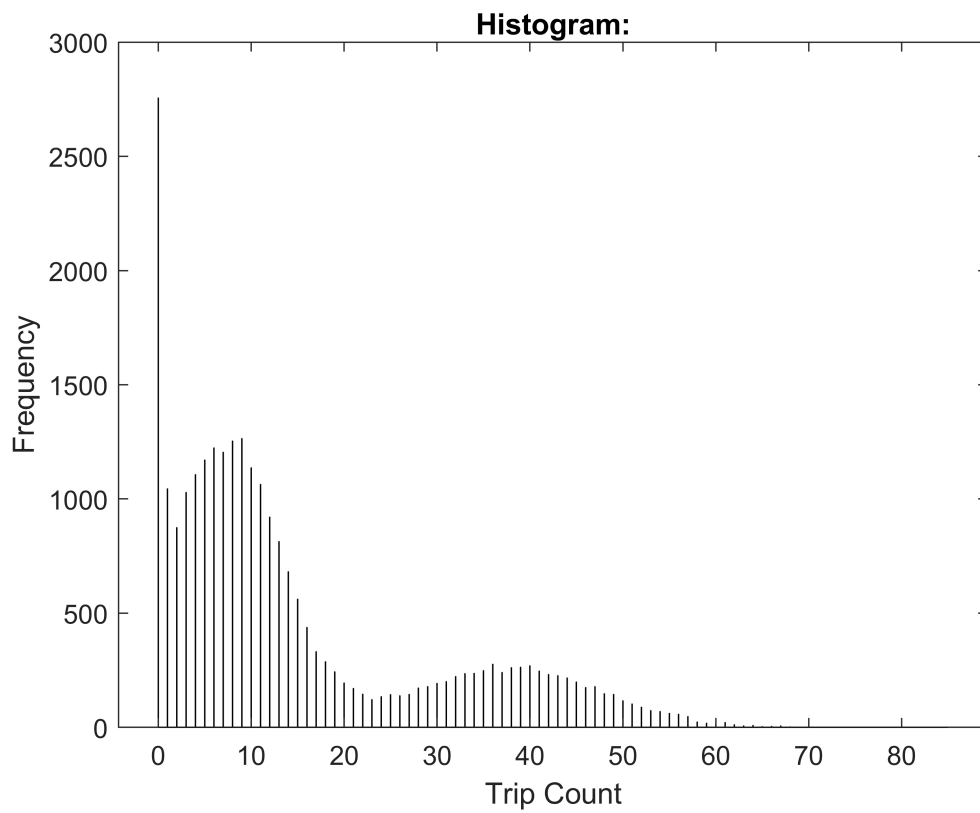
The script **generateTaxiPickupTable.mlx** in the previous section saved copies of the tables to a MAT file named **taxiPickupData.mat**. To avoid having to run the script again if you continue working after clearing your workspace, the code below loads the saved data.

```
% Make sure to follow the instructions in the previous section
if ~isempty(which('-all','taxiPickupData.mat'))
    load taxiPickupData.mat
else
    error("The file taxiPickupData.mat is not found on the MATLAB path. Add it to the path or r
end
```

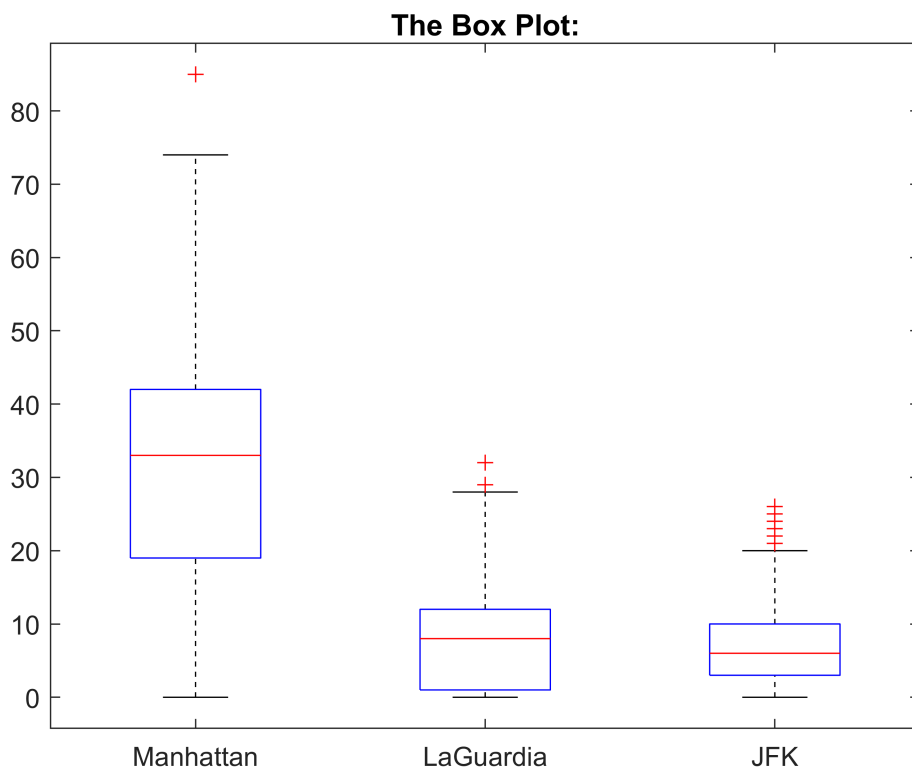
Visualization and Analysis

The code below will provide a visualization of the distribution (histogram) of the response variable TripCount, as well as a box plot for TripCount grouped by Location.

```
histogram(taxiPickups.TripCount,5000)
title('Histogram:')
xlabel('Trip Count')
ylabel('Frequency')
```



```
boxplot(taxiPickups.TripCount,taxiPickups.Location)  
title('The Box Plot:')
```



Separate Test Data

The following code will use `cvpartition` to separate 20% of the data set for testing later on, and create the training data. The results are repeatable because we are setting the random number generator seed to 10.

```
rng(10)
taxi_holdout = cvpartition(height(taxiPickups), "Holdout", 0.2);
trainingDataR = taxiPickups(training(taxi_holdout), :);
testDataR = taxiPickups(test(taxi_holdout), :);
```

Models Training and Validation

Preprocessing

The code below uses `providedPreprocessing.mlx` to add the following features to your training/validation data set:

- TimeOfDay (numerical)
- DayOfWeek (categorical)
- DayOfMonth (numerical)
- DayOfYear (numerical)

```
trainingDataR = providedPreprocessing(trainingDataR);
testDataR = providedPreprocessing(testDataR);
taxiPickups = providedPreprocessing(taxiPickups);
```

Model Training

The code below trains a model to predict `TripCount` using the processed test/validation data and **Report the validation approach and validation *RMSE* for the best model**. The goal will be to get a validation *RMSE* at or below 4.9. The script can reproduce the final model, including the model training.

```
%rng(10)
[trainedModel, validationRMSE] = trainRegressionModel(trainingDataR);
myModel = trainedModel.RegressionTree;
```

Validation *RMSE*:

```
validationRMSE
```

```
validationRMSE = 4.4813
```

The Validation *RMSE* is 4.4813

Validation Method: Holdout Validation.

Model Testing and Evaluation

Testing

the following code will report the metrics of the model in a table (the *RMSE* and R^2 are included).

```
testDataR = providedPreprocessing(testDataR);  
testOutput=predict(myModel,testDataR);  
testOutputMetrics=rMetrics(testOutput,testDataR.TripCount)
```

testOutputMetrics = 1×4 table

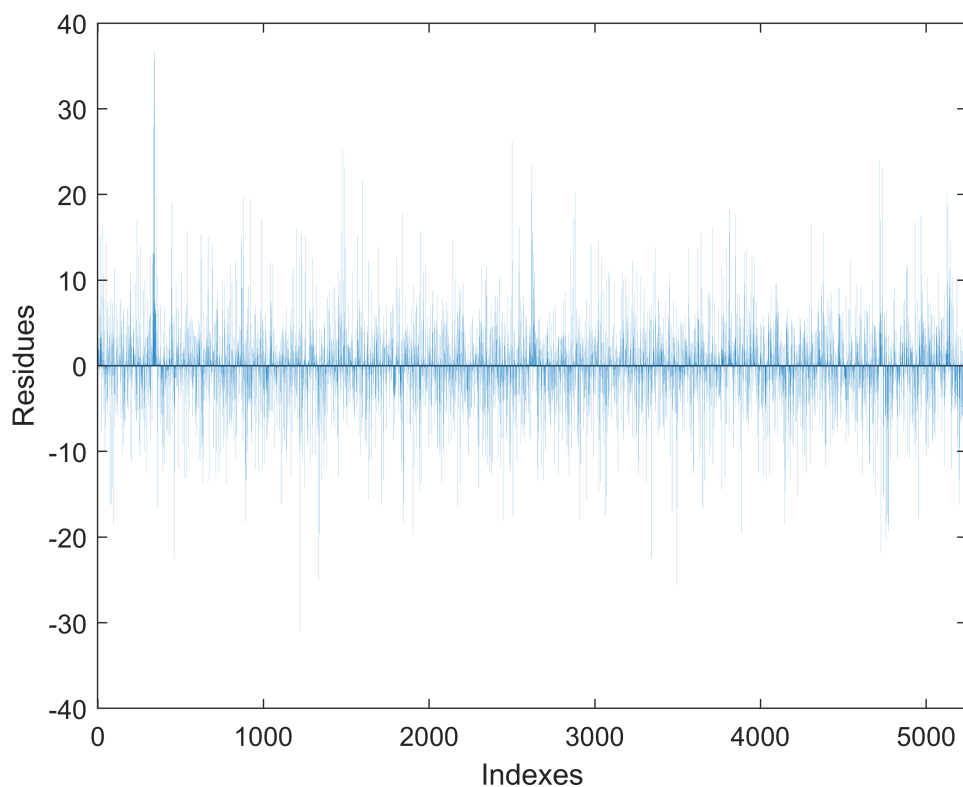
	mae	mse	rmse	Rsquared
1	3.4091	24.8395	4.9839	0.8758

RMSE is 4.9 and **Rsquared** is 0.8 as shown by the table.

Evaluation

This code will create a plot of the residuals:

```
myResiduals=(testOutput - testDataR.TripCount);  
bar(myResiduals);  
xlabel("Indexes");  
ylabel("Residuals");
```



The plot shows that the residuals can not go beyond the magnitude of 30, so we can be sure that the model is working correctly.

Model Application, Results, and Analysis

Apply Model

Using the code below, we create a new table of starting features for 2016.

```
taxiPickups2016 = table;  
taxiPickups2016.PickupTime = taxiPickups.PickupTime + years(1);  
taxiPickups2016.Location = taxiPickups.Location;  
taxiPickups2016 = providedPreprocessing(taxiPickups2016);  
% Display only the first 8 rows of the table  
head(taxiPickups2016);
```

I choose 2016-1-5 as my favorite day in 2016 and run the code so as to extract that day from the table.

```
myDay = datetime("2016-1-5");  
taxiPickupsMyDay = taxiPickups2016(day(taxiPickups2016.PickupTime, "dayofyear") == day(myDay, "dayofyear"));
```

Now, using my best model to predict TripCount on the day I've chosen, this will add it to the table.

```
taxiPickupsMyDay.TripCount = predict(myModel, taxiPickupsMyDay);  
head(taxiPickupsMyDay)
```

ans = 8x7 table

...

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
1	2016-01-05 ...	Manhattan	0.8200	Tuesday	5	5
2	2016-01-05 ...	LaGuardia	0.8200	Tuesday	5	5
3	2016-01-05 ...	JFK	0.8200	Tuesday	5	5
4	2016-01-05 ...	Manhattan	1.8200	Tuesday	5	5
5	2016-01-05 ...	LaGuardia	1.8200	Tuesday	5	5
6	2016-01-05 ...	JFK	1.8200	Tuesday	5	5
7	2016-01-05 ...	Manhattan	2.8200	Tuesday	5	5
8	2016-01-05 ...	LaGuardia	2.8200	Tuesday	5	5

The code below will use the model predictions to give the predicted fraction of trips happening in each hour on my selected day.

Uncomment below once you have defined the table taxiPickupsMyDay above.

```
taxiPickupsMyDayTotals = groupsummary(taxiPickupsMyDay, "PickupTime", "sum", "TripCount");  
taxiPickupsMyDay = join(taxiPickupsMyDay, taxiPickupsMyDayTotals, "RightVariables", "sum_TripCount");  
taxiPickupsMyDay.PickupFraction = taxiPickupsMyDay.TripCount ./ taxiPickupsMyDay.sum_TripCount;  
taxiPickupsMyDayFractions = unstack(taxiPickupsMyDay, "PickupFraction", "Location", "GroupingVariables");  
head(taxiPickupsMyDayFractions)
```

ans = 8x4 table

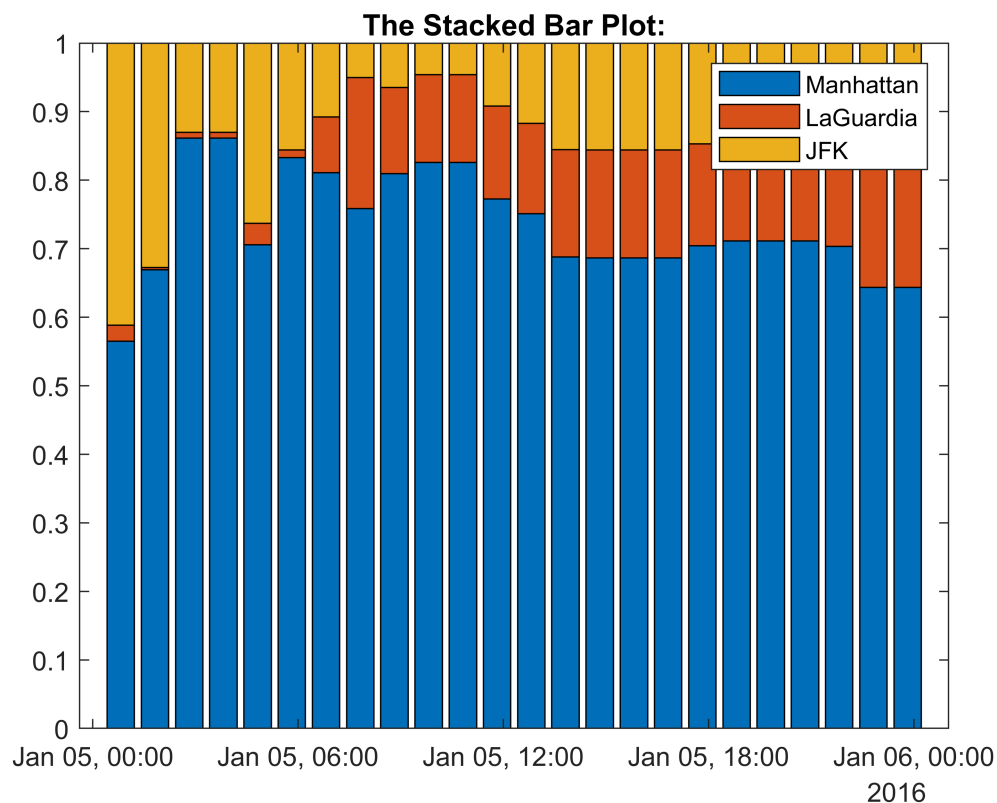
	PickupTime	Manhattan	LaGuardia	JFK
1	2016-01-05 ...	0.5650	0.0235	0.4116

	PickupTime	Manhattan	LaGuardia	JFK
2	2016-01-05 ...	0.6691	0.0034	0.3275
3	2016-01-05 ...	0.8614	0.0082	0.1304
4	2016-01-05 ...	0.8614	0.0082	0.1304
5	2016-01-05 ...	0.7055	0.0314	0.2631
6	2016-01-05 ...	0.8328	0.0115	0.1558
7	2016-01-05 ...	0.8109	0.0814	0.1077
8	2016-01-05 ...	0.7585	0.1914	0.0500

Use the Results to Allocate Fleet

To discuss the results we were able to obtain, and recommendations how to allocate the fleet of taxis on the chosen day. We can use a stacked bar plot generated by the code below:

```
xa=(taxiPickupsMyDayFractions.PickupTime)';
ya=[taxiPickupsMyDayFractions.Manhattan taxiPickupsMyDayFractions.LaGuardia taxiPickupsMyDayFractions.JFK];
bar(xa,ya,'stacked');
legend({'Manhattan','LaGuardia','JFK'})
title('The Stacked Bar Plot:')
```



The stacked bar plot shows that the greatest fraction of taxi pickups has to be dedicated to the Manhattan region. JFK is the second in taxi pickup demands and LaGuardia is the third. The allocation of taxis will have to

depend much on the hour of the day, because, for example, the Manhattan will require many taxis between 2 a.m and 4 a.m.