Taxi Deployment

Table of Contents

Data Extraction and Introduction 1
Data Exploration and Partitioning 3
  Visualization and Analysis 3
  Separate Test Data 7
Models Training and Validation 7
  Preprocessing 7
  Model Training 8
Model Testing and Evaluation 9
  Testing 9
  Evaluation 10
Residuals Plot 10
Model Application, Results, and Analysis 12
  Apply Model 12
  Use the Results to Allocate Fleet 15

Data Extraction and Introduction

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

Note it may take a while to finish.

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

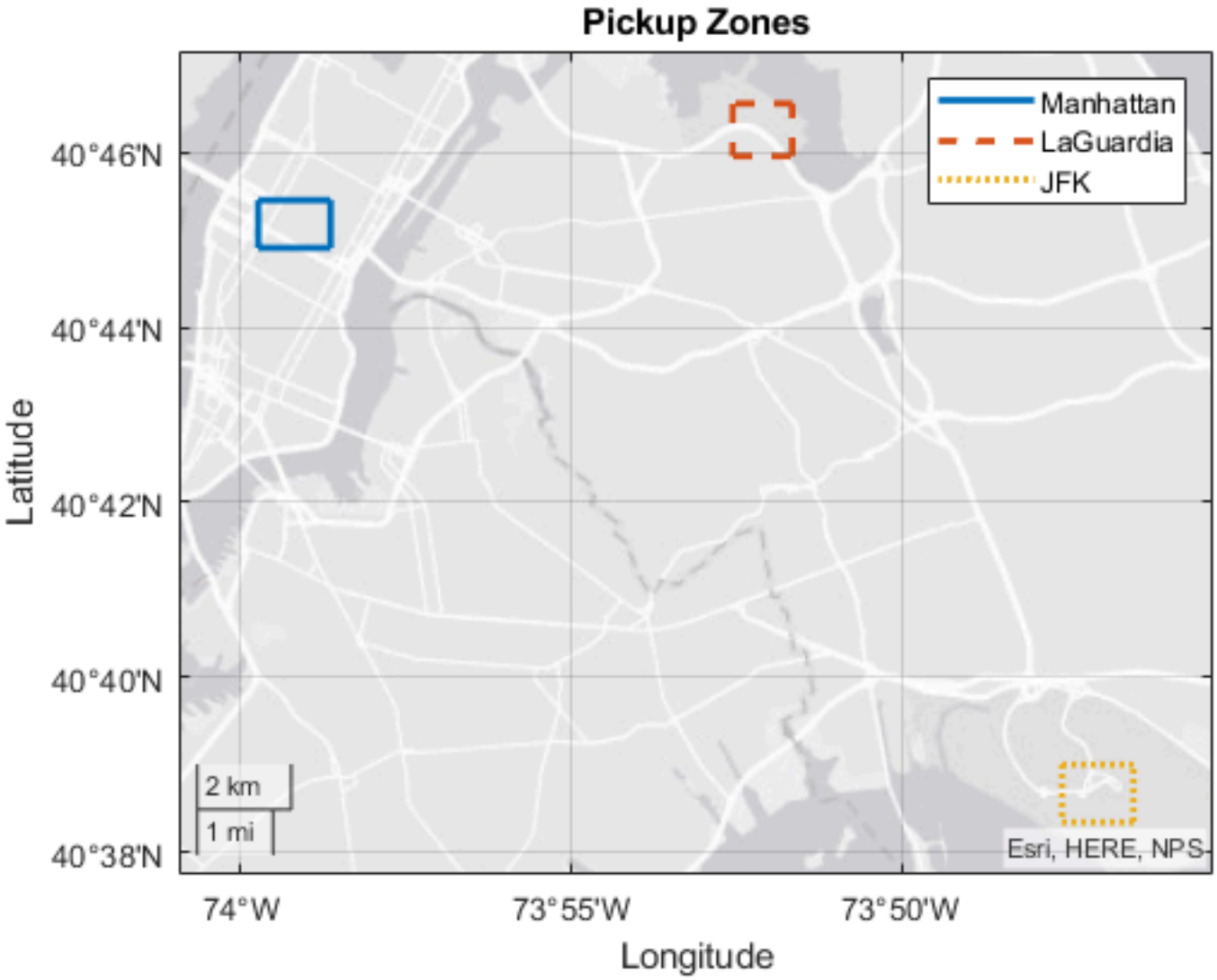
`pickupLocations = 3×5 table`

	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 table `pickupLocations` gives the latitude and logitude bounds for three pickup zones:

- **Manhattan**: here meaning an area of high taxi traffic surrounding Penn Station, Grand Central Station, and the Port Authority Bus Terminal
- **LaGuardia**: meaning an area surrounding LaGuardia airport
- **JFK**: similarly meaning an area surrounding JFK airport.

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 `taxiPickpus` represents the number of pickups in your data over one hour intervals of 2015 in the zones defined in `pickupLocations`. The start of each hour is specified in the `PickupTime` datetime variable, the zone is specified by the `Location` categorical variable, and the number of pickups is given by the `TripCount`.

Data Exploration and Partitioning

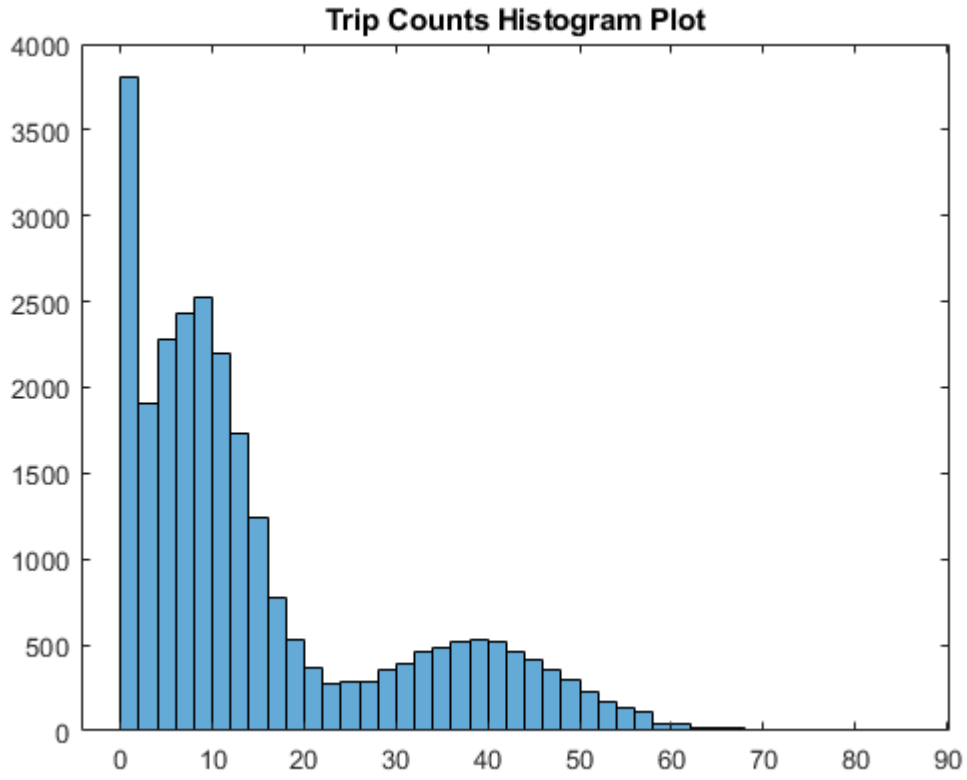
The script **generateTaxiPickupTable.mlx** in the previous section saved copies of the tables to a MAT file named **taxiPickupData.mat**. To avoid having to run the script again if you continue working after clearing your workspace, the code below loads the saved data.

```
% 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

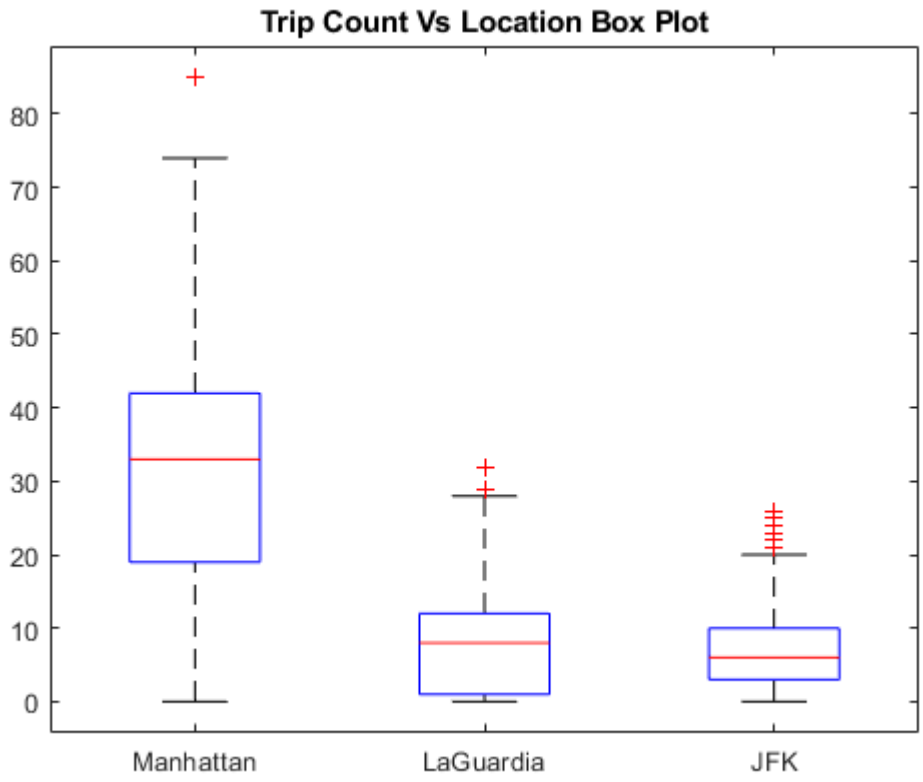
Analyze the data in the `taxiPickups` table. At a minimum, provide a visualization of the distribution (histogram) of the response variable `TripCount`, as well as a box plot for `TripCount` grouped by `Location`.

```
pickupLocations.Names=categorical(pickupLocations.Names);
histogram(taxiPickups.TripCount)
title('Trip Counts Histogram Plot')
```

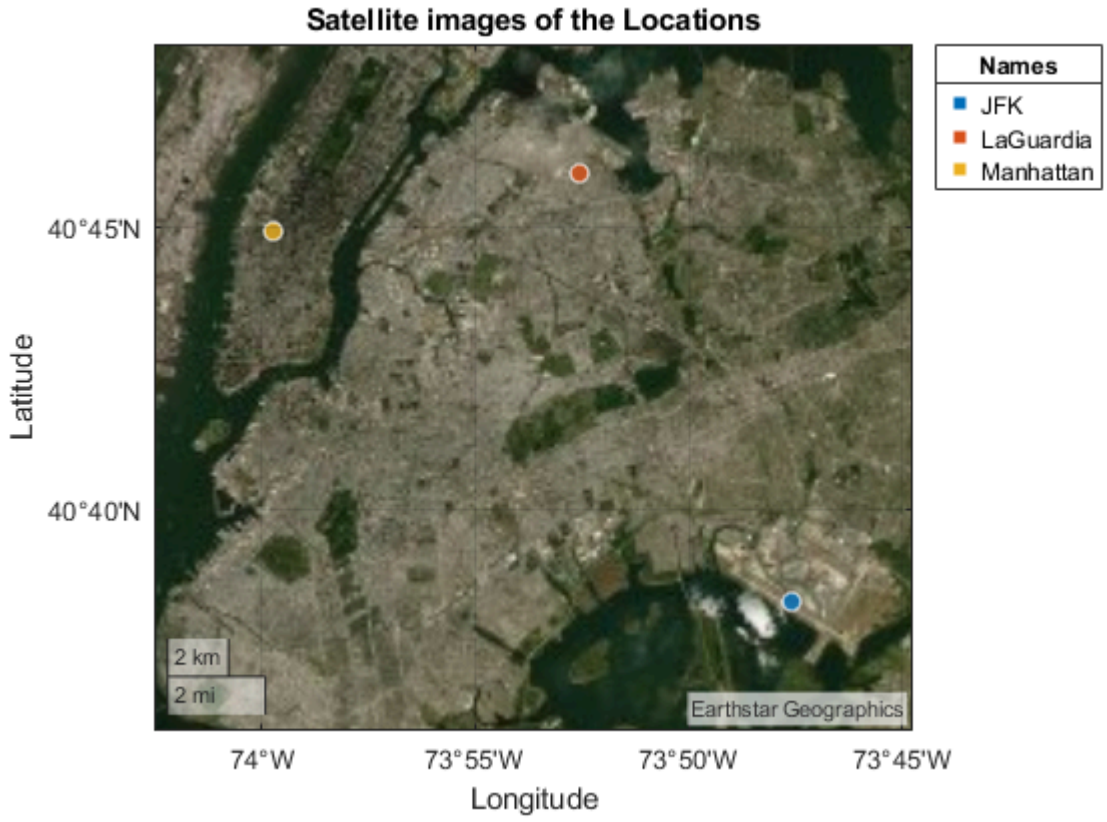



```
boxplot(taxiPickups.TripCount,taxiPickups.Location)

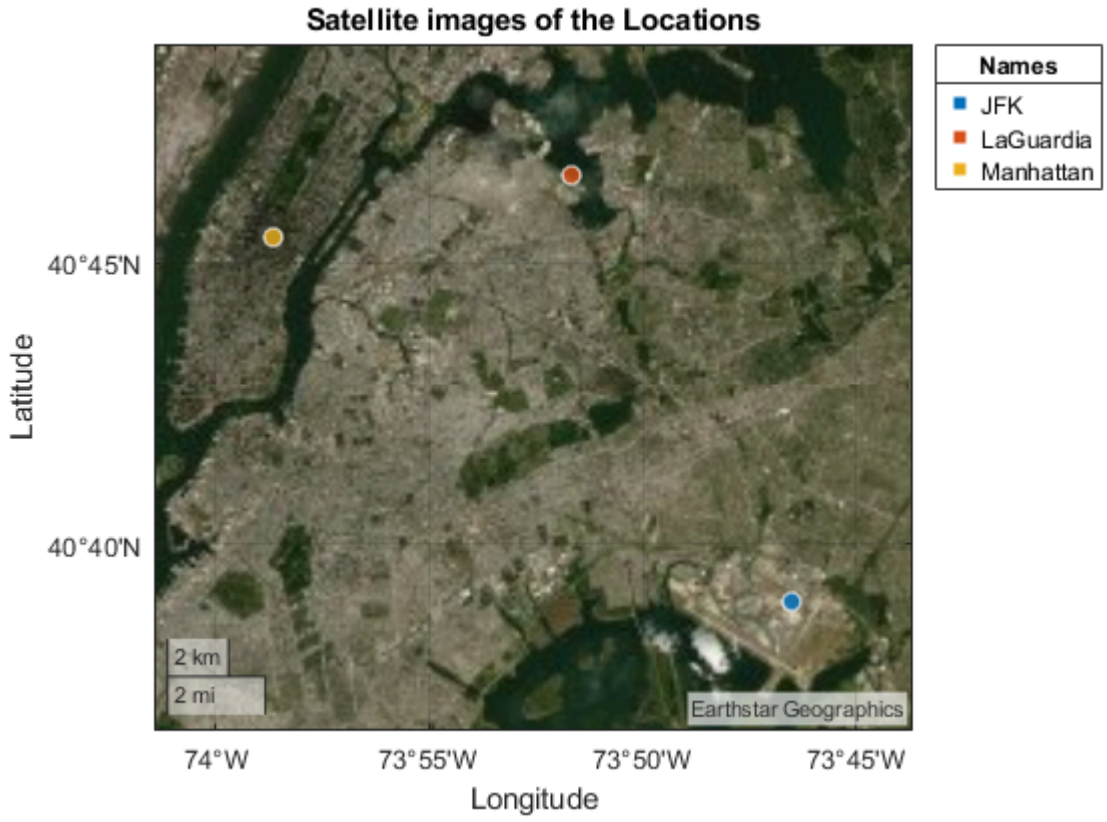
title('Trip Count Vs Location Box Plot')
```




```
geobubble(pickupLocations,"Lat1","Lon1","Basemap","satellite","ColorVariable","Names")
title('Satellite images of the Locations')
geolimits([40.613 40.792],[-74.042 -73.746])
```




```
geobubble(pickupLocations,"Lat2","Lon2","Basemap","satellite","ColorVariable","Names")
title('Satellite images of the Locations')
geolimits([40.624 40.802],[-74.024 -73.728])
```



Separate Test Data

Use `cvpartition` to separate 20% of the data set for testing later on, and create the training data. Ensure your results are repeatable by setting the random number generator seed to 10. Provide your code.

```
rng(10);
cv=cvpartition(height(taxiPickups),"Holdout",0.2)
```

```
cv =
Hold-out cross validation partition
   NumObservations: 26217
       NumTestSets: 1
         TrainSize: 20974
          TestSize: 5243
```

```
pickupstrain=taxiPickups(training(cv),:);
pickupstest=taxiPickups(test(cv),:);
```

Models Training and Validation

Preprocessing

As the focus of this course is machine learning, we've provided a function to do some feature engineering for you. Use **providedPreprocessing.mlx** to add the following features to your training/validation data set:

- `TimeOfDay` (numerical)

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

For example:

```
taxiPickupsTrain = providedPreprocessing(taxiPickupsTrain)
```

```
pickupstrain=providedPreprocessing(pickupstrain)
```

```
pickupstrain = 20974×7 table
```

...

	PickupTime	Location	TripCount	TimeOfDay	DayOfWeek	DayOfMonth
1	2015-01-01 ...	Manhattan	22	0	Thursday	1
2	2015-01-01 ...	LaGuardia	2	0	Thursday	1
3	2015-01-01 ...	JFK	2	0	Thursday	1
4	2015-01-01 ...	Manhattan	10	1	Thursday	1
5	2015-01-01 ...	JFK	2	1	Thursday	1
6	2015-01-01 ...	Manhattan	14	2	Thursday	1
7	2015-01-01 ...	LaGuardia	0	2	Thursday	1
8	2015-01-01 ...	JFK	0	2	Thursday	1
9	2015-01-01 ...	Manhattan	9	3	Thursday	1
10	2015-01-01 ...	LaGuardia	0	3	Thursday	1
11	2015-01-01 ...	Manhattan	10	4	Thursday	1
12	2015-01-01 ...	LaGuardia	1	4	Thursday	1
13	2015-01-01 ...	JFK	0	4	Thursday	1
14	2015-01-01 ...	JFK	1	5	Thursday	1

⋮

Model Training

Train models to predict `TripCount` using the processed test/validation data. **Report your validation approach and validation $RMSE$ for your best model**. Your goal will be to get a validation $RMSE$ at or below 4.9. Include code so that your script can reproduce your final model, including the model training. **If using the app, export the training function** and include a correct call to it in your script. Note you do not need to include the

generated training function code itself in your script, just a correct call to it. For example, if you'd trained a tree model in the app and exported the training function, you could include:

```
[modelStruct, validationRMSE] = trainRegressionModel(taxiPickupsTrain)
myModel = modelStruct.RegressionTree
```

```
[trainedModel, validationRMSE] = trainRegressionModel(pickupstrain)
```

```
trainedModel = struct with fields:
            predictFcn: @(x)ensemblePredictFcn(predictorExtractionFcn(x))
     RequiredVariables: {'DayOfMonth'  'DayOfWeek'  'DayOfYear'  'Location'  'TimeOfDay'}
    RegressionEnsemble: [1×1 classreg.learning.regr.RegressionBaggedEnsemble]
                 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' with th
validationRMSE = 4.5975
```

The RMSE is 4.5975 .Since this is a huge data I have used held out Validation as it takes less amount of Time and results are still good. I used 25% held out Validation. It can be further improved but it will take a lot of time.

Model Testing and Evaluation

Testing

Preprocess the test data as needed, and use it to test your best model. Provide your code and report at least the $RMSE$ and R^2. You will need to achieve a test $RMSE$ at or below 4.9 to receive full points here.

```
pickupstest=providedPreprocessing(pickupstest)
```

```
pickupstest = 5243×7 table
```

...

	PickupTime	Location	TripCount	TimeOfDay	DayOfWeek	DayOfMonth
1	2015-01-01 ...	LaGuardia	0	1	Thursday	1
2	2015-01-01 ...	JFK	0	3	Thursday	1
3	2015-01-01 ...	Manhattan	8	5	Thursday	1
4	2015-01-01 ...	LaGuardia	0	5	Thursday	1
5	2015-01-01 ...	JFK	2	6	Thursday	1
6	2015-01-01 ...	LaGuardia	0	7	Thursday	1
7	2015-01-01 ...	JFK	1	9	Thursday	1
8	2015-01-01 ...	Manhattan	15	12	Thursday	1
9	2015-01-01 ...	JFK	4	12	Thursday	1
10	2015-01-01 ...	LaGuardia	8	13	Thursday	1
11	2015-01-01 ...	JFK	5	16	Thursday	1
12	2015-01-01 ...	LaGuardia	8	18	Thursday	1
13	2015-01-01 ...	LaGuardia	1	19	Thursday	1

	PickupTime	Location	TripCount	TimeOfDay	DayOfWeek	DayOfMonth
14	2015-01-01 ...	JFK	1	19	Thursday	1

⋮

Evaluation

Discuss the results from training and testing. How well did your model generalize to new data? Include at least a plot of the residuals, and discuss your observations from the plot(s).

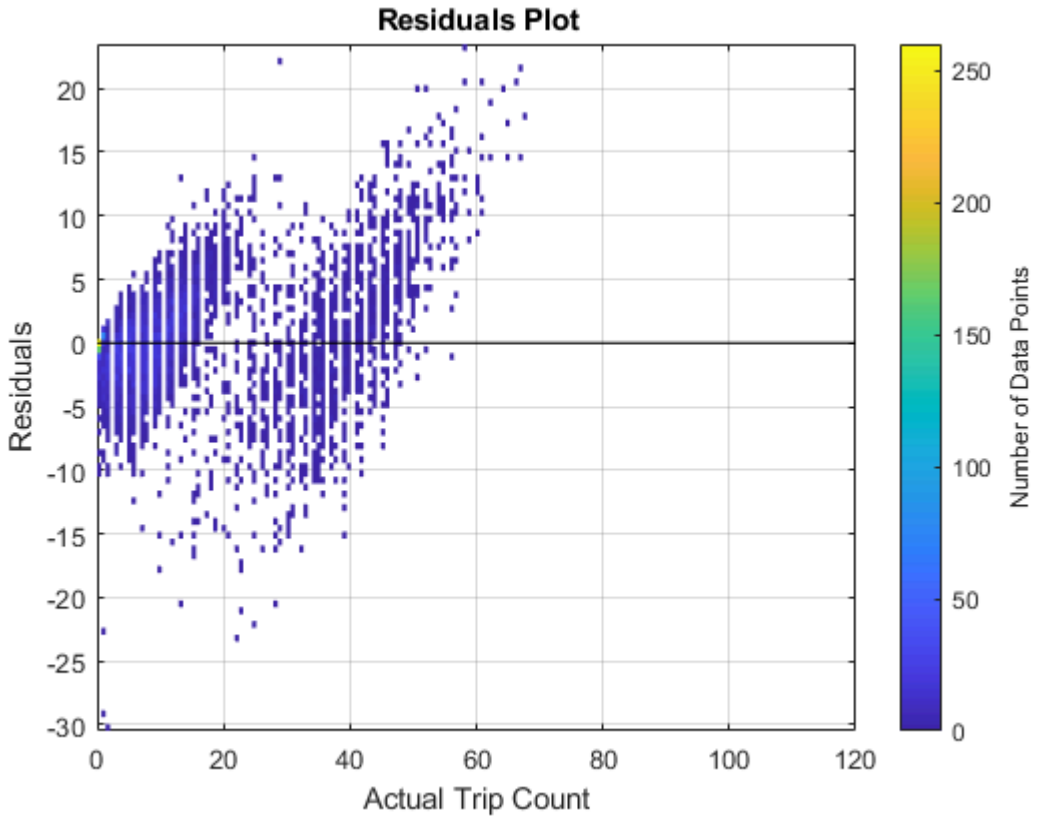
```
yActual=pickupstest.TripCount;
yPredict= trainedModel.predictFcn(pickupstest);
residuals= yActual-yPredict;
testMetrics=rMetrics(yActual,yPredict)
```

testMetrics = 1×4 table

	mae	mse	rmse	Rsq
1	3.2180	21.2131	4.6058	0.9053

Residuals Plot

```
histogram2(yActual,residuals,100,"DisplayStyle","tile")
line([0 120],[0 0],"Color","k")
xlabel("Actual Trip Count")
ylabel("Residuals")
title("Residuals Plot")
cb=colorbar();
cb.Label.String="Number of Data Points";
```



The tabe shows the value of residuals in comparison with the actual pickups number.

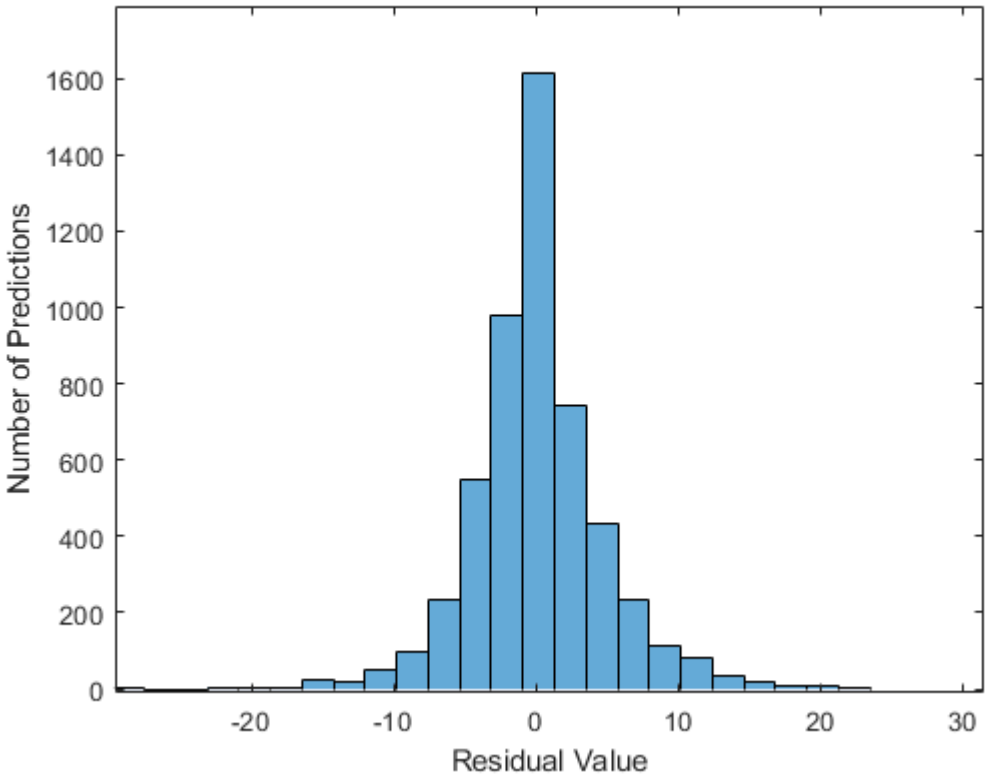
As We can see the Model performed good as it is not only lower than the required RMSE but is also very close to the train data RMSE

From the plots We can see its similar to the actual data. There are some **outliers which i think might be an error caused due to wrong entries.**

To demonstrate the accuracy of the trained model, the following histogram is plotted for residuals. As it can be seen, most of the unseen pickup numbers have near to zero residuals.

```
histogram(residuals,"NumBins",25)
xlabel("Residual Value")
ylabel("Number of Predictions")

xlim([-29.6 31.5])
ylim([-10 1790])
```



Model Application, Results, and Analysis

Apply Model

As the focus of this course is machine learning, we've provided some skeleton code in this section. Below, we create a new table of staring features for 2016.

Note that you will need to use the variable names provided in comments or make your own additional edits to the code.

```
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)
```

ans = 8×6 table

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
1	2016-01-01 ...	Manhattan	5.8200	Friday	1	1
2	2016-01-01 ...	LaGuardia	5.8200	Friday	1	1
3	2016-01-01 ...	JFK	5.8200	Friday	1	1
4	2016-01-01 ...	Manhattan	6.8200	Friday	1	1
5	2016-01-01 ...	LaGuardia	6.8200	Friday	1	1

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
6	2016-01-01 ...	JFK	6.8200	Friday	1	1
7	2016-01-01 ...	Manhattan	7.8200	Friday	1	1
8	2016-01-01 ...	LaGuardia	7.8200	Friday	1	1

Choose your favorite day in 2016 and edit the variable `myDay` below which will be used to extract that day from the table.

```
myDay = datetime("2016-09-11")
```

```
myDay = datetime
   11-Sep-2016
```

```
taxiPickupsMyDay = taxiPickups2016(day(taxiPickups2016.PickupTime,"dayofyear") == day(myDay,"da
```

Now, use your best model to predict `TripCount` on the day you've chosen and add it to the table.

```
 taxiPickupsMyDay.TripCount = trainedModel.predictFcn(taxiPickupsMyDay)
```

```
taxiPickupsMyDay = 72×7 table
```

...

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
1	2016-09-11 ...	Manhattan	0.8200	Sunday	11	255
2	2016-09-11 ...	LaGuardia	0.8200	Sunday	11	255
3	2016-09-11 ...	JFK	0.8200	Sunday	11	255
4	2016-09-11 ...	Manhattan	1.8200	Sunday	11	255
5	2016-09-11 ...	LaGuardia	1.8200	Sunday	11	255
6	2016-09-11 ...	JFK	1.8200	Sunday	11	255
7	2016-09-11 ...	Manhattan	2.8200	Sunday	11	255
8	2016-09-11 ...	LaGuardia	2.8200	Sunday	11	255
9	2016-09-11 ...	JFK	2.8200	Sunday	11	255
10	2016-09-11 ...	Manhattan	3.8200	Sunday	11	255
11	2016-09-11 ...	LaGuardia	3.8200	Sunday	11	255
12	2016-09-11 ...	JFK	3.8200	Sunday	11	255
13	2016-09-11 ...	Manhattan	4.8200	Sunday	11	255
14	2016-09-11 ...	LaGuardia	4.8200	Sunday	11	255

⋮

Again, to focus on machine learning, we have provided the necessary table manipulations and calculations below to use your model predictions to give the predicted fraction of trips happening in each hour on your selected day.

Uncomment below once you have defined the table `taxiPickupsMyDay` above.

```
taxiPickupsMyDayTotals = groupsummary(taxiPickupsMyDay,"PickupTime","sum","TripCount");
taxiPickupsMyDay = join(taxiPickupsMyDay,taxiPickupsMyDayTotals,"RightVariables","sum_TripCour
```

taxiPickupsMyDay = 72×8 table

...

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
1	2016-09-11 ...	Manhattan	0.8200	Sunday	11	255
2	2016-09-11 ...	LaGuardia	0.8200	Sunday	11	255
3	2016-09-11 ...	JFK	0.8200	Sunday	11	255
4	2016-09-11 ...	Manhattan	1.8200	Sunday	11	255
5	2016-09-11 ...	LaGuardia	1.8200	Sunday	11	255
6	2016-09-11 ...	JFK	1.8200	Sunday	11	255
7	2016-09-11 ...	Manhattan	2.8200	Sunday	11	255
8	2016-09-11 ...	LaGuardia	2.8200	Sunday	11	255
9	2016-09-11 ...	JFK	2.8200	Sunday	11	255
10	2016-09-11 ...	Manhattan	3.8200	Sunday	11	255
11	2016-09-11 ...	LaGuardia	3.8200	Sunday	11	255
12	2016-09-11 ...	JFK	3.8200	Sunday	11	255
13	2016-09-11 ...	Manhattan	4.8200	Sunday	11	255
14	2016-09-11 ...	LaGuardia	4.8200	Sunday	11	255

⋮

```
taxiPickupsMyDay.PickupFraction = taxiPickupsMyDay.TripCount./taxiPickupsMyDay.sum_TripCount
```

taxiPickupsMyDay = 72×9 table

...

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
1	2016-09-11 ...	Manhattan	0.8200	Sunday	11	255
2	2016-09-11 ...	LaGuardia	0.8200	Sunday	11	255
3	2016-09-11 ...	JFK	0.8200	Sunday	11	255
4	2016-09-11 ...	Manhattan	1.8200	Sunday	11	255
5	2016-09-11 ...	LaGuardia	1.8200	Sunday	11	255
6	2016-09-11 ...	JFK	1.8200	Sunday	11	255
7	2016-09-11 ...	Manhattan	2.8200	Sunday	11	255
8	2016-09-11 ...	LaGuardia	2.8200	Sunday	11	255
9	2016-09-11 ...	JFK	2.8200	Sunday	11	255
10	2016-09-11 ...	Manhattan	3.8200	Sunday	11	255
11	2016-09-11 ...	LaGuardia	3.8200	Sunday	11	255

	PickupTime	Location	TimeOfDay	DayOfWeek	DayOfMonth	DayOfYear
12	2016-09-11 ...	JFK	3.8200	Sunday	11	255
13	2016-09-11 ...	Manhattan	4.8200	Sunday	11	255
14	2016-09-11 ...	LaGuardia	4.8200	Sunday	11	255

⋮

```
taxiPickupsMyDayFractions = unstack(taxiPickupsMyDay,"PickupFraction","Location","GroupingVari
```

taxiPickupsMyDayFractions = 24×4 table

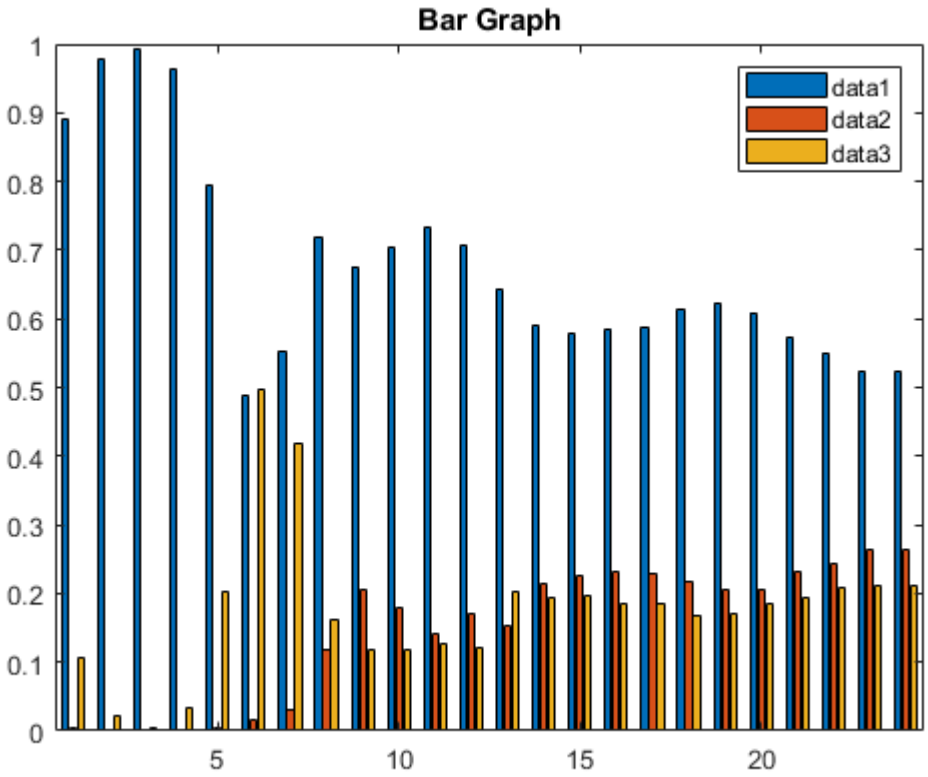
	PickupTime	Manhattan	LaGuardia	JFK
1	2016-09-11 00:49...	0.8896	0.0042	0.1063
2	2016-09-11 01:49...	0.9767	0.0009	0.0224
3	2016-09-11 02:49...	0.9936	0.0011	0.0053
4	2016-09-11 03:49...	0.9636	0.0027	0.0337
5	2016-09-11 04:49...	0.7933	0.0053	0.2013
6	2016-09-11 05:49...	0.4876	0.0160	0.4964
7	2016-09-11 06:49...	0.5523	0.0297	0.4180
8	2016-09-11 07:49...	0.7186	0.1192	0.1622
9	2016-09-11 08:49...	0.6761	0.2061	0.1178
10	2016-09-11 09:49...	0.7036	0.1786	0.1179
11	2016-09-11 10:49...	0.7324	0.1411	0.1265
12	2016-09-11 11:49...	0.7060	0.1720	0.1220
13	2016-09-11 12:49...	0.6443	0.1535	0.2022
14	2016-09-11 13:49...	0.5909	0.2146	0.1946

⋮

Use the Results to Allocate Fleet

Now it is time to present to Mr. Walker. Discuss the results you were able to obtain, and provide recommendations how you would allocate the fleet of taxis on the chosen day. Provide your reasoning, and also present your case using at least one visualization, e.g. a stacked bar plot.

```
display=bar([taxiPickupsMyDayFractions.Manhattan taxiPickupsMyDayFractions.LaGuardia taxiPickup
legend('show')
title('Bar Graph')
```



Good Morning ,

Mr Walker , here's the Prediction for the Date 11 September 2016.

Here **Data 1 is Manhattan , Data 2 is LaGuardia , Data 3 is JFk**

As You can clearly see from the Bar chart, Manhattan Area is having the most number of trips on 11th September , So I would strictly recommend you to **send the taxis to Manhattan.**

Usually the **pickups expected to be a little more in JFK than LaGuardia**,

however, during 9:00 AM to 2:00 PM it is the opposite.