Table of Contents

Objectives

- The main objective of this project is to create a model that predicts taxi demand around Manhattan and the airports region in New York City.
- The demand is divided into three categories: High, Medium and Low. The model will assign each region to one of the three categories on a hourly basis.
- One of the main focus of the model is to accurately distinguish the high and medium demand regions from the low demand regions which would enable the taxicabs to focus on high demand regions and avoid low demand ones, thus improving efficiency and increasing profits.
- To explore the revenue (Fare) and cost (Duration and Distance) of each region to see if certain regions of the city are more profitable than others.
- To analyze the model performance to see how much loss was generated from model's erroneous prediction.

The New York City is divided into 6 regions for this project. This 6 regions include:

- JFK Airport
- LaGuardia Airport
- Lower Manhattan (Manhattan Area)
- Midtown (Manhattan Area)
- Upper East Side (Manhattan Area)
- Upper West Side (Manhattan Area)

It can also be divided into even more sub-regions. The "Taxi Regions and Zones.csv" file contains full information regarding this.

We want to come up with a model which can predict the level of demand in each region and time periods of 1 hour through the whole year.

Data

1. The main data used for this project is taxi trip records of yellow taxicabs in New York City of the year 2015. The data is separated by month. A file datastore is created to aggregate all the records of 2015. A summary of the data is created to understand distribution of the feature variables.

- The data contains total 2922266 instances and 19 features.
- The features include pick-up and drop-off information (dates, times and locations), trip distances, fares, rate types, payment types, driver-reported passenger counts, tax provided, tips received and several other features for each trip.

2. Another dataset that is utilized is the dates of holidays throughout the year. This feature is added to see if it can be a predictor of demand level.

3. In addition to that, a dataset containing weather information in different regions of New York is used to extract Temperature and other information associated to each taxi trip.

4. Moreover, for the airport regions, flight data of the same period (time and number of arrival and departure flights) is utilized as a predictor of demand to taxis in the airport areas.

Reconstructing the Data

1. New features like pick-up and drop-off zones were added with the help of accessory function "addTaxiZones".

This adds four features to `taxiTable`: `PickupZone`, `tzPickupBorough`, `DropoffZone`, and `tzDropoffBorough`. These features indicate the zones and boroughs where trips started and ended. It uses the boundaries defined in the specified *shapefile* to determine the zones.

2. Then the several zones were grouped into a region. Total 6 such regions were created using the "Taxi Regions and Zones.csv" file. The code can be found in "**Converting zones to taxis"** section.

Data Cleaning

Real-world datasets are often messy and require significant cleaning. From the summary, it can be found that several cleaning steps were required. Some values are very unusual.

All the cleaning steps are assembled in this file Data_cleaning.mlx. The file also contains necessary figures to justify the cleaning steps.

Missing or Invalid Data

Some of the cleaning steps are -

- Some of pickup and drop-off locations were invalid
- The trip distance should be >0
- The minimum valid fare for taxi trips is 2.50
- The minimum number passengers is 1

Outliers

Additionally, the data contains outliers. And outliers can greatly affect model's performance. There are many ways to handle outliers. Percentile method was mostly used to handle outliers in this project. Any data that was not within 99.99% of the distribution was considered as an outlier.

Removing Other Regions

Our point of interest mainly lies on 6 regions: "Lower Manhattan", "Midtown", "Upper East Side", "Upper West Side", "JFK Airport", "LaGuardia Airport". All the other regions combined consists only about 10% of the overall data. So they were removed.

Group-wise Summary

- First, the pickup and drop-off time was grouped by hour. So, two new columns were created namely hourly pickup and drop-off.

- Then the taxi trips were grouped by region and hour both simultaneouly. It was done for pickup regions and drop-off regions separately. So these grouped data contains pickup or drop-off count, mean duration, distance, fare etc. The code file for grouping is attached in the appendix.
- Next, these grouped pickup and drop-off table were merged together using region and hourly data as key variables. This merged table contained several missing values. For example, some regions only have pickup at a region in a particular hour but no-drop off in that region in that hour. These missing values were filled with 0. The process can be found in data_joining.mlx file.

Feature Engineering

Data Splitting

Our processed dataset is splitted into train and test set. A 20% holdout validation was used to split the data.

Generating Response Variable

Our Modeling task is to predict if demand is “low,” “medium,” or “high,”; but these categories do not exist in the data. So we engineered a response variable named "demand" for machine learning. It is based on net pickups which is defined as pickup_counts - dropoff_counts.

- net_pickups < 0 : low
- 0<=net_pickups <15 : medium
- net_pickups >=15 : high

Summary Statistics

Before moving into feature selection and evaluation, it is important to have an understanding on the relation among features and distribution. Hence, some summary statistics were performed on data.

It was found that the distribution was unbalanced. The group count for the training data is as follows -

- High demand = 3250
- Medium demand = 19278
- Low demand = 17981

The test data also followed similar distribution.

Feature Creation

Some new features like day of the week, hour of the day, day of the month, day of the year were created based on the pickup and dropoff time.

Other features like is_holiday etc were created with help of external data.

The methods are compiled here.

Feature Selection

The response variable is based on the pickup and dropoff counts. Hence these features cannot be considered as a predictor variable. So, they were omitted.

Several feature selection methods were used to identify and evaluate important features. For example, correlation values, chi2 score etc were calculated. Heatmap, scatter plot etc were used as visualization technique to understand the relation among features.

Some featueres like tax, fare, no of passengers, day of the year etc were deemed as unimportant features.

While features like hour of day, day of week etc were identified as more important features.

It reuqires lots of exploration and visualization which are compiled in the file feature_selection.mlx.

Modeling

This is a classification problem with three response classes. In addition to model accuracy, further investigation was performed how the model performs for each response class.

Since machine learning is a highly iterative process, a lot of trial and error is required to develop an appropriate model.

During development many models were created and their performance were evaluated. Here only some of the selected few are mentioned.

For all cases, 5-fold cross validation was used.

Taxi Deployment Strategy

The following strategies were considered during model development for taxi deployment -

- Always go to the nearest High demand region when one is available
- Go to the nearest Medium demand region if there is no High demand region available
- Never go to or stay in a Low demand region

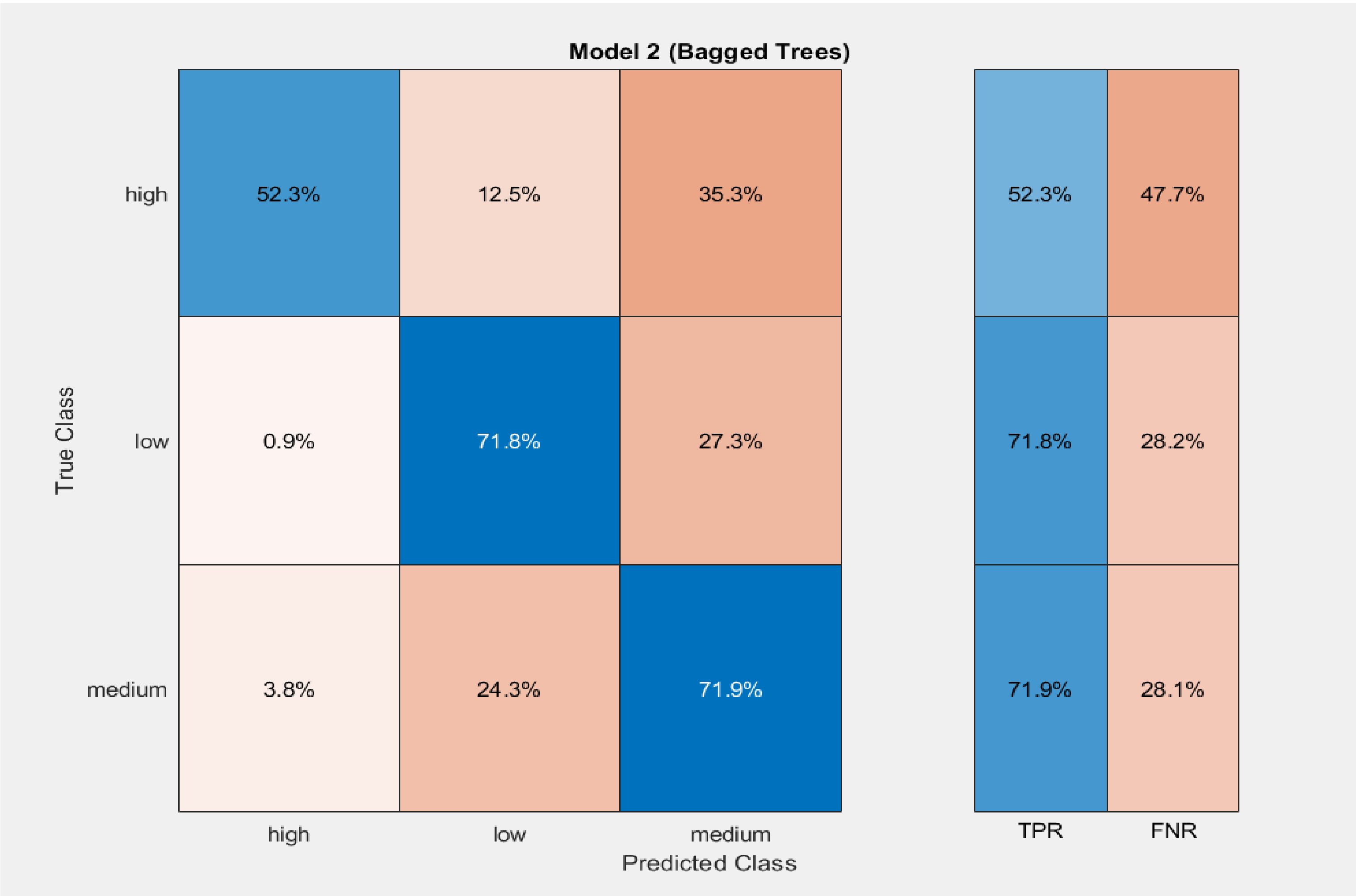
Based on this strategy, the model was developed.

Base Model (Model- 01)

At the initial stage, a base model was created without taking into consideration any other things. Total 8 features were used. It was found that Bagged ensemble model had the highest accuracy.

- Bagged ensemble = 70.7%
- SVM (linear) = 55.5%
- DT = 70%
- RUSBoosted tree = 63.7%

The confusion chart is attached here.



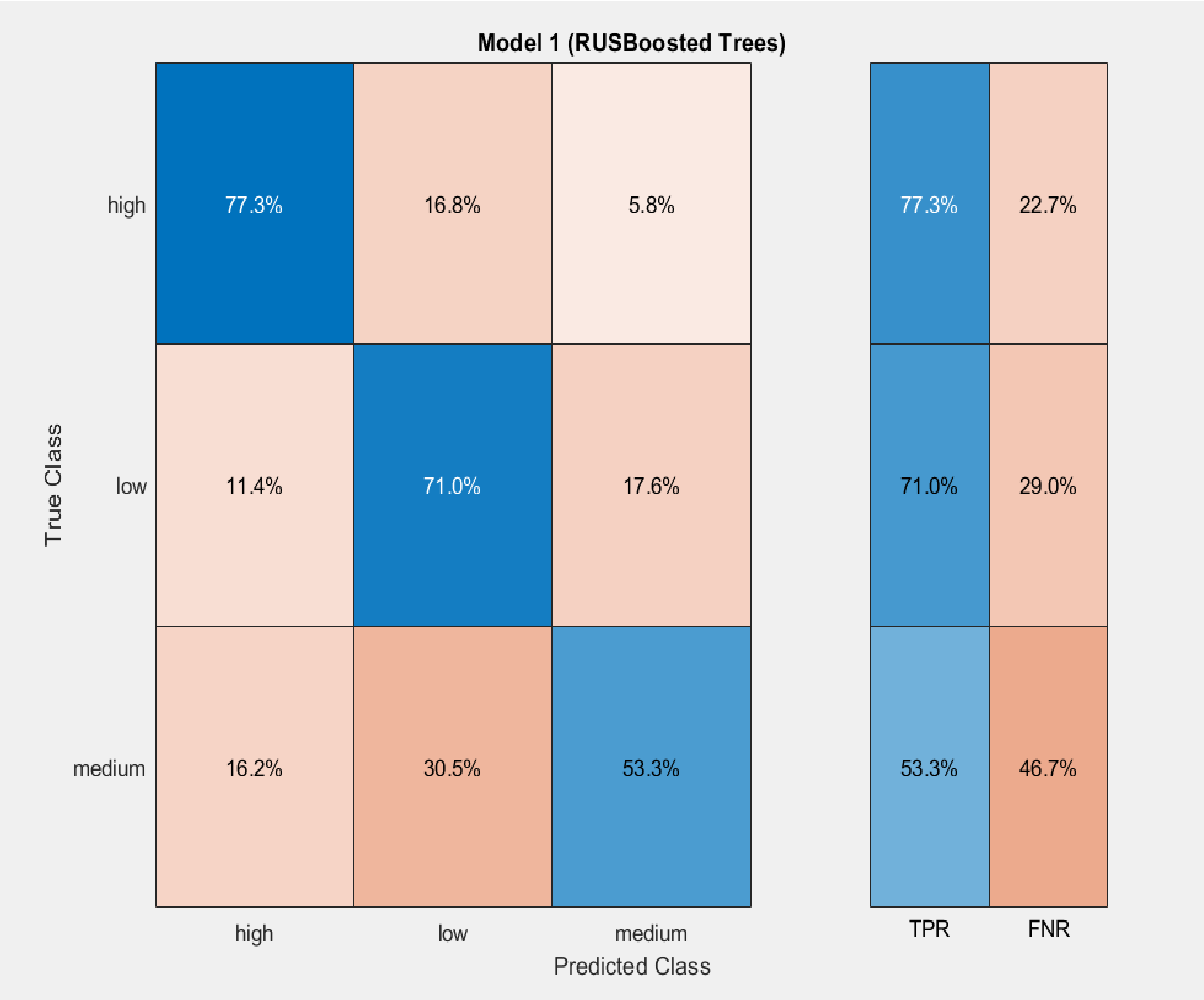
This model has fails to address two major issues - imbalance class and high FNR for 'high' class.

It puts equal emphasis on all classes.

In the next models that are developed, these issues were addressed.

Model - 02

To address the class imbalance issue, a RUSBoosted tree based model was developed. It performs undersampling of the majority class automatically. There was a drop in accuracy; 63.1% accuracy was achieved from this model but there was a significant drop in the FNR rate for 'high' demand class but slightly increased FNR for low demand class.



As it can be seen from the confusion matrix, FNR reduced to 22.7% for high demand class compared to 47.7% in Random forest model.

Model - 03

To further enhance the performance of our model - we tried different techiniques.

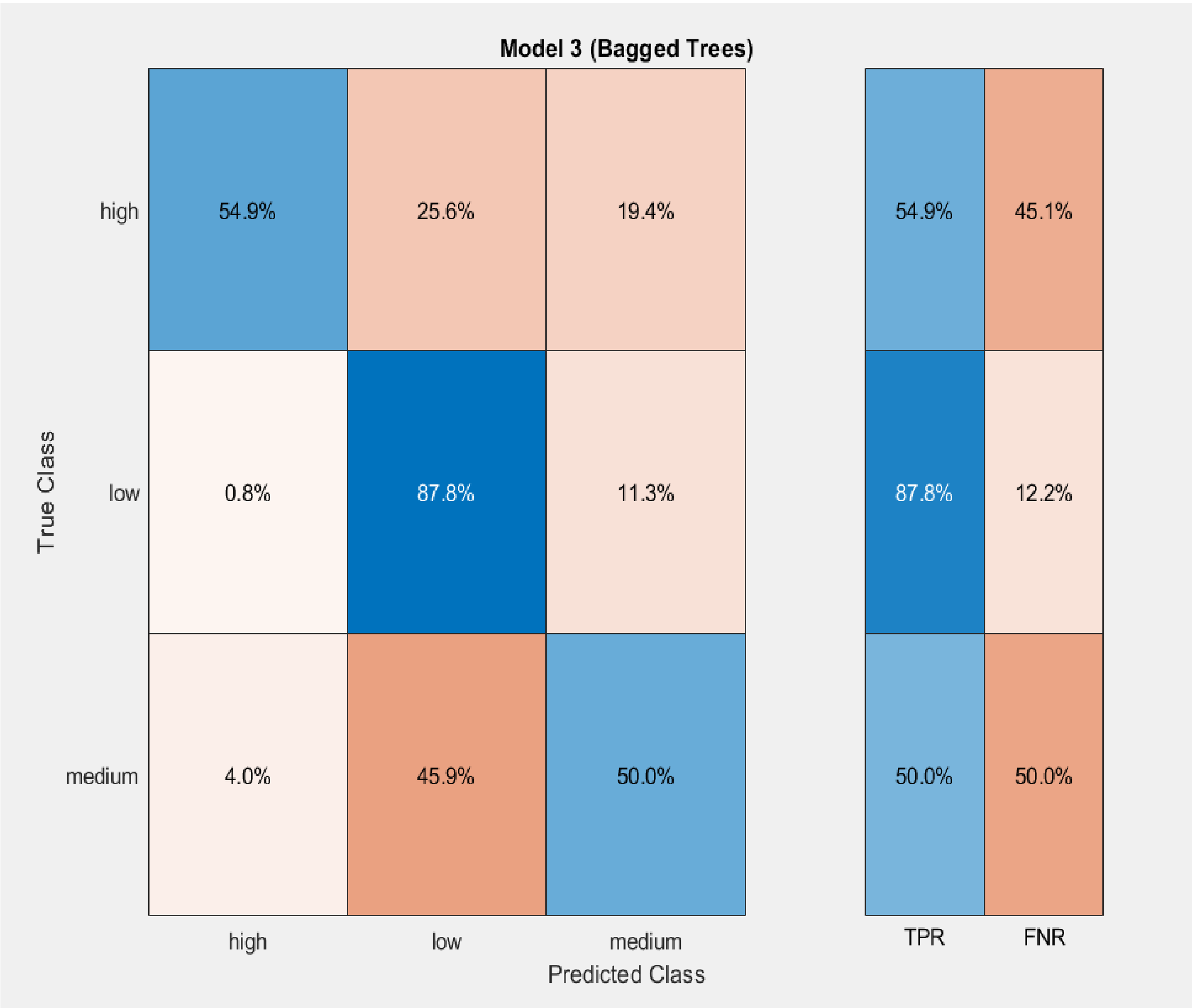
One important target of the model was to reduce false prediction of low demand class as high or medium demand class.

Also to reduce prediction of high demand regions as low demand regions.

In order to achieve that, a cost matrix was generated. Different combinations of costs were tried out. For example,

- low - high cost = 5
- low - medium cost = 3
- high - low cost =2
- others(non-diagonal) = 1

The output from bagged model is shown below -



The accuracy of the model is about 67.3% but it has an FNR of 12.2% for low class (0.8% for low->high); which is a significant improvement in the performance. Previously FNR was around 28% for low class.

Final Model (Model - 04)

For further improvement, oversampling was performed on the high demand class since it was the minority class.

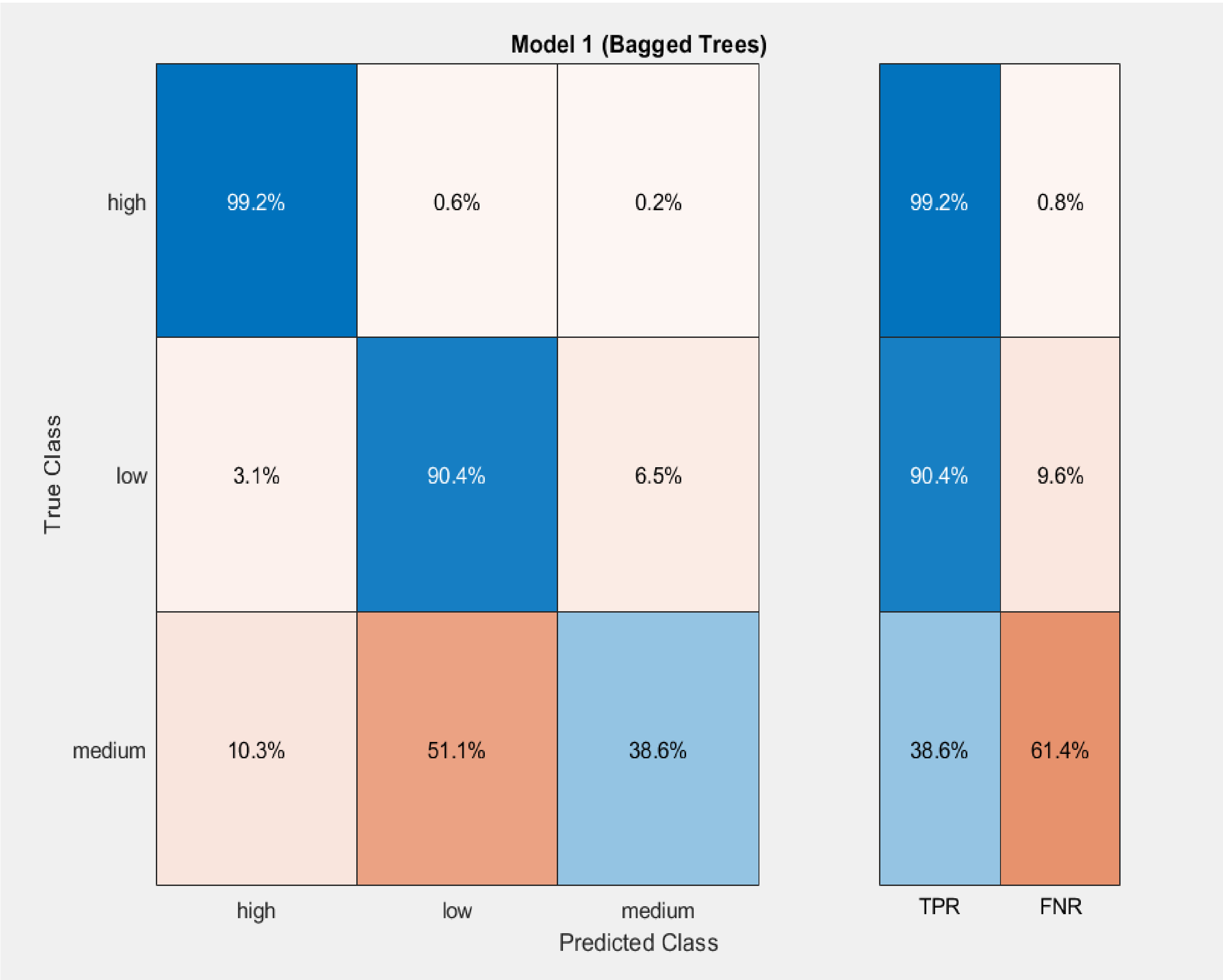
There were around 3000 high demand instances while there were around 18000 low and medium demand instances each. So, the high demand instances were oversampled to 12000 instances.The code is provided here.

It was first tested separately. Then it was combined with a cost matrix.

After several trial and errors, we've found the following cost provides a better result on all fronts.

- low - high cost = 10
- low - medium cost = 6
- high - low cost =4
- others(non-diagonal) = 2

The accuracy of the bagged model was found 73.3% and confusion matrix -

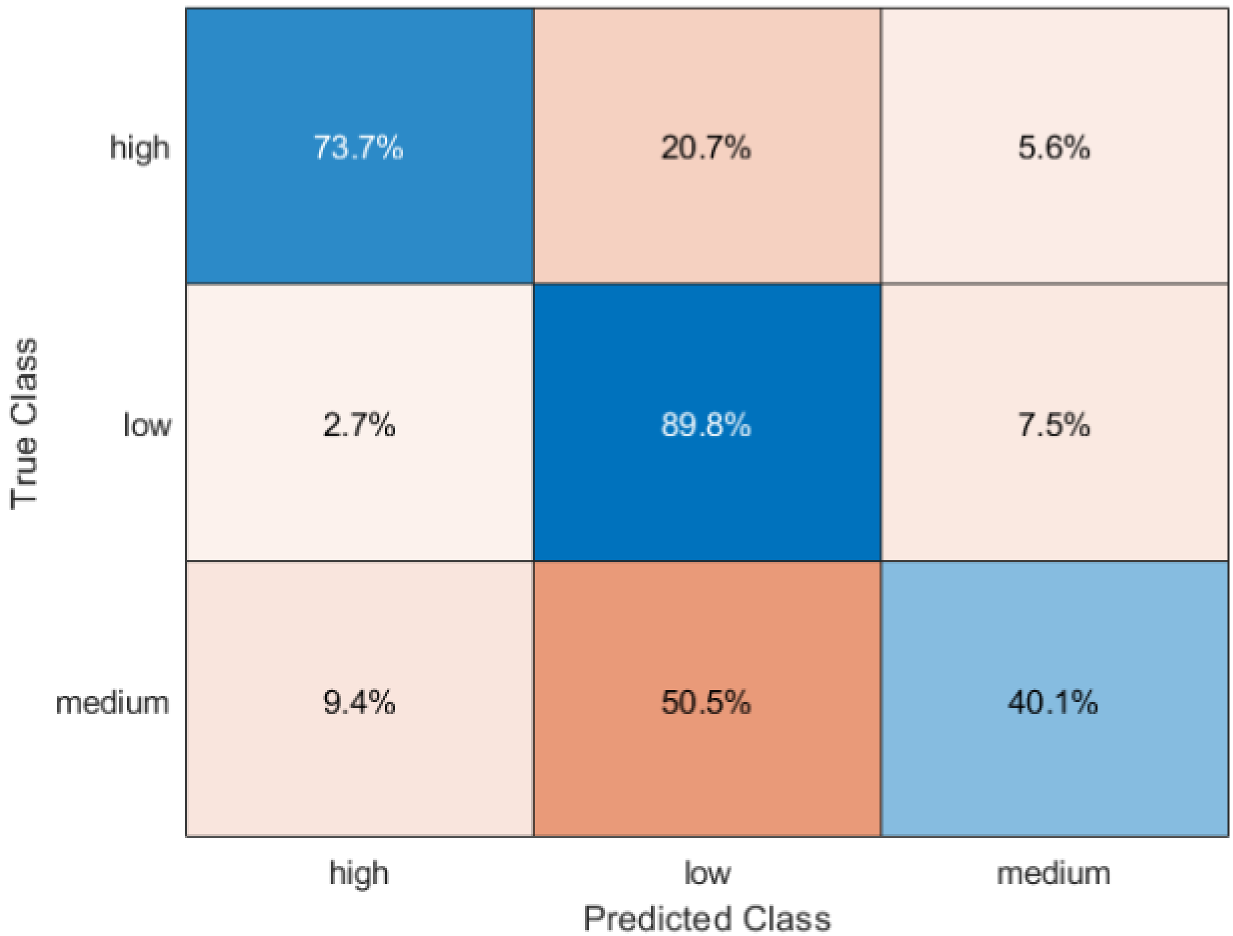


The FNR for low class is at 9.6% while FNR of high class is at 0.8% only. Which is a very significant improvement in performance.

Performance on Test Data

The model was trained using classification learner tool. It was exported and then tested on the test data which separated out at the very beginning.

The Model achieves an accuracy of 64.4% on the test set which is a bit lower than the train accuracy. But it is expected since we oversampled our trained model.



If we look at the FNR, we can see it is at 10.2% for the low class - very close to train data performance.

But there was drop in model's performance for high class.

Since it was the main focus of the model to minimize the number of falsely prediction of low class as high or medium class; looking at the test performance, it is clear that the model achieves it.

The relevant codes are provided here.

The function for generating the model can be found here - Classifier_function.

Analysis

Analyzing the modeling scenario

The taxi deployment strategy can be analyzed as follows -

Based on the strategy above, to avoid having taxis end up in a Low demand region, it is critical to avoid classifying a region as High demand or Medium demand when it is Low demand. Therefore, emphasize was given reducing the false negatives for Low, with extra emphasis on Low demand regions misclassified as High demand.

Falsely identifying High demand or Medium demand regions as Low demand is not as much of an issue, since assuming the actual Low demand regions are correctly identified, taxis will still deploy into another Medium or High demand region. Therefore, if it helps reduce false negatives for Low demand, some increase to the number of false positives for Low demand is an acceptable trade-off. However, High demand is relatively rare, so the misclassification of High demand as Low demand was tried to be kept at minimum.

Mixing up High and Medium demand is less of an issue. A surprise High demand is not a bad outcome, and a surprise Medium demand is not the end of the world.

Train - Test Performance analysis

As it was pointed out in the last passage that keeping False negatives for low class especially low misclassified as High demand was the first priority, looking at our model's performance for that scenario -

- FNR (low misclassified as High) = 3.1% (training data)
- FNR (low misclassified as High) = 2.7% (test data)
- FNR (low misclassified as Medium) = 6.5% (training data)
- FNR (low misclassified as Medium) = 7.5% (test data)

FNR is lower for testing than training; So, it is clear that the model is generalized well enough to ensure that FNR is kept at minimum for that scenario.

The second target was not to misclassify too many high demand instances.

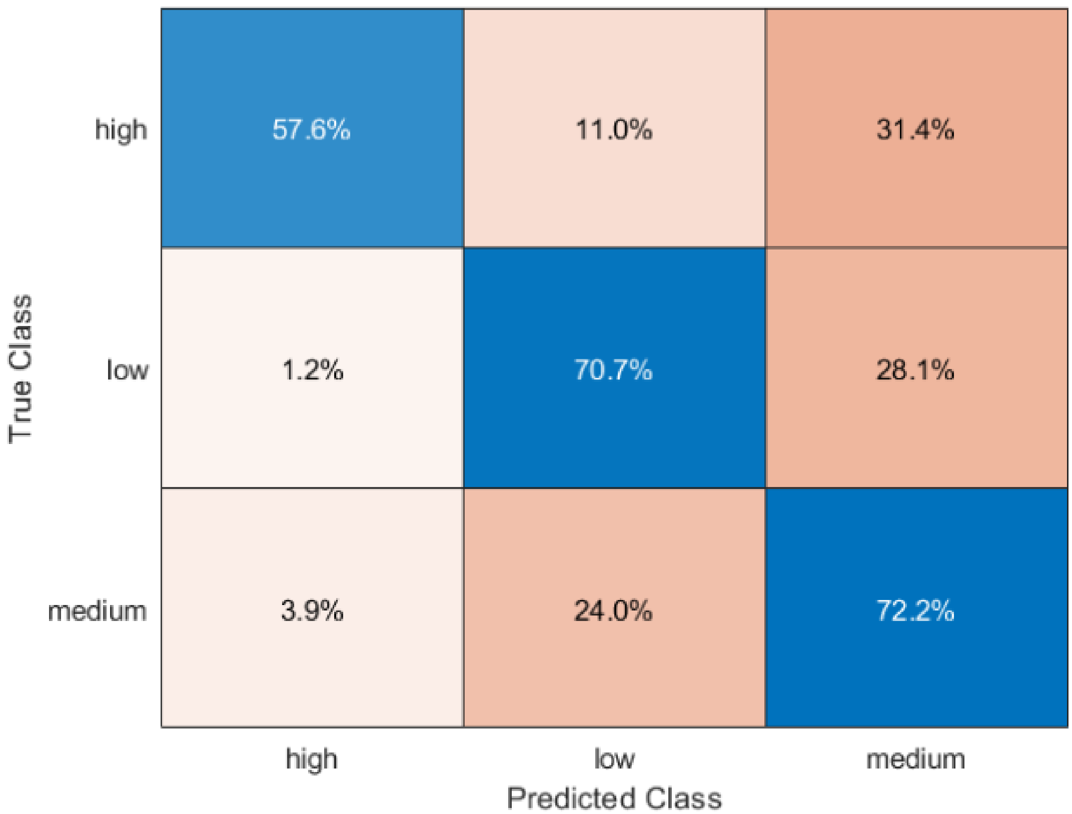
- FNR (high misclassified as low) = 0.6% (training data)
- FNR (hign misclassified as low) = 20.7% (test data)

As it can be seen, performance on the test set deteriorated for this case. It was trade-off that we had to make to satisfy the first priority.

Loss Analysis

Since the raw model and the final model both had pretty good FNR for low class, the loss in fare for both was minimum and very close.

But when combined for all cases, raw model predicts a large number of low class as medium class. As a result, loss in fare is higher in raw model compared to the final model.



The code for loss analysis - loss.mlx.

Performance Improvement

Based on the understanding of the model, it can be proposed that the model's performance can be improved by introducing some new featues like the flights arrival data, weather events etc. These areas were not explored. It is kept for future work.

Appendix

Creating a File Datastore

```
raw_data= fileDatastore('F:\15.Cody\Data Science\Predictive Modeling and Machine Learning\Taxi Data\Taxi Data\
```

```
data=readall(raw_data)
```

Adding Pickup and Dropoff Zones

```
data_2=addTaxiZones(data)
```

Summary of the Data

```
summary(data)
```

internal linkVariables:

Vendor: 2922266×1 double

Values:

Min 1

Median 2

Max 2

PickupTime: 2922266×1 datetime

Values:

Min 2015-01-01 00:00:43

Median 2015-06-20 18:21:55

Max 2015-12-31 23:59:59

DropoffTime: 2922266×1 datetime

Values:

Min 2015-01-01 00:04:02

Median 2015-06-20 18:35:14

Max 2016-01-01 22:10:58

Passengers: 2922266×1 double

Values:

Min 0

Median 1

Max 9

Distance: 2922266×1 double

Values:

Min 0

Median 1.71

Max 1.468e+07

PickupLon: 2922266×1 double

Values:

Min -171.8

Median -73.982

Max 0

PickupLat: 2922266×1 double

Values:

Min 0

Median 40.753

Max 69.703

RateCode: 2922266×1 categorical

Values:

99 127

Group 28

JFK 61564

Nassau 1051

Negotiated 9110

Newark 5046

Standard 2.8453e+06

HeldFlag: 2922266×1 categorical

Values:

N 2.898e+06

Y 24296

DropoffLon: 2922266×1 double

Values:

Min -171.8

Median -73.98

Max 0

DropoffLat: 2922266×1 double

Values:

Min	0
Median	40.753
Max	456.37

PayType: 2922266×1 categorical

Values:

Cash	1.0764e+06
Credit card	1.8323e+06
Dispute	3413
No charge	10130
Unknown	3

Fare: 2922266×1 double

Values:

Min	-150
Median	9.5
Max	4.1027e+05

ExtraCharge: 2922266×1 double

Values:

Min -45.2

Median 0

Max 579.72

Tax: 2922266×1 double

Values:

Min -1.7

Median 0.5

Max 80.35

Tip: 2922266×1 double

Values:

Min -2.7

Median 1.16

Max 650

Tolls: 2922266×1 double

Values:

Min -15

Median 0

Max 911.08

ImpSurcharge: 2922266×1 double

Values:

Min	-0.3
Median	0.3
Max	0.3

TotalCharge: 2922266×1 double

Values:

Min	-150.8
Median	11.8
Max	4.1027e+05

pickup_region: 2922266×1 categorical

Values:

Allerton/Pelham Gardens	13
Arrochar/Fort Wadsworth	2
Astoria	6167
Astoria Park	49
Auburndale	9
Baisley Park	414
Bath Beach	18
Bay Ridge	133
Bay Terrace/Fort Totten	7
Bayside	19
Bedford	1331

Bedford Park	42
Bellerose	12
Belmont	36
Bensonhurst East	31
Bensonhurst West	54
Bloomfield/Emerson Hill	3
Boerum Hill	3438
Borough Park	60
Breezy Point/Fort Tilden/Riis…	1
Briarwood/Jamaica Hills	217
Brighton Beach	18
Bronx Park	11
Bronxdale	14
Brooklyn Heights	3509
Brooklyn Navy Yard	110
Brownsville	54
Bushwick North	886
Bushwick South	1689
Cambria Heights	14
Canarsie	37
Carroll Gardens	2040
Central Harlem	8758
Central Harlem North	4022
Central Park	37173
Charleston/Tottenville	1
City Island	4
Claremont/Bathgate	38
Clinton Hill	1716
Co-Op City	9
Cobble Hill	1856

College Point	18
Columbia Street	214
Coney Island	23
Corona	108
Country Club	2
Crotona Park	2
Crotona Park East	13
Crown Heights North	1226
Crown Heights South	225
Cypress Hills	17
DUMBO/Vinegar Hill	1307
Douglaston	4
Downtown Brooklyn/MetroTech	4075
Dyker Heights	26
East Concourse/Concourse Vill…	132
East Elmhurst	487
East Flatbush/Farragut	53
East Flatbush/Remsen Village	50
East Flushing	5
East Harlem North	9290
East New York	95
East New York/Pennsylvania Av…	21
East Tremont	27
East Williamsburg	2909
Eastchester	6
Elmhurst	1055
Elmhurst/Maspeth	485
Eltingville/Annadale/Prince s…	1
Erasmus	105
Far Rockaway	7

Flatbush/Ditmas Park	385
Flatlands	47
Flushing	188
Flushing Meadows-Corona Park	304
Fordham South	24
Forest Hills	583
Forest Park/Highland Park	7
Fort Greene	3103
Fresh Meadows	10
Glen Oaks	16
Glendale	35
Governor s Island/Ellis Islan…	2
Gowanus	708
Gravesend	10
Great Kills	1
Green-Wood Cemetery	9
Greenpoint	2653
Hamilton Heights	3361
Hammels/Arverne	7
Heartland Village/Todt Hill	1
Highbridge	91
Highbridge Park	17
Hillcrest/Pomonok	33
Hollis	23
Homecrest	40
Howard Beach	20
Hunts Point	23
Inwood	297
Inwood Hill Park	13
JFK Airport	62178

Jackson Heights	1855
Jamaica	248
Jamaica Bay	2
Jamaica Estates	30
Kensington	175
Kew Gardens	169
Kew Gardens Hills	61
Kingsbridge Heights	39
LaGuardia Airport	70720
Laurelton	2
Long Island City/Hunters Point	4303
Long Island City/Queens Plaza	3242
Longwood	15
Lower Manhattan	5.5444e+05
Lower Manhattan City	26156
Madison	21
Manhattan Beach	18
Manhattanville	2827
Marble Hill	21
Marine Park/Floyd Bennett Field	7
Marine Park/Mill Basin	23
Mariners Harbor	3
Maspeth	181
Melrose South	157
Middle Village	59
Midtown	1.2961e+06
Midtown-Queens	21
Midwood	66
Morningside Heights	14152
Morrisania/Melrose	72

Mott Haven/Port Morris	593
Mount Hope	55
New Dorp/Midland Beach	3
Newark Airport	160
North Corona	117
Norwood	33
Oakland Gardens	4
Ocean Hill	97
Ocean Parkway South	50
Old Astoria	2095
Ozone Park	27
Park Slope	4490
Parkchester	27
Pelham Bay	13
Pelham Parkway	20
Port Richmond	1
Prospect Heights	1224
Prospect Park	192
Prospect-Lefferts Gardens	487
Queens Village	32
Queensboro Hill	37
Queensbridge/Ravenswood	1076
Randalls Island	193
Red Hook	291
Rego Park	325
Richmond Hill	108
Ridgewood	126
Rikers Island	3
Riverdale/North Riverdale/Fie…	29
Roosevelt Island	234

Rosedale	11
Saint Albans	9
Saint George/New Brighton	5
Saint Michaels Cemetery/Woods…	258
Schuylerville/Edgewater Park	13
Sheepshead Bay	22
Soundview/Bruckner	31
Soundview/Castle Hill	41
South Beach/Dongan Hills	3
South Jamaica	174
South Ozone Park	178
South Williamsburg	375
Springfield Gardens North	18
Springfield Gardens South	149
Spuyten Duyvil/Kingsbridge	82
Starrett City	2
Steinway	2228
Stuyvesant Heights	609
Sunnyside	5435
Sunset Park East	34
Sunset Park West	426
University Heights/Morris Hei…	79
Upper East Side	4.2224e+05
Upper West Side	2.6552e+05
Van Cortlandt Park	8
Van Cortlandt Village	41
Van Nest/Morris Park	30
Washington Heights North	653
Washington Heights South	2711
West Brighton	1

West Concourse	390
West Farms/Bronx River	24
Westchester Village/Unionport	35
Westerleigh	3
Whitestone	4
Willets Point	10
Williamsbridge/Olinville	20
Williamsburg (North Side)	6641
Williamsburg (South Side)	5352
Windsor Terrace	139
Woodhaven	60
Woodlawn/Wakefield	20
Woodside	1897
NumMissing	48799

drop_region: 2922266×1 categorical

Values:

Allerton/Pelham Gardens	170
Arden Heights	16
Arrochar/Fort Wadsworth	63
Astoria	15209
Astoria Park	77
Auburndale	176
Baisley Park	837
Bath Beach	249
Bay Ridge	2546
Bay Terrace/Fort Totten	217
Bayside	388

Bedford	6192
Bedford Park	468
Bellerose	143
Belmont	296
Bensonhurst East	397
Bensonhurst West	610
Bloomfield/Emerson Hill	69
Boerum Hill	5455
Borough Park	681
Breezy Point/Fort Tilden/Riis…	28
Briarwood/Jamaica Hills	803
Brighton Beach	278
Broad Channel	16
Bronx Park	128
Bronxdale	231
Brooklyn Heights	8264
Brooklyn Navy Yard	431
Brownsville	396
Bushwick North	3981
Bushwick South	6462
Cambria Heights	178
Canarsie	510
Carroll Gardens	3372
Central Harlem	17038
Central Harlem North	11821
Central Park	33415
Charleston/Tottenville	12
City Island	43
Claremont/Bathgate	266
Clinton Hill	6660

Co-Op City	210
Cobble Hill	2462
College Point	297
Columbia Street	972
Coney Island	234
Corona	1028
Country Club	62
Crotona Park	14
Crotona Park East	159
Crown Heights North	6454
Crown Heights South	1622
Cypress Hills	310
DUMBO/Vinegar Hill	4184
Douglaston	224
Downtown Brooklyn/MetroTech	5395
Dyker Heights	493
East Concourse/Concourse Vill…	1112
East Elmhurst	1203
East Flatbush/Farragut	501
East Flatbush/Remsen Village	438
East Flushing	172
East Harlem North	20357
East New York	681
East New York/Pennsylvania Av…	241
East Tremont	246
East Williamsburg	7665
Eastchester	124
Elmhurst	2853
Elmhurst/Maspeth	1462
Eltingville/Annadale/Prince s…	24

Erasmus	485
Far Rockaway	117
Flatbush/Ditmas Park	2790
Flatlands	537
Flushing	1351
Flushing Meadows-Corona Park	560
Fordham South	181
Forest Hills	3673
Forest Park/Highland Park	55
Fort Greene	5485
Fresh Meadows	316
Freshkills Park	2
Glen Oaks	153
Glendale	412
Governor s Island/Ellis Islan…	1
Gowanus	1920
Gravesend	161
Great Kills	25
Green-Wood Cemetery	46
Greenpoint	9947
Grymes Hill/Clifton	24
Hamilton Heights	7957
Hammels/Arverne	107
Heartland Village/Todt Hill	59
Highbridge	562
Highbridge Park	149
Hillcrest/Pomonok	454
Hollis	121
Homecrest	380
Howard Beach	235

Hunts Point	310
Inwood	2237
Inwood Hill Park	190
JFK Airport	25608
Jackson Heights	5368
Jamaica	765
Jamaica Bay	3
Jamaica Estates	380
Kensington	968
Kew Gardens	703
Kew Gardens Hills	639
Kingsbridge Heights	343
LaGuardia Airport	35891
Laurelton	171
Long Island City/Hunters Point	9892
Long Island City/Queens Plaza	3846
Longwood	288
Lower Manhattan	4.9608e+05
Lower Manhattan City	28694
Madison	345
Manhattan Beach	183
Manhattanville	4741
Marble Hill	152
Marine Park/Floyd Bennett Field	19
Marine Park/Mill Basin	339
Mariners Harbor	39
Maspeth	1135
Melrose South	850
Middle Village	961
Midtown	1.2059e+06

Midtown-Queens	394
Midwood	611
Morningside Heights	21721
Morrisania/Melrose	430
Mott Haven/Port Morris	2065
Mount Hope	560
New Dorp/Midland Beach	35
Newark Airport	4649
North Corona	866
Norwood	401
Oakland Gardens	208
Oakwood	13
Ocean Hill	693
Ocean Parkway South	279
Old Astoria	4417
Ozone Park	222
Park Slope	11263
Parkchester	372
Pelham Bay	163
Pelham Bay Park	32
Pelham Parkway	365
Port Richmond	13
Prospect Heights	3550
Prospect Park	697
Prospect-Lefferts Gardens	2373
Queens Village	283
Queensboro Hill	239
Queensbridge/Ravenswood	1737
Randalls Island	470
Red Hook	1295

Rego Park	1175
Richmond Hill	600
Ridgewood	1952
Rikers Island	3
Riverdale/North Riverdale/Fie…	848
Rockaway Park	114
Roosevelt Island	1425
Rosedale	228
Rossville/Woodrow	18
Saint Albans	295
Saint George/New Brighton	66
Saint Michaels Cemetery/Woods…	267
Schuylerville/Edgewater Park	337
Sheepshead Bay	355
Soundview/Bruckner	323
Soundview/Castle Hill	379
South Beach/Dongan Hills	55
South Jamaica	261
South Ozone Park	1059
South Williamsburg	1190
Springfield Gardens North	276
Springfield Gardens South	541
Spuyten Duyvil/Kingsbridge	1094
Stapleton	50
Starrett City	85
Steinway	6889
Stuyvesant Heights	3921
Sunnyside	8326
Sunset Park East	693
Sunset Park West	2163

University Heights/Morris Hei...	525
Upper East Side	4.216e+05
Upper West Side	2.4929e+05
Van Cortlandt Park	123
Van Cortlandt Village	510
Van Nest/Morris Park	301
Washington Heights North	5533
Washington Heights South	10589
West Brighton	35
West Concourse	1156
West Farms/Bronx River	301
Westchester Village/Unionport	255
Westerleigh	45
Whitestone	339
Willets Point	28
Williamsbridge/Olinville	274
Williamsburg (North Side)	11766
Williamsburg (South Side)	10825
Windsor Terrace	1601
Woodhaven	552
Woodlawn/Wakefield	377
Woodside	3860
NumMissing	51915

Data Cleaning

```
%invalid ratecode
dp2 = dp(dp.RateCode ~= "99", :);

%invalid location
dp3 = standardizeMissing(dp2, 0, "DataVariables", ["PickupLat","PickupLon","DropoffLat","DropoffLon"]);
dp3 = rmmissing(dp3, "DataVariables", ["PickupLat","PickupLon","DropoffLat","DropoffLon"]);

%passengers
dp4=dp3(dp3.Passengers>0,:);

% distance
```

```
boxplot(dp4.Distance)
histogram(dp4.Distance)
prctile(dp4.Distance,[0,99])
dp5=dp4(dp4.Distance>0,:);
prctile(dp5.Distance,[0,99])
prctile(dp5.Distance,[0,99.9])
prctile(dp5.Distance,[0,99.99])

dp6=rmoutliers(dp5,"percentiles",[0,99.99],"DataVariables","Distance")
histogram(dp6.Distance)

% fare
prctile(dp6.Fare,[0,99])
sum(dp6.Fare<=0)
sum(dp6.Fare<=2.5)

dp7=dp6(dp6.Fare>0,:);

prctile(dp7.Fare,[0,99.9])
prctile(dp7.Fare,[0,99.99])
sum(dp7.Fare>140)
histogram(dp7.Fare)

dp8=rmoutliers(dp7,"percentiles",[0,99.99],"DataVariables","Fare")

boxplot(dp8.Fare)

dp9=dp8(dp8.Fare>=2.5,:);

% extra charge
dp9=dp9(dp9.ExtraCharge>=0,:);
dp9=dp9(dp9.Tax>=0,:);
dp9=dp9(dp9.Tip>=0,:);
dp9=dp9(dp9.Tolls>=0,:);
dp9=dp9(dp9.ImpSurcharge>=0,:);
dp10=dp9(dp9.TotalCharge>=0,:);

boxplot(dp10.Tax)
histogram(dp10.Tax)

dp11 = dp10(abs(dp10.ImpSurcharge-0.3) < 0.01, :);
dp11 = dp11(abs(dp11.Tax-0.5) < 0.01, :);
dp11 = dp11(abs(dp11.Fare + dp11.ExtraCharge + dp11.Tax + dp11.Tip + dp11.Tolls + dp11.ImpSurcharge - dp11.Tot

boxplot(dp11.Tax)

% fare distance ratio
x=dp11.Fare./dp11.Distance;
prctile(x,[0,99])
prctile(x,[0,99.99])
sum(x>520)

dp12=dp11(x<=520,:);



df = addDuration(dp12); % minutes
```

```
df = addAveSpeed(df); % mph

boxplot(df.Duration)
histogram(df.Duration)
prctile(df.Duration,[0,99])
prctile(df.Duration,[0,99.5])
sum(df.Duration<=0)

df2=df(df.Duration>=1 & df.Duration<120,:)

df2=df2(df2.AveSpeed>=0.1 & df2.AveSpeed<100,:);
df2=df2(df2.TotalCharge>=0.5 & df2.TotalCharge<=120,:);
df3=df2(df2.Tolls<=20,:);



timeofday(df3.PickupTime);

hour(df3.PickupTime(1:6))

writetable(df3,'prepared_dataset_01.csv')

df4=prepareddataset01;

lat = [40.5612 40.9637];
lon = [-74.1923 -73.5982];

inROI = inpolygon(df4.PickupLat,df4.PickupLon, lat([1 2 2 1]),lon([1 1 2 2])) ...
    & inpolygon(df4.DropoffLat,df4.DropoffLon, lat([1 2 2 1]),lon([1 1 2 2]));

% Only keep trips that begin and end inside the region of interest.
df5 = df4(inROI,:);
```

Hourly Data

```
df5.hourly_data= dateshift(df5.PickupTime,"start","hour")

df5.hourly_data_dropoff= dateshift(df5.DropoffTime,"start","hour");

df6=df5(:,["PickupTime","DropoffTime","Distance","Fare","ExtraCharge","Tax","Tip","Tolls","ImpSurcharge","Tota

writetable(df6,'short_dataset.csv')
```

Removing Other Regions

```
ds=df6

regions=["Lower Manhattan","Midtown","Upper East Side","Upper West Side","JFK Airport","LaGuardia Airport"]

%sort(unique(ds.pickup_region),'d')
%ds.pickup_region=string(ds.pickup_region);
%ds2=ds(contains(ds.pickup_region,regions),:)
%unique(ds2.pickup_region)

ds2=ds((ds.pickup_region==regions(1) | ds.pickup_region==regions(2) | ds.pickup_region==regions(3) | ds.pickup

unique(ds2.pickup_region)
```

```
ds3=ds2((ds2.drop_region==regions(1) | ds2.drop_region==regions(2) | ds2.drop_region==regions(3) | ds2.drop_re

unique(ds3.drop_region)
```

grouping

```
gp=groupsummary(ds3,["pickup_region","hourly_data"],"mean",["Duration","Distance","Fare"])
gp.Properties.VariableNames(3)="pickup_count"

gd=groupsummary(ds3,["drop_region","hourly_data_dropoff"],"mean",["Duration","Distance","Fare"])
gd.Properties.VariableNames(3)="drop_count"

sum(ismissing(gd))
sum(ismissing(gp))

gp.Properties.VariableNames(1)="region"
gd.Properties.VariableNames(1)="region"
gd.Properties.VariableNames(2)="hourly_data"
```

Joining

```
dj= outerjoin(gp,gd,"Keys",["region","hourly_data"])

dj_2= outerjoin(gp,gd,"Keys",["region","hourly_data"],"MergeKeys",true)

Data Missing
sum(ismissing(dj_2))

%dj_3=fillmissing(dj_2,"constant",0)

dj_3=fillmissing(dj_2,"constant",0,'DataVariables',@isnumeric)

Combining

dj_3.netpickups= dj_3.pickup_count- dj_3.drop_count

dj_3.avg_duration=(dj_3.mean_Duration_gd + dj_3.mean_Duration_gp)/2

dj_3.avg_distance=(dj_3.mean_Distance_gd + dj_3.mean_Distance_gp)/2

dj_3.avg_fare=(dj_3.mean_Fare_gd + dj_3.mean_Fare_gp)/2
```

Train-test split

```
rng(1)

partition=cvpartition(height(dj_4),"HoldOut",0.2)

train_idx=training(partition);
test_idx=test(partition);

train_data= dj_4(train_idx,:);
test_data = dj_4 (test_idx,:);
```

Generating Response Variable

```
train_data.demand= discretize(train_data.netpickups,[-inf,0,15,inf],"categorical",["low","medium","high"])
```

Summary Statistics

```
groupsummary(train_data,"demand")

groupsummary(test_data,"demand")

groupsummary(train_data,["demand","region"])

groupsummary(test_data,["demand","region"])
```

Feature Selection

```
%heatmap(df.demand,df.DayOfWeek)

crosstab(df.demand,df.DayOfWeek)

[a,chi2,p]=crosstab(df.demand,df.DayOfWeek)

[a,chi2,p]=crosstab(df.demand,df.dayofyear)

[p,tbl]=anova1(df.dayofyear,df.demand)

[p,tbl]=anova1(df.netpickups,df.dayofyear)

%[p,tbl]=anova1(string(df.demand),df.dayofyear)

df_2= isholiday(df,holidays)

[a,chi2,p]=crosstab(df_2.demand,df_2.isholiday)

[a,chi2,p]=crosstab(df_2.demand,df_2.DayOfWeek)

s=groupsummary(df_2,["DayOfWeek","region"],"mean","netpickups")

%gscatter(s.DayOfWeek,s.region,s.mean_netpickups)

heatmap(s,"DayOfWeek","mean_netpickups")

df_2.hourofday=hours(timeofday(df_2.hourly_data))

%corr(df_2.demand,df_2.avg_duration)

df_3=df_2

%df_3.demand(df_2.demand=='low')=0

df_3.demand=grp2idx(df_3.demand)

summary(df_3)

corr(df_3.demand,df_3.avg_duration)

corr(df_3.demand,df_3.avg_distance)

corr(df_3.demand,df_3.avg_fare)

%corr(df_3.demand,df_3.DayOfWeek)
```

Feature Creation

```
df=dj_4

[~,df.DayOfWeek] = weekday(df.hourly_data,"long")

df.DayOfWeek = categorical(cellstr(df.DayOfWeek));

x=df.hourly_data(26)
x2=datevec(x)
x3=datenum(x2(1:3))
day = x3 - datenum(x2(1), 1,0)

df=adddayofyear(df)

df.demand= discretize(df.netpickups,[-inf,0,15,inf],"categorical",["low","medium","high"])
```

Raw Model

```
y_pred=raw_model_bagged.predictFcn(test_data)

cMetrics(test_data.demand,y_pred)
```

Oversampling the Minority Class

```
x_train_v1= [x_train,y_train]
xhigh=x_train_v1(x_train_v1.demand=='high',:)
xothers=x_train_v1(x_train_v1.demand~='high',:)

histogram(x_train_v1.demand)
[a,b]=histcounts(x_train_v1.demand)
xhigh_os= datasample(xhigh,12000,"Replace",true)

combining

x_comb=[xhigh_os;xothers]

histogram(x_comb.demand)
```

Prediction on Test Data

```
y_pred=model_bag_unb_cost.predictFcn(test_data)
cMetrics(test_data.demand,y_pred)
confusionchart(test_data.demand,y_pred,"Normalization","row-normalized")

y_pred=model_bag_unb_2.predictFcn(test_data)
cMetrics(test_data.demand,y_pred)
confusionchart(test_data.demand,y_pred,"Normalization","row-normalized")
```

Classifier Function

```
function [trainedClassifier, validationAccuracy] = trainClassifier_01(trainingData)
```

```
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 09-Apr-2021 17:24:47


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'region', 'avg_duration', 'avg_distance', 'avg_fare', 'DayOfWeek', 'dayofyear', 'isholiday',
predictors = inputTable(:, predictorNames);
response = inputTable.demand;
isCategoricalPredictor = [true, false, false, false, true, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
template = templateTree(...
    'MaxNumSplits', 40508);
classificationEnsemble = fitcensemble(...
    predictors, ...
    response, ...
```

```
    'Method', 'Bag', ...
    'NumLearningCycles', 30, ...
    'Learners', template, ...
    'Cost', [0 4 2; 10 0 6; 1 1 0], ...
    'ClassNames', categorical({'high'; 'low'; 'medium'}));

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
ensemblePredictFcn = @(x) predict(classificationEnsemble, x);
trainedClassifier.predictFcn = @(x) ensemblePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'DayOfWeek', 'avg_distance', 'avg_duration', 'avg_fare', 'dayofyear', '
trainedClassifier.ClassificationEnsemble = classificationEnsemble;
trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2020a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit = c.predictFcn(

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'region', 'avg_duration', 'avg_distance', 'avg_fare', 'DayOfWeek', 'dayofyear', 'isholiday',
predictors = inputTable(:, predictorNames);
response = inputTable.demand;
isCategoricalPredictor = [true, false, false, false, true, false, false, false];

% Perform cross-validation
partitionedModel = crossval(trainedClassifier.ClassificationEnsemble, 'KFold', 5);

% Compute validation predictions
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);

% Compute validation accuracy
validationAccuracy = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```