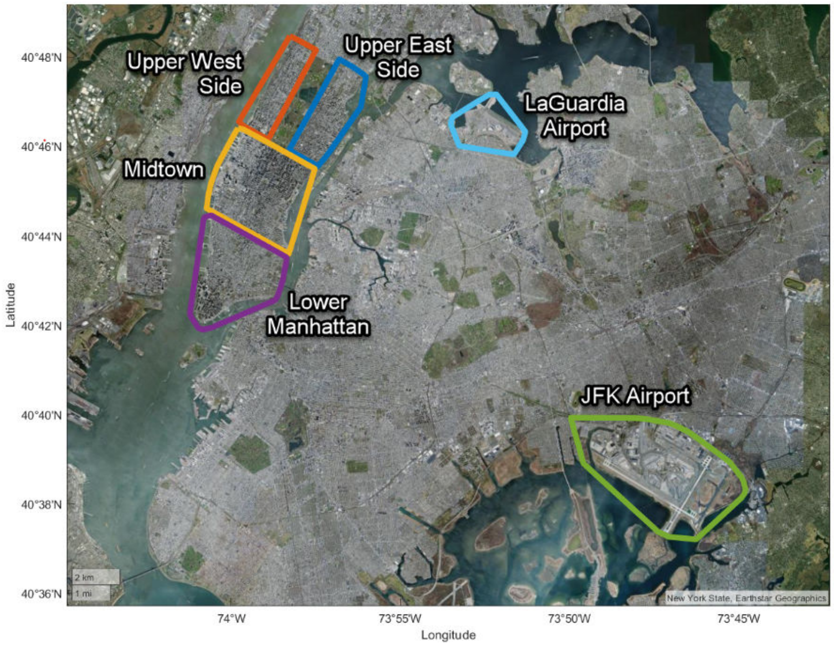
Capstone Project : Predicting Demand for Taxi Data

Table of Contents

Objectives

In this report, I **create** a new **model** that predicts **demand around Manhattan and the airports**. This report wll present my findings to Mr. Walker and the business team. I hope this is a compelling final report detailing my findings to get the model implemented. A model that successfully predicts demand will enable Super Taxis to focus on high demand regions and avoid low demand ones, thus improving efficiency and increasing profits.



Data

Brief Description

Manhattan is often broken into distinct regions: Lower Manhattan, Midtown, Upper East Side, and Upper West Side, as shown in the map above. In addition to the Manhattan regions, my analysis needs to include LaGuardia and JFK airports.

Load all taxi data into datastore

```
taxiDataStore = fileDatastore("/MATLAB Drive/Predictive Modeling and Machine Learning/Taxi Data/yellow_tripdata_2015-*.csv","ReadFcn",@importTaxiDataWithoutCleaning,"UniformRead",true);
taxiAll = readall(taxiDataStore);
```

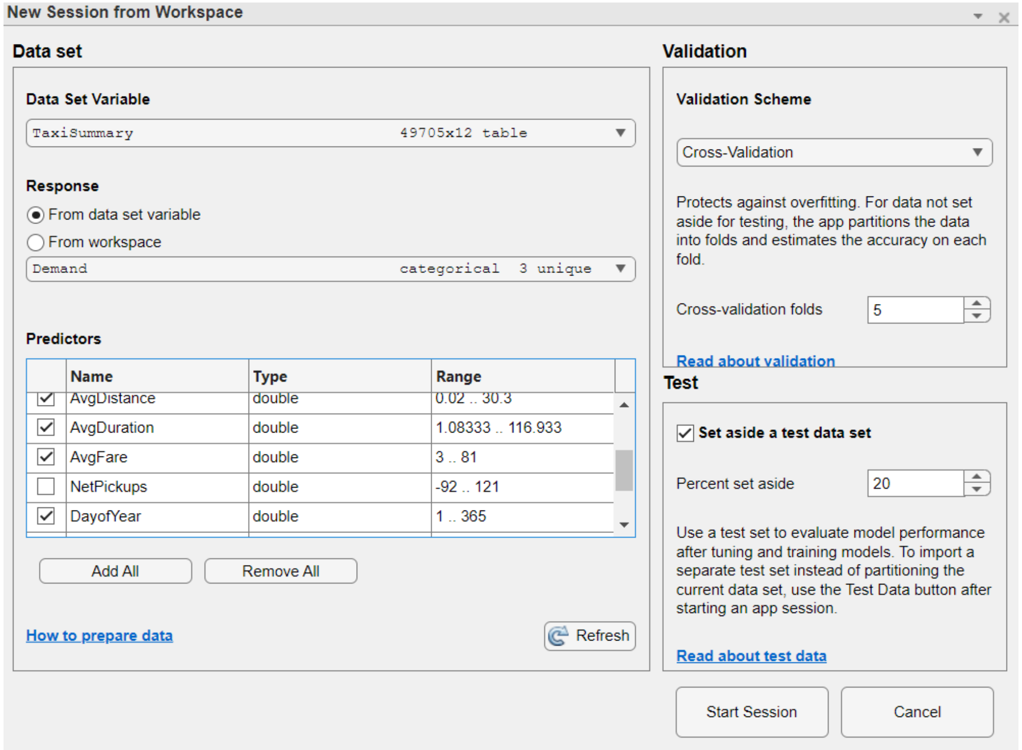
Preprocessing Data

I'll start by adding the pickup region and drop-off region for each taxi trip to the table of individual rides. After that, I can then start grouping rides by region and hour to find the number of pickups and drop-offs in a given region for a given hour.

- Creating the taxi regions (Task 1 : Creating the taxi regions)
- Data cleaning (Task 2 : Data Cleaning)
- Data Restructuring - Creating Group Summary Table (Task 3 : Data Restructuring - Creating Group Summary Table)

Description of test data split

For creating machine learning model, we have to split the dataset into a training set and a test set. I split the data into 80% data training and 20% of data testing




```
rng(1);
taxiPartitions = cvpartition(height(TaxiSummary),"HoldOut",0.2) % make data testing 20% of all taxi data
```

```
taxiPartitions =
Hold-out cross validation partition
   NumObservations: 49705
       NumTestSets: 1
         TrainSize: 39764
          TestSize: 9941
          IsCustom: 0
```

```
taxiTestIdx = test(taxiPartitions);
taxiTest = TaxiSummary(taxiTestIdx,:);
taxiTrainIdx = training(taxiPartitions);
taxiTrain = TaxiSummary(taxiTrainIdx,:);
```

Feature Engineering

After splitting data into a training set and test set, we then create response feature. I have previously computed the number of net pickups (pickups - dropoffs) for each time interval and region and I wanna used this variable to gauge demand for taxi service.

To achieve this goal, I will convert net pickups to a categorical feature, 'Demand', and use it as the response variable for the classification model(s). To create *Demand*, I convert my numerical net pickup values to categorical values according to the following definitions:

- Net Pickups < 0: 'Low'
- 0 <= Net Pickups < 15: 'Medium'
- Net Pickups >= 15: 'High'

```
categories = {'Low', 'Medium', 'High'};
taxiTrain.Demand = discretize(taxiTrain.NetPickups, [-Inf, 0, 15, Inf],
'Categorical', categories);
```

```
percentage_high_demand = 100*height(taxiTrain.Demand(taxiTrain.Demand ==
"High"))/height(taxiTrain.Demand)
```

```
percentage_high_demand = 16.0874
```

```
groupsummary(taxiTrain,"Demand","none")
```

ans = 3x2 table

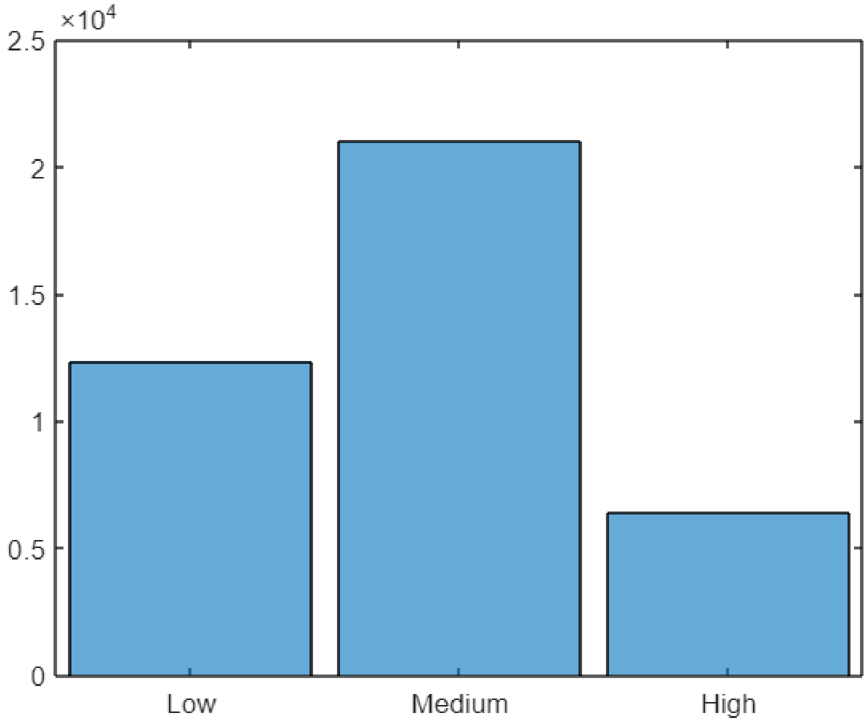
	Demand	GroupCount
1	Low	12355
2	Medium	21012
3	High	6397

```
taxiTrain_Low = taxiTrain(taxiTrain.Demand == "Low",:);
groupsummary(taxiTrain_Low,"Region","none")
```

ans = 6x2 table

	Region	GroupCount
1	JFK Airport	771
2	LaGuardia Airport	974
3	Lower Manhattan	2342
4	Midtown	2159
5	Upper East Side	3353
6	Upper West Side	2756

```
histogram(taxiTrain.Demand)
```



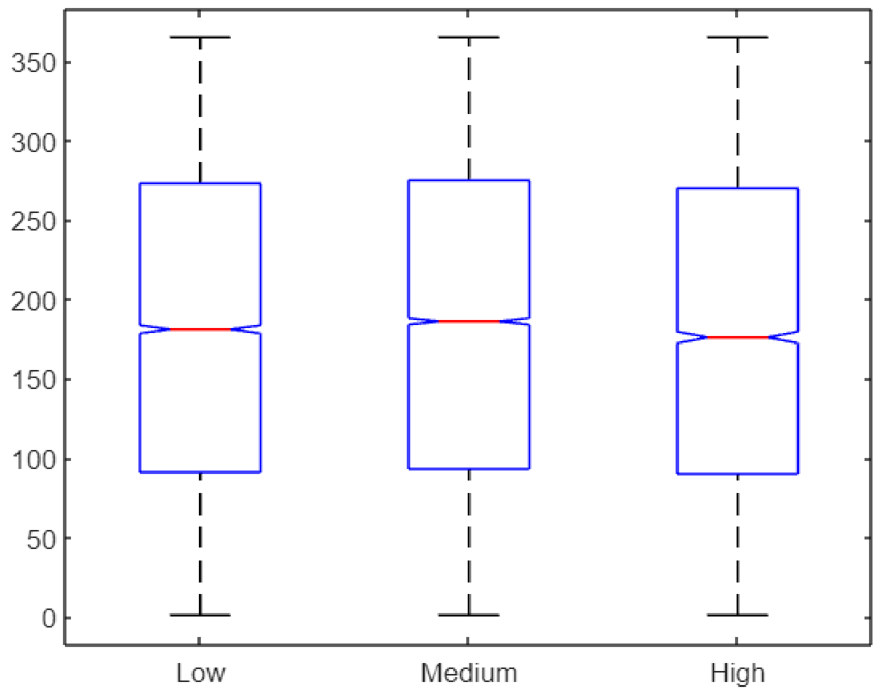

```
% Create and evaluate feature
taxiTrain_evaluate = taxiTrain;
taxiTrain_evaluate.DayofYear = day(taxiTrain_evaluate.HourlyBin,'dayofyear');
DayofYear_category = categorical(taxiTrain_evaluate.DayofYear);
[~,~,p] = crosstab(DayofYear_category,taxiTrain_evaluate.Demand)
```

```
p = 0.8363
```

```
p = anova1(taxiTrain_evaluate.DayofYear,taxiTrain_evaluate.Demand)
```

ANOVA Table

Source	SS	df	MS	F	Prob>F
Groups	96594.1	2	48297.1	4.37	0.0127
Error	439526899.6	39761	11054.2		
Total	439623493.7	39763			




```
p = 0.0127
```

```
BankHoliday = readtable("/MATLAB Drive/Predictive Modeling and Machine Learning/2015 Bank Holidays.csv")
```

BankHoliday = 12x2 table

	Date	Holiday
1	01-Jan-2015	'New Year's Day'
2	19-Jan-2015	'Martin Luther King Jr. Day'
3	16-Feb-2015	'Washington's Birthday/President's Day'
4	03-Apr-2015	'Good Friday (not federal holiday, but stock markets closed)'
5	25-May-2015	'Memorial Day'
6	03-Jul-2015	'Independence Day (observeded)'
7	04-Jul-2015	'Independence Day'
8	07-Sep-2015	'Labor Day'
9	12-Oct-2015	'Columbus Day'
10	11-Nov-2015	'Veteran's Day'
11	26-Nov-2015	'Thanksgiving'
12	25-Dec-2015	'Christmas'

```
BankHoliday.DayofYear = day(BankHoliday.Date,'dayofyear')
```

BankHoliday = 12x3 table

	Date	Holiday	DayofYear
1	01-Jan-2015	'New Year's Day'	1
2	19-Jan-2015	'Martin Luther King Jr. Day'	19
3	16-Feb-2015	'Washington's Birthday/President's Day'	47
4	03-Apr-2015	'Good Friday (not federal holiday, but stock markets closed)'	93
5	25-May-2015	'Memorial Day'	145
6	03-Jul-2015	'Independence Day (observeded)'	184
7	04-Jul-2015	'Independence Day'	185
8	07-Sep-2015	'Labor Day'	250
9	12-Oct-2015	'Columbus Day'	285
10	11-Nov-2015	'Veteran's Day'	315
11	26-Nov-2015	'Thanksgiving'	330
12	25-Dec-2015	'Christmas'	359

```
isHoliday = ismember(taxiTrain_evaluate.DayofYear,BankHoliday.DayofYear);
[~,chi2,p] = crosstab(isHoliday,taxiTrain_evaluate.Demand)
```

```
chi2 = 11.5734
p = 0.0031
```

Modelling

Objectives

In this section, I try to find the best model for predicts demand around Manhattan and the airports. To find the best model, I will show some of the process of creating model I have tried and how performance of the model is, that is determined by three parameters:

1. Accuracy
2. Confusion matrix
3. **cMetrics** output (or equivalent)

Final Model Description

Model 3: SVM
Status: Tested

Training Results
Accuracy (Validation) 99.5%
Total cost (Validation) 189 (Models have different cost matrices)
Prediction speed ~360000 obs/sec
Training time 46.08 sec
Model size (Compact) ~22 kB

Test Results
Accuracy (Test) 98.7%
Total cost (Test) 134 (Models have different cost matrices)

Model Hyperparameters

Preset: Linear SVM
Kernel function: Linear
Kernel scale: Automatic
Box constraint level: 1
Multiclass coding: One-vs-One
Standardize data: Yes

Feature Selection: 9/9 individual features selected

	Select	Features
1	☑	Region
2	☑	PickupCount
3	☑	DropoffCount

Training Description

First, for creating training model, I have created this code

```
taxiTrain.IsHoliday = isHoliday;
taxiTrain.HourOfDay = hour(taxiTrain_evaluate.HourlyBin);
head(taxiTrain)
```

Region	HourlyBin	PickupCount	DropoffCount	AvgDistance	AvgDuration	Av
JFK Airport	01-Jan-2015 05:00:00	1	0	19.93	36	
JFK Airport	01-Jan-2015 07:00:00	2	2	19.815	27.067	
JFK Airport	01-Jan-2015 09:00:00	1	3	17.27	23.7	
JFK Airport	01-Jan-2015 14:00:00	4	1	15.777	27.333	
JFK Airport	01-Jan-2015 15:00:00	3	1	16.14	32.933	44
JFK Airport	01-Jan-2015 16:00:00	5	2	18.28	35.597	
JFK Airport	01-Jan-2015 17:00:00	5	3	15.972	29.68	
JFK Airport	01-Jan-2015 18:00:00	3	3	17.293	29.133	

```
taxiTest.DayofYear = day(taxiTest.HourlyBin,'dayofyear');
isHoliday1 = ismember(taxiTest.DayofYear,BankHoliday.DayofYear);
taxiTest.IsHoliday = isHoliday1;
taxiTest.HourOfDay = hour(taxiTest.HourlyBin);
head(taxiTest)
```

Region	HourlyBin	PickupCount	DropoffCount	AvgDistance	AvgDuration	Av
JFK Airport	01-Jan-2015 00:00:00	2	0	11.71	22.525	3
JFK Airport	01-Jan-2015 01:00:00	2	0	18.25	27.183	4
JFK Airport	01-Jan-2015 06:00:00	2	2	17.855	24.917	4
JFK Airport	01-Jan-2015 11:00:00	2	4	15.95	31.817	

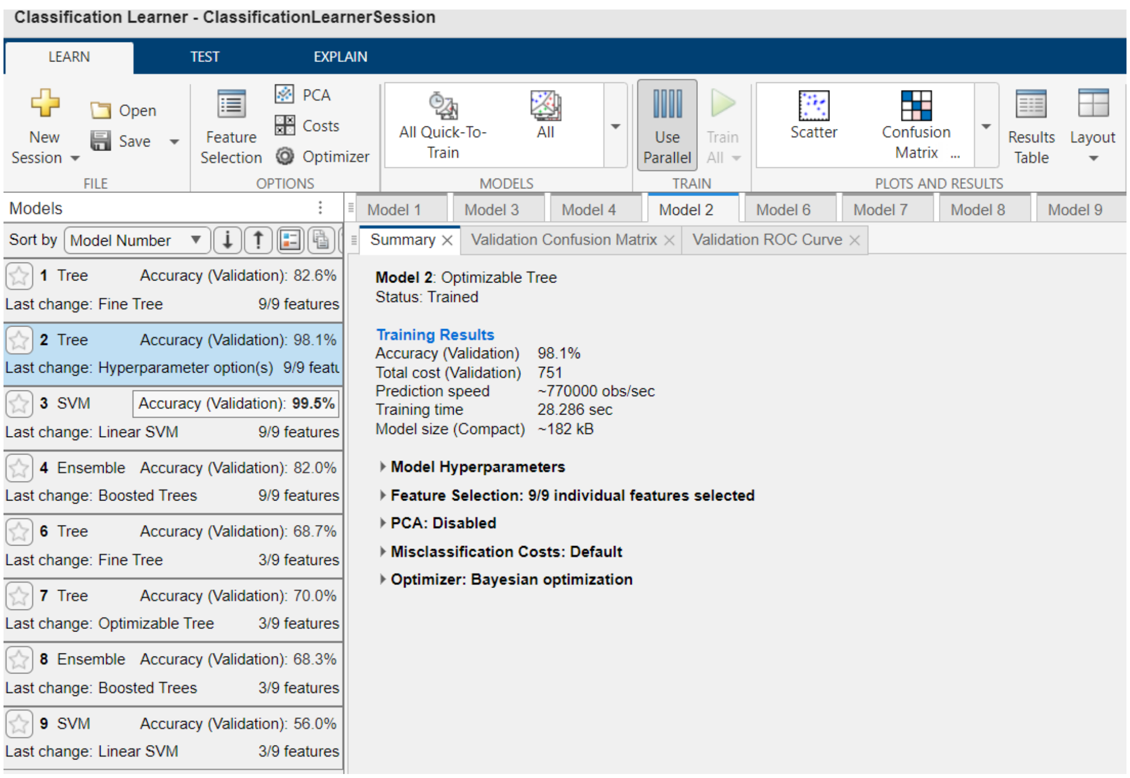
JFK Airport	01-Jan-2015 12:00:00	4	1	14.953	24.358	4
JFK Airport	01-Jan-2015 13:00:00	4	2	15.045	23.242	4
JFK Airport	02-Jan-2015 00:00:00	3	0	17.873	23.994	
JFK Airport	02-Jan-2015 14:00:00	7	2	19.383	41.838	52

```
TaxiSummary.DayofYear = day(TaxiSummary.HourlyBin,'dayofyear');
isHoliday_2 = ismember(TaxiSummary.DayofYear,BankHoliday.DayofYear);
TaxiSummary.IsHoliday = isHoliday_2;
TaxiSummary.HourOfDay = hour(TaxiSummary.HourlyBin);
categories = {'Low', 'Medium', 'High'};
TaxiSummary.Demand = discretize(TaxiSummary.NetPickups, [-Inf, 0, 15, Inf],
'Categorical', categories);
head(TaxiSummary)
```

Region	HourlyBin	PickupCount	DropoffCount	AvgDistance	AvgDuration	Av
JFK Airport	01-Jan-2015 00:00:00	2	0	11.71	22.525	3
JFK Airport	01-Jan-2015 01:00:00	2	0	18.25	27.183	4
JFK Airport	01-Jan-2015 05:00:00	1	0	19.93	36	
JFK Airport	01-Jan-2015 06:00:00	2	2	17.855	24.917	4
JFK Airport	01-Jan-2015 07:00:00	2	2	19.815	27.067	
JFK Airport	01-Jan-2015 09:00:00	1	3	17.27	23.7	
JFK Airport	01-Jan-2015 11:00:00	2	4	15.95	31.817	
JFK Airport	01-Jan-2015 12:00:00	4	1	14.953	24.358	4

I tried several method for model type and doing hyperparameter optimization and manipulating cost

Results



Overall, the training and validation shows us that using only 3 features (Region, IsHoliday, and HourOfDay) are not give great accuracy, while 9 features in group summary table, excluding the NetPickUps give us great accuracy for training and vaidation, about 99.5 % for Linear SVM.

Applying two scenario,

Scenario 1

Start off with a baseline model that emphasizes overall accuracy. Use test data set to verify your modeling results. This model will be useful for analysis and comparison later. It may be also a good way to investigate model types and hyperparameters, class imbalance, and which features are most useful.

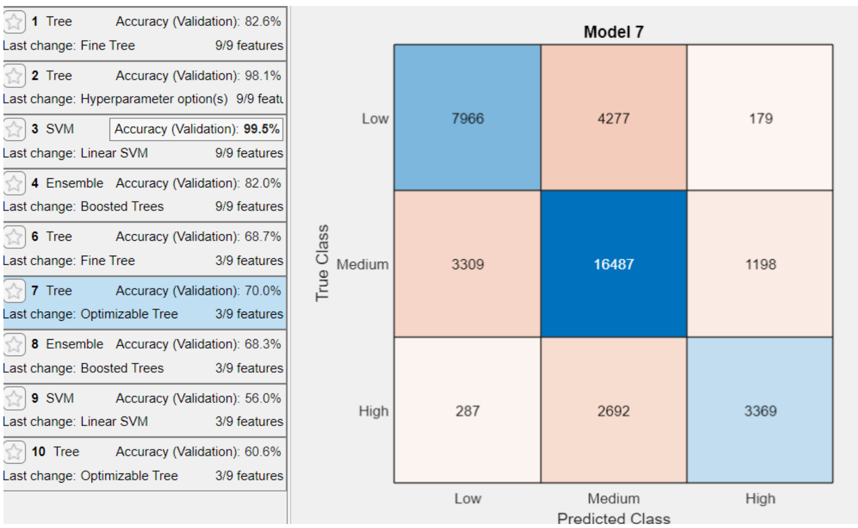
Scenario 2

Assume the following **taxi deployment strategy** will be followed based on demand:

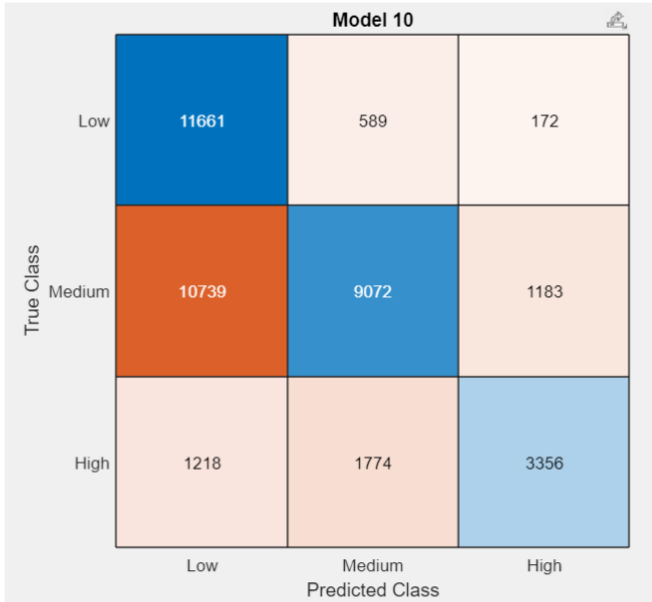
- Always go to the nearest High demand region when one is available
- Go to the nearest Medium demand region if there is no High demand region available
- Never go to or stay in a Low demand region

Applying those 2 scenario to analysis of 3 feature as predictor give this results

- Scenario 1

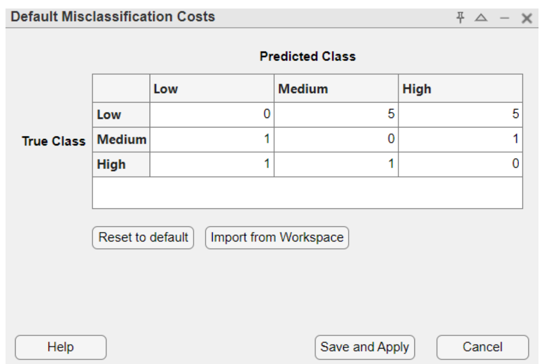


- Scenario 2



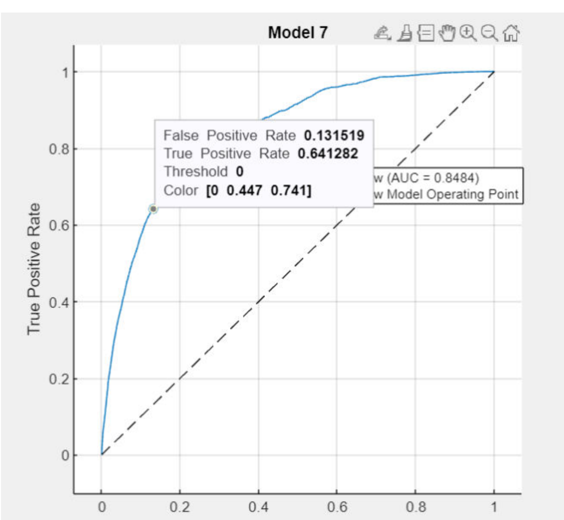
Due to the result, it show that The Scenario 1 model has a lower recall for the Low demand class than the Scenario 2 model.

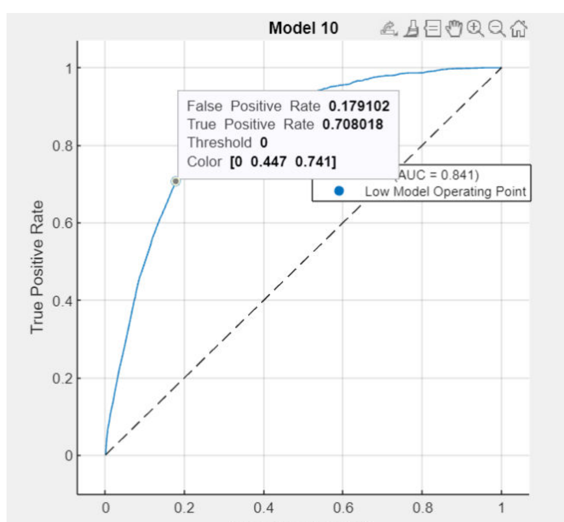
For scenario 2, I give a bigger cost metric



	Low	Medium	High
Low	0	5	5
Medium	1	0	1
High	1	1	0

After training and validation, we can also see those analysis from the ROC Curve



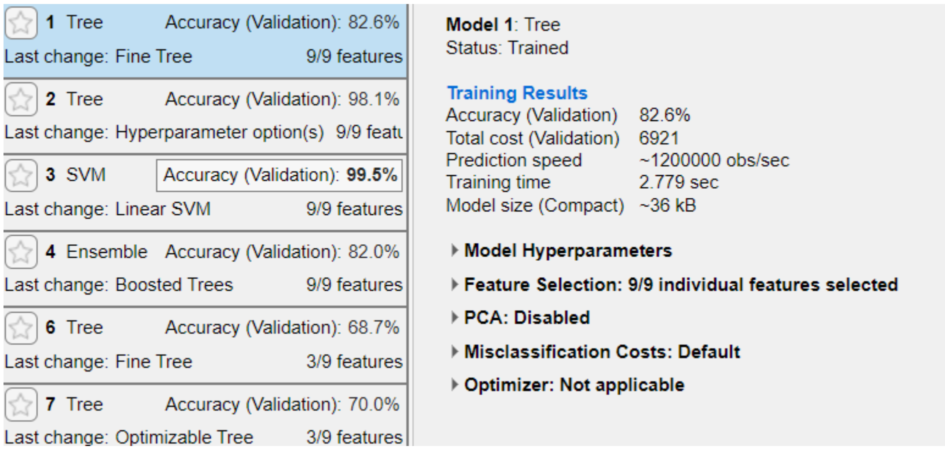


Although the recall is better for scenario 2, it has the drawback that the fallout get bigger.

Considering using more feature

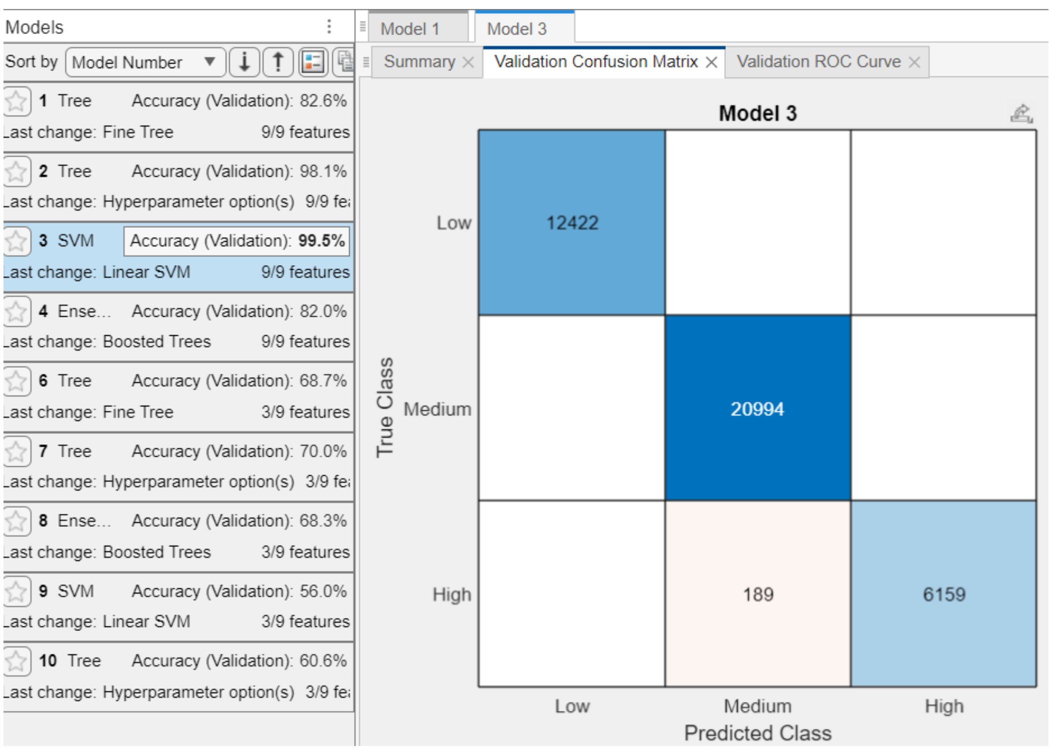
After trial and error for predicting the demand (Appendix 2), I think using 9 feature in groupsummary table, except NetPickups because it doesn't mean anything can help us get better result.

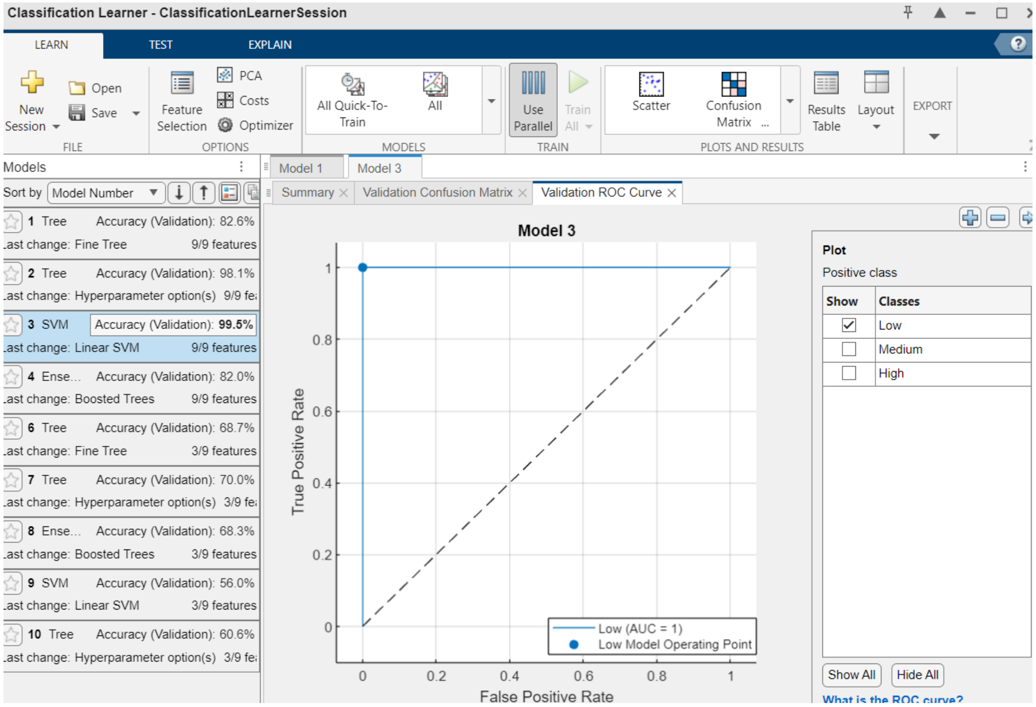
As shown in the first training, the accuracy of using finetree and 9 features is 82.6% without hyperparameter tuning or optimazation method.



After applying hyperparameter, the model show 98.1 % accuracy, which is better than without hyperparameter. Using different model, such as SVM with 9 features can give 99.5% accuracy for training and validation.

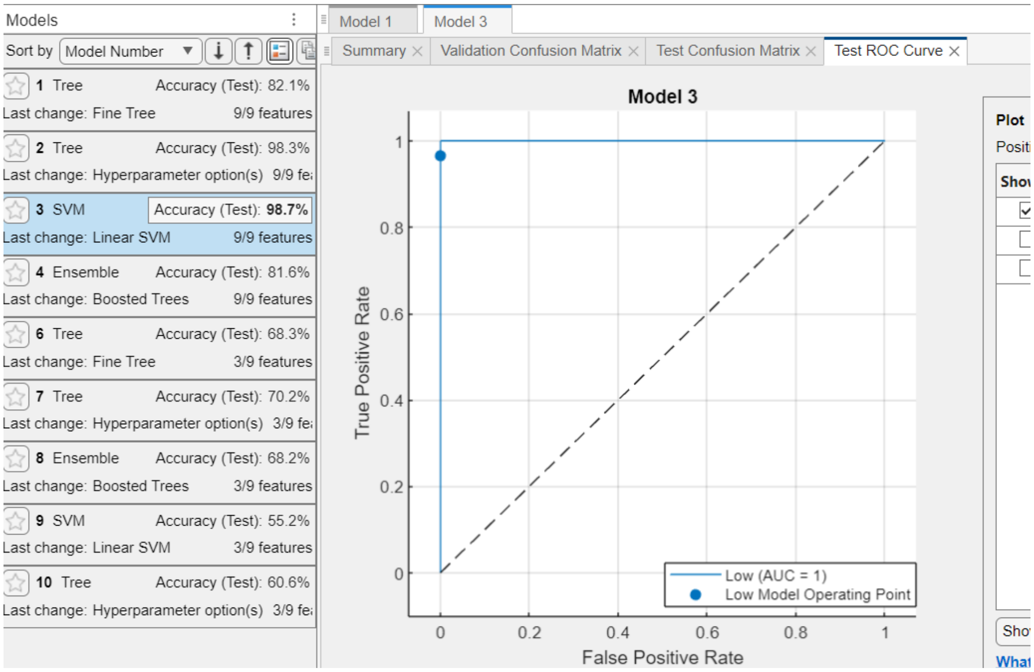
It can predict well to the Demand using those feature

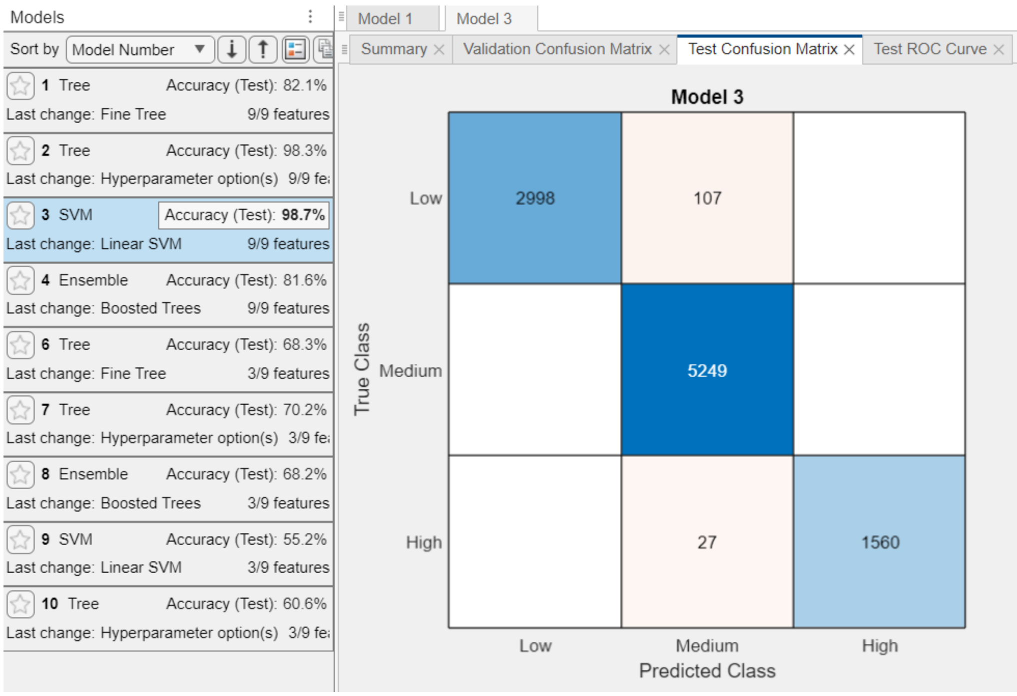




Test metrics

After doing test, we can see that model 3 give us the best model for predicting the Demand





Conclusions

Using some of groupsummary variable and variable generated from feature engineering process can give best model for predicting the Demand around Manhattan and the airports. The final model, which is linear SVM with 9 predictors give 99.5% accuracy for training and validation and 98.7% in testing. Using the model 2 with consideration of using more feature give better prediction than model 1.

This succesful model demand hopefully will enable Super Taxis to focus on high demand regions and avoid low demand ones, thus improving efficiency and increasing profits.

Summary of model

As shown above, the model that give the best prediction is the final model. The result summary of the Classification learner app are as follow

```
resultsTable1
```

resultsTable1 = 9x8 table

...

	Favorite	Model Number	Model Type	Status	Accuracy % (Validation)
1	0	"1"	"Tree"	"Tested"	82.5948
2	0	"2"	"Tree"	"Tested"	98.1114
3	0	"3"	"SVM"	"Tested"	99.5247
4	0	"4"	"Ensemble"	"Tested"	81.9938
5	0	"6"	"Tree"	"Tested"	68.6752
6	0	"7"	"Tree"	"Tested"	69.9678
7	0	"8"	"Ensemble"	"Tested"	68.3357

	Favorite	Model Number	Model Type	Status	Accuracy % (Validation)
8	0	"9"	"SVM"	"Tested"	56.0281
9	0	"10"	"Tree"	"Tested"	60.5799

Appendix

Appendix 1 : Data section

Task 1 : Creating the taxi regions

Add zone to taxi data

```
taxiAll_zones = addTaxiZones(taxiAll);
```

Add duration to taxi data

```
taxiAll_zones = addDuration(taxiAll_zones);
```

Create the region for Pickup and Dropoff

```
dataTable = taxiAll_zones;
A = dataTable.PickupZone;
TaxiRegion = readtable('/MATLAB Drive/Predictive Modeling and Machine Learning/Taxi Regions
and Zones.csv');
B = TaxiRegion.LowerManhattan;
B1 = TaxiRegion.Midtown;
B2 = TaxiRegion.UpperEastSide;
B3 = TaxiRegion.UpperWestSide;
B4 = TaxiRegion.JFKAirport;
B5 = TaxiRegion.LaGuardiaAirport;
B = categorical(B);
B1 = categorical(B1);
B2 = categorical(B2);
B3 = categorical(B3);
B4 = categorical(B4);
B5 = categorical(B5);
LowMan = ismember(A,B);
MidTown = ismember(A,B1);
UpEast = ismember(A,B2);
UpWest = ismember(A,B3);
JFKAirport = ismember(A,B4);
LaGuardiaAirport = ismember(A,B5);
dataTable.PickupRegion(LowMan) = "Lower Manhattan";
dataTable.PickupRegion(MidTown) = "Midtown";
dataTable.PickupRegion(UpEast) = "Upper East Side";
dataTable.PickupRegion(UpWest) = "Upper West Side";
dataTable.PickupRegion(JFKAirport) = "JFK Airport";
dataTable.PickupRegion(LaGuardiaAirport) = "LaGuardia Airport";
dataTable.PickupRegion = categorical(dataTable.PickupRegion);
A1 = dataTable.DropoffZone
B = TaxiRegion.LowerManhattan;
B1 = TaxiRegion.Midtown;
B2 = TaxiRegion.UpperEastSide;
B3 = TaxiRegion.UpperWestSide;
B4 = TaxiRegion.JFKAirport;
B5 = TaxiRegion.LaGuardiaAirport;
```

```
B = categorical(B);
B1 = categorical(B1);
B2 = categorical(B2);
B3 = categorical(B3);
B4 = categorical(B4);
B5 = categorical(B5);
LowMan = ismember(A1,B);
MidTown = ismember(A1,B1);
UpEast = ismember(A1,B2);
UpWest = ismember(A1,B3);
JFKAirport = ismember(A1,B4);
LaGuardiaAirport = ismember(A1,B5);
dataTable.DropoffRegion(LowMan) = "Lower Manhattan";
dataTable.DropoffRegion(MidTown) = "Midtown";
dataTable.DropoffRegion(UpEast) = "Upper East Side";
dataTable.DropoffRegion(UpWest) = "Upper West Side";
dataTable.DropoffRegion(JFKAirport) = "JFK Airport";
dataTable.DropoffRegion(LaGuardiaAirport) = "LaGuardia Airport";
dataTable.DropoffRegion = categorical(dataTable.DropoffRegion);
```

Task 2 : Data Cleaning

```
dataTable = basicPreprocessing(dataTable);
dataTable = dataTable(dataTable.Fare >= 2.5,:);
```

Task 3 : Data Restructuring - Creating Group Summary Table

Group Summary Table include this following:

1. A categorical variable that records the 6 taxi **regions**.

2. A datetime variable that records each **hour** of the year for each of the taxi regions.

These first two variables establish the groups for the taxi data, such that there is only one group for each combination of **region** and **hour**. The remaining features record summary statistics for each group. You must include variables for:

3. The number of taxi **pickups**.

4. The number of taxi **drop-offs**.

5. The number of **net pickups**, defined as **pickups** minus **drop-offs**.

6. The **mean or median distance** for all pickups within the group.

7. The **mean or median duration** for all pickups within the group.

8. The **mean or median fare** for all pickups within the group.

```
dataTable.HourlyBin = dateshift(dataTable.PickupTime,"start","hour");
count_pickup = groupsummary(dataTable,
["PickupRegion","HourlyBin"],"none","IncludeMissingGroups",false);
count_dropoff = groupsummary(dataTable,
["DropoffRegion","HourlyBin"],"none","IncludeMissingGroups",false);
% Join tables
joinedData = outerjoin(count_pickup,count_dropoff,"LeftKeys",["HourlyBin", ...
    "PickupRegion"],"RightKeys",["HourlyBin","DropoffRegion"],"MergeKeys",true);
newdata = fillmissing(joinedData.GroupCount_count_dropoff,"constant",0);
```

```
newdata1 = fillmissing(joinedData.GroupCount_count_pickup,"constant",0);
joinedData.GroupCount_count_dropoff = newdata;
joinedData.GroupCount_count_pickup = newdata1;
joinedData.Properties.VariableNames = ["Region","HourlyBin","PickupCount","DropoffCount"];
mean_pickup = groupsummary(dataTable,["PickupRegion","HourlyBin"],"mean",
["Distance","Duration","Fare"],"IncludeMissingGroups",false);
mean_pickup.Properties.VariableNames =
["Region","HourlyBin","Count","AvgDistance","AvgDuration","AvgFare"];
avg = fillmissing(mean_pickup.AvgDistance,"constant",0);
avg1 = fillmissing(mean_pickup.AvgDuration,"constant",0);
avg2 = fillmissing(mean_pickup.AvgFare,"constant",0);
mean_pickup.AvgDistance = avg;
mean_pickup.AvgDuration = avg1;
mean_pickup.AvgFare = avg2;
mean_pickup = mean_pickup(:,["Region","HourlyBin","AvgDistance","AvgDuration","AvgFare"]);

% Join tables
TaxiSummary = innerjoin(joinedData,mean_pickup,"LeftKeys",["Region", ...
    "HourlyBin"],"RightKeys",["Region","HourlyBin"]);
TaxiSummary.NetPickups = TaxiSummary.PickupCount - TaxiSummary.DropoffCount;

p_remove_cleaning = 100*(height(taxiAll_zones)-height(dataTable))/height(taxiAll_zones);
Taximid = TaxiSummary(TaxiSummary.Region == 'Midtown',:);
groupsummary(TaxiSummary,"Region","range","NetPickups");
groupsummary(TaxiSummary,"Region","mean","AvgFare");

TaxiSummary = TaxiSummary(~ismissing(TaxiSummary.Region),:);

taxiJFK = TaxiSummary(TaxiSummary.Region == "JFK Airport",:);
taximidtown = TaxiSummary(TaxiSummary.Region == "Midtown",:);
taxiLGA = TaxiSummary(TaxiSummary.Region == "LaGuardia Airport",:);
taxilowmanhattan = TaxiSummary(TaxiSummary.Region == "Lower Manhattan",:);
taxiUES = TaxiSummary(TaxiSummary.Region == "Upper East Side",:);
taxiUWS = TaxiSummary(TaxiSummary.Region == "Upper West Side",:);

corr_JFK = corr(taxiJFK.AvgDistance,taxiJFK.AvgDuration)
corr_midtown = corr(taximidtown.AvgDistance,taximidtown.AvgDuration)
corr_LGA = corr(taxiLGA.AvgDistance,taxiLGA.AvgDuration)
corr_lowmanhattan = corr(taxilowmanhattan.AvgDistance,taxilowmanhattan.AvgDuration)
corr_UES = corr(taxiUES.AvgDistance,taxiUES.AvgDuration)
corr_UWS = corr(taxiUWS.AvgDistance,taxiUWS.AvgDuration)
```

Appendix 2 : Model Training, validation, and testing

Code for Model training, validation, and testing

```
function [trainedClassifier, validationAccuracy] = trainClassifier(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.
%
%
```

```matlab
%  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 representing the validation accuracy as
%       a percentage. In the app, the Models pane displays the validation
%       accuracy 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,scores] = 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 29-Dec-2023 16:16:20


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Region', 'PickupCount', 'DropoffCount', 'AvgDistance', 'AvgDuration',
'AvgFare', 'DayofYear', 'IsHoliday', 'HourOfDay'};
predictors = inputTable(:, predictorNames);
response = inputTable.Demand;
isCategoricalPredictor = [true, false, false, false, false, false, false, true, false];
classNames = categorical({'Low'; 'Medium'; 'High'}, {'Low' 'Medium' 'High'}, 'Ordinal', true);

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
template = templateSVM(...
    'KernelFunction', 'linear', ...
    'PolynomialOrder', [], ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', classNames);
```

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

% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'Region', 'PickupCount', 'DropoffCount', 'AvgDistance',
'AvgDuration', 'AvgFare', 'DayofYear', 'IsHoliday', 'HourOfDay'};
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained model exported from Classification Learner
R2023b.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use:
\n  [yfit,scores] = c.predictFcn(T) \nreplacing ''c'' with the name of the variable that
is this struct, e.g. ''trainedModel''. \n \nThe table, T, must contain the variables
returned by: \n  c.RequiredVariables \nVariable formats (e.g. matrix/vector, datatype)
must match the original training data. \nAdditional variables are ignored. \n \nFor
more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
''appclassification_exportmodeltoworkspace'')">How to predict using an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Region', 'PickupCount', 'DropoffCount', 'AvgDistance', 'AvgDuration',
'AvgFare', 'DayofYear', 'IsHoliday', 'HourOfDay'};
predictors = inputTable(:, predictorNames);
response = inputTable.Demand;
isCategoricalPredictor = [true, false, false, false, false, false, false, true, false];
classNames = categorical({'Low'; 'Medium'; 'High'}, {'Low' 'Medium' 'High'}, 'Ordinal', true);

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

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

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