In-Vehicle Coupon Recommendation

HarvardX - PH125.9x: Data Science: Capstone

Ringgold P. Atienza

August 02, 2022

1. Introduction

Every data-driven company's most dominant machine learning task is predicting customer behavior. Machine learning predicts customer behavior using data mining and statistical techniques to uncover insights from the historical data. An accurate predictive model of customer behavior can lead to an effective marketing strategy, which results in brand growth and an increase in revenue [1]. Thus, companies are always looking to improve their predictions to stay ahead of the competition.

For this project, I created a machine learning model to predict whether a person will accept a coupon or not using three classification algorithms. The three algorithms' predictive values were compared. The data for this project was downloaded from the UCI machine learning repository website [2]. The survey data was initially collected from Amazon Mechanical Turk. The survey describes different driving scenarios and then asks whether the person will accept the coupon given the person is the driver. The data includes information on destination, passenger, weather, temperature, time, type of coupon, coupon expiration, gender, age, marital status, education, occupation, income, frequency of going to bars and coffee houses, frequency of having takeaways, frequency of going to restaurants, and driving distance to restaurant or bar for using a coupon. The data has 12,685 observations and was made publicly available on September 15, 2020. This project aims to create a classification algorithm with an $F_1 > 0.75$ in the validation set.

Since the project is a classification problem, I utilized three of the popular classification algorithm, namely the classification and regression tree (CART) model, random forest, and XGBoost. CART is a tree-based algorithm that looks at ways to locally partition or split data into smaller segments based on differing values and combinations of predictors. CART selects the best performing splits, then repeats this process recursively until the optimal collection is found [3]. The result of CART is visualized as a decision tree with a series of binary splits that ultimately leads to terminal nodes (i.e., to accept the coupon or not). Random forest is an ensemble learning method for classification based on a collection of CART Trees, bringing together independent trees to determine the overall prediction of the forest [4]. The Extreme Gradient Boosting or XGBoost is a powerful tool for classification that implements gradient boosted decision trees designed for speed and performance [5].

2. Method and Analysis

2.1. Preparing the Data

2.1.1. Installation of Packages

```
library(tidyverse)
library(dplyr)
library(caTools)
library(randomForest)
library(xgboost)
library(MLmetrics)
```

2.1.2. Train, Test, and Validation Sets

In machine learning, the best practice is to split the data into three independent sets: a training set, a test set, and a validation set [6]. The training dataset is used to fit the model. The test and validation sets are a sample of data taken from the training dataset to estimate the model's performance while tuning the model's hyperparameter. A test dataset is different from the validation dataset as it provides an unbiased evaluation of the model fit on the training dataset. This process is important to cross-validate and refines the final model without the risk of over-fitting. Meanwhile, the validation dataset provides an unbiased evaluation of the performance of the final tuned model. The validation set is not used for training, developing, or selecting algorithms but only to evaluate the final model.The process of the prediction modelling is presenting below (Figure 1)

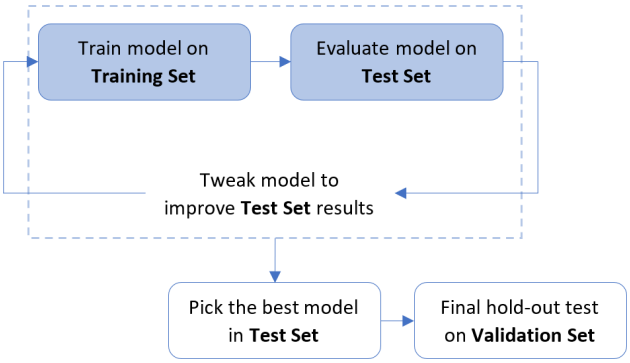


Figure 1: Modelling Process

The In-Vehicle Coupon Recommendation Dataset was downloaded from the website: https://archive.ics.uci.edu/ml/datasets/in-vehicle+coupon+recommendation. The partitioning was then applied using the 70-30-30 split method (see Figure 2), as it is considered to be one of the most common partition practices in machine learning [6].

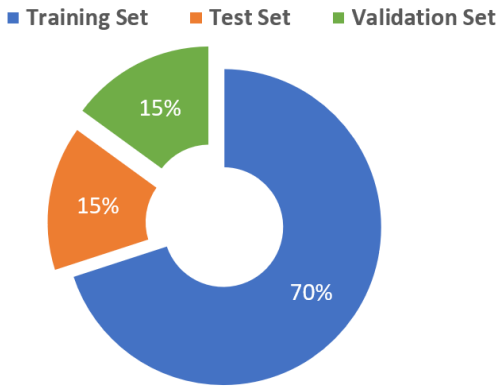


Figure 2: Data Partition

2.2. Data Inspection

2.2.1. Check for NAs and Singular Level Features

I indentified and understand the attibutes of the dataset. Below we see the following column names belows. The 'direction_opp' variable is just the opposite of 'direction_same'. This will result to a very high correlation between both variables. The 'direction_opp' was excluded in the dataset.

```
colnames(trainingset)
```

```
##  [1] "destination"          "passanger"            "weather"
##  [4] "temperature"          "time"                 "coupon"
##  [7] "expiration"           "gender"               "age"
## [10] "maritalStatus"        "has_children"         "education"
## [13] "occupation"           "income"               "car"
## [16] "Bar"                  "CoffeeHouse"          "CarryAway"
## [19] "RestaurantLessThan20" "Restaurant20To50"     "toCoupon_GEQ5min"
## [22] "toCoupon_GEQ15min"    "toCoupon_GEQ25min"    "direction_same"
## [25] "Y"
```

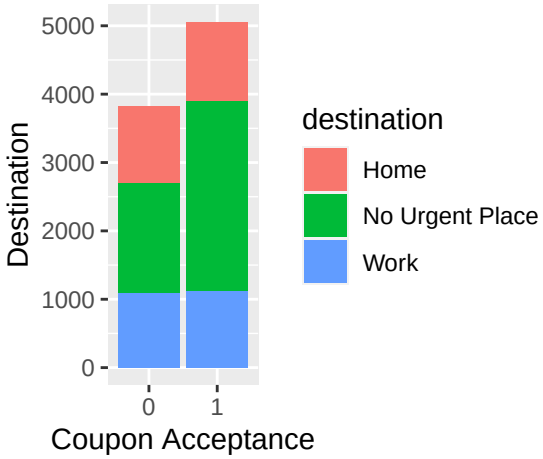
I checked for the the missing values and the results showed that no NA were found as shown below:

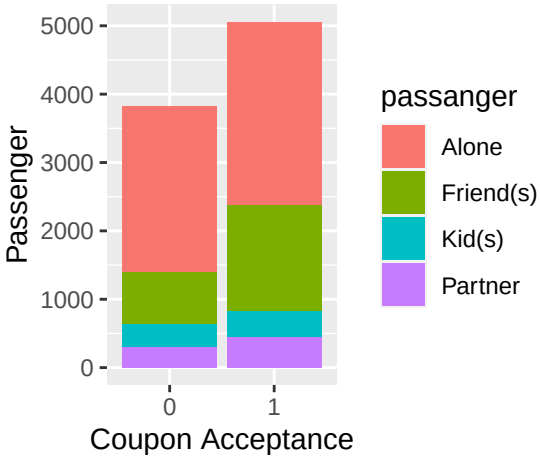
```
na_count <- sapply(trainingset, function(y) sum(length(which(is.na(y)))))
na_count
```

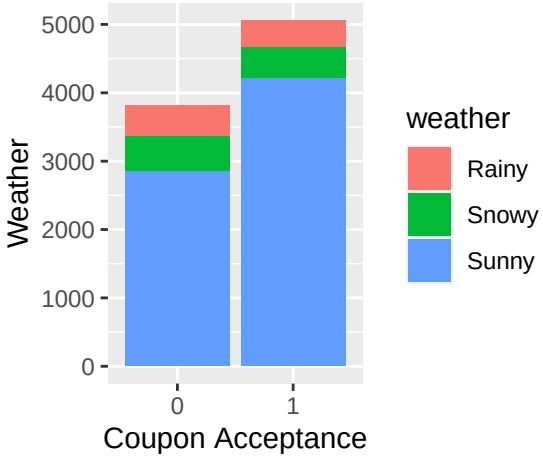
```
##          destination            passanger              weather
##                    0                    0                    0
##          temperature                 time               coupon
##                    0                    0                    0
##           expiration               gender                  age
##                    0                    0                    0
##        maritalStatus         has_children            education
##                    0                    0                    0
##           occupation               income                  car
##                    0                    0                    0
##                  Bar          CoffeeHouse            CarryAway
##                    0                    0                    0
## RestaurantLessThan20     Restaurant20To50     toCoupon_GEQ5min
##                    0                    0                    0
##    toCoupon_GEQ15min    toCoupon_GEQ25min       direction_same
##                    0                    0                    0
##                    Y
##                    0
```

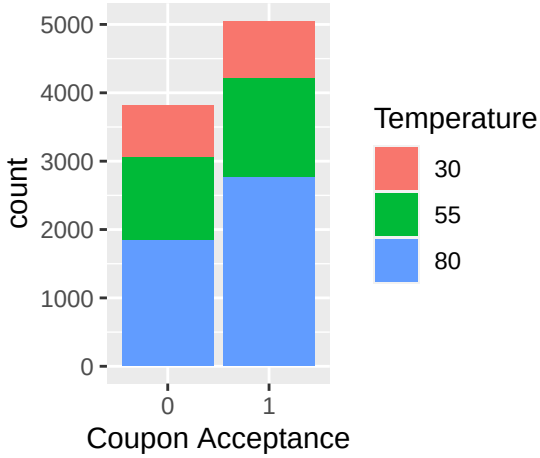
2.2.2. Data Visualization

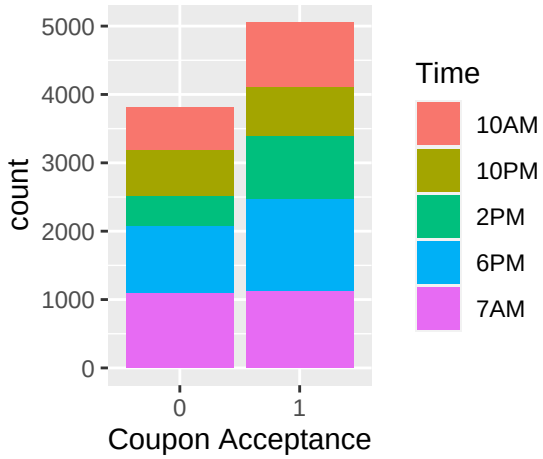
I further check the data through visualization. Results below showed that the features 'toCoupon_GEQ5min', 'toCoupon_GEQ15min', 'car' showed no variation (i.e., only has one value), and were included in the data set. Moreover, it also showed that there were features with no values that were not detected using 'is.na'.

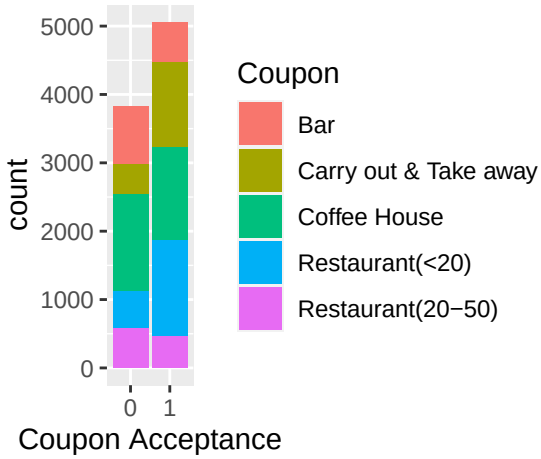

Destination
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
destination
Home
No Urgent Place
Work

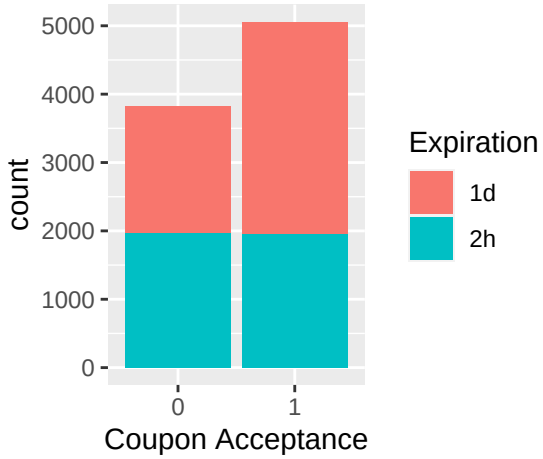

Passenger
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
passanger
Alone
Friend(s)
Kid(s)
Partner

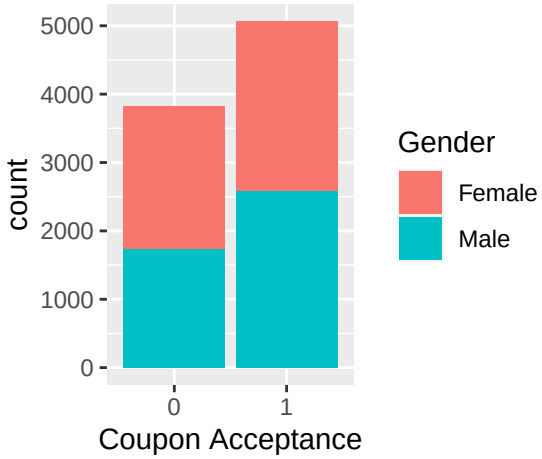

Weather
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
weather
Rainy
Snowy
Sunny

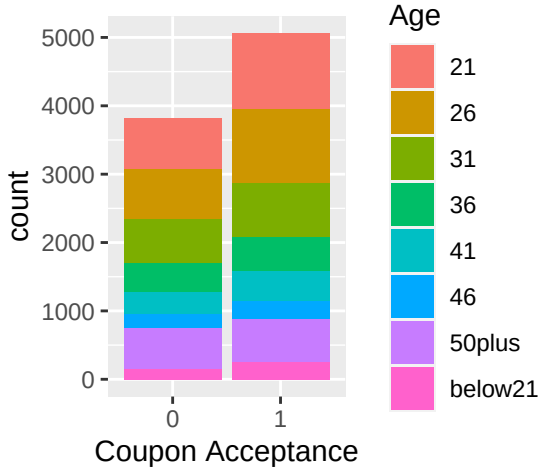

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Temperature
30
55
80

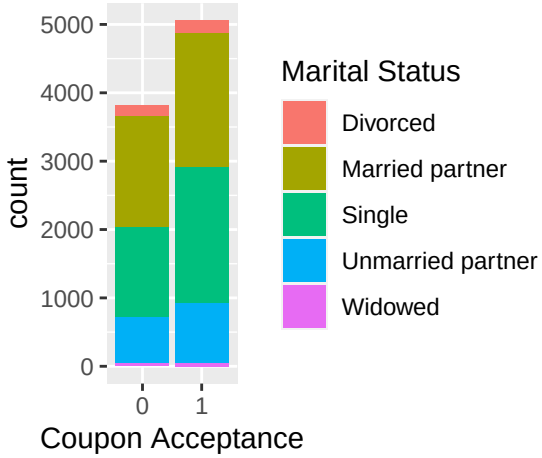

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Time
10AM
10PM
2PM
6PM
7AM

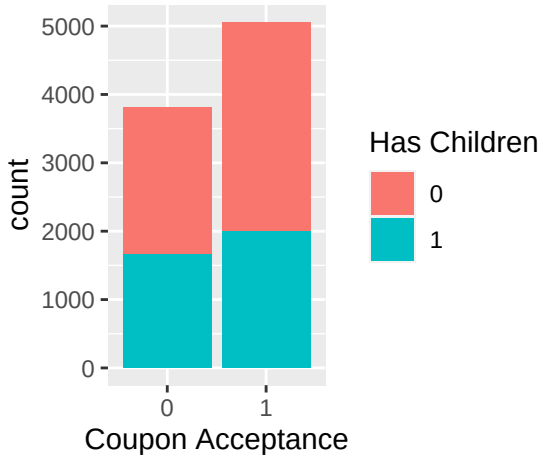

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Coupon
Bar
Carry out & Take away
Coffee House
Restaurant(<20)
Restaurant(20−50)

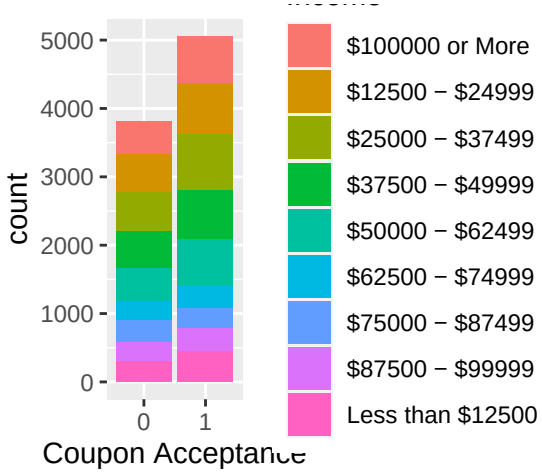

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Expiration
1d
2h

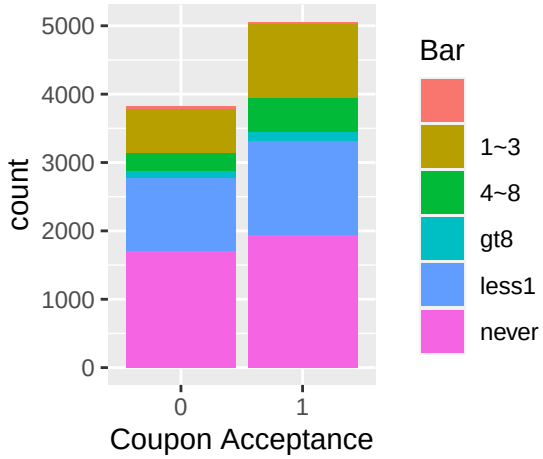

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Gender
Female
Male

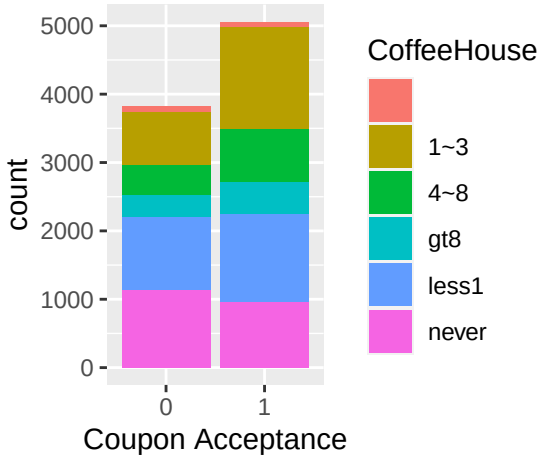

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Age
21
26
31
36
41
46
50plus
below21

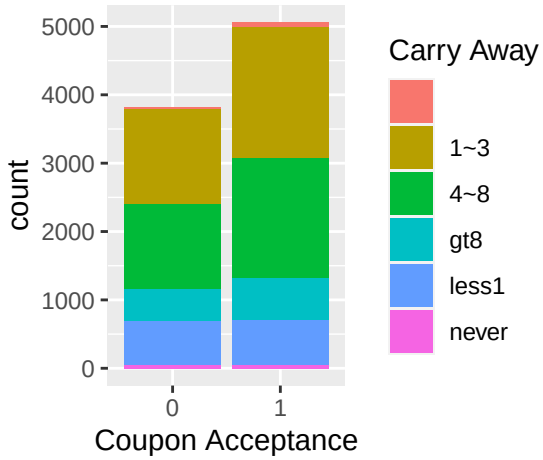

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Marital Status
Divorced
Married partner
Single
Unmarried partner
Widowed

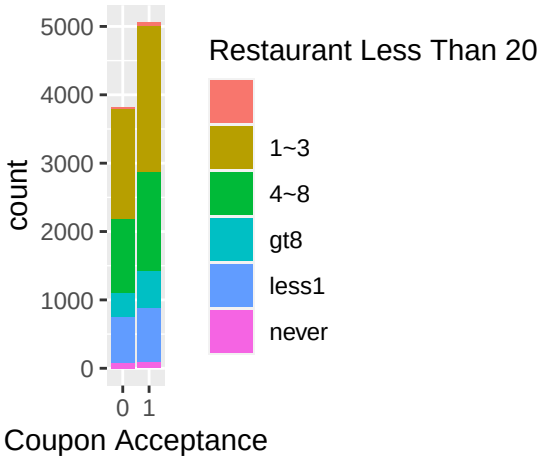

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
Has Children
0
1

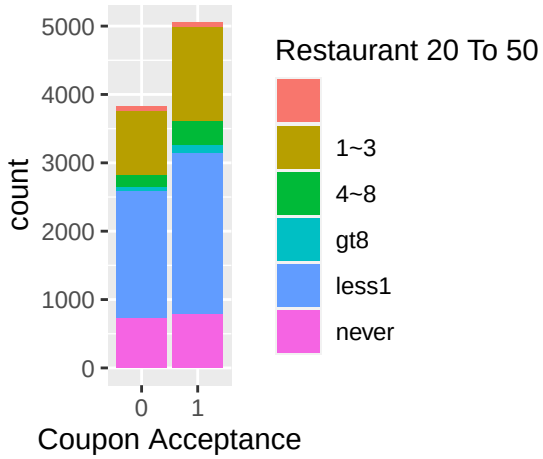

count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
$100000 or More
$12500 − $24999
$25000 − $37499
$37500 − $49999
$50000 − $62499
$62500 − $74999
$75000 − $87499
$87500 − $99999
Less than $12500

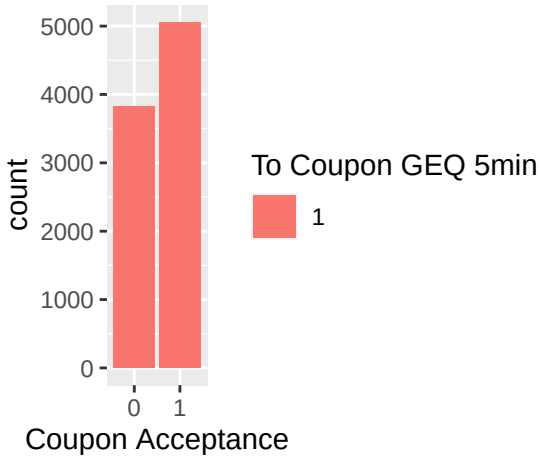

Bar
count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
1~3
4~8
gt8
less1
never

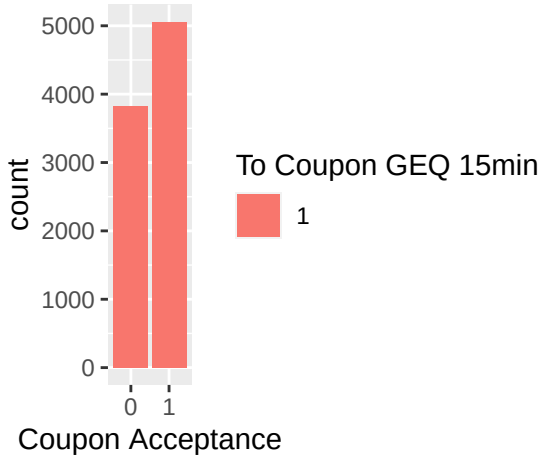

CoffeeHouse
count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
1~3
4~8
gt8
less1
never

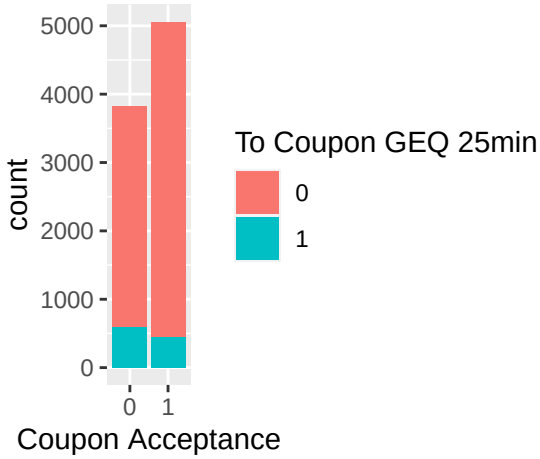

Carry Away
count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
1~3
4~8
gt8
less1
never

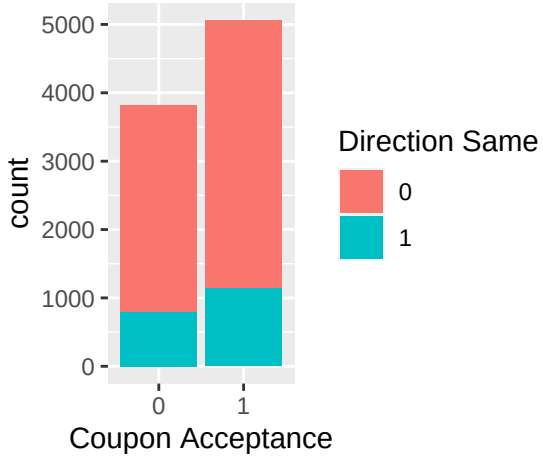

Restaurant Less Than 20
count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
1~3
4~8
gt8
less1
never

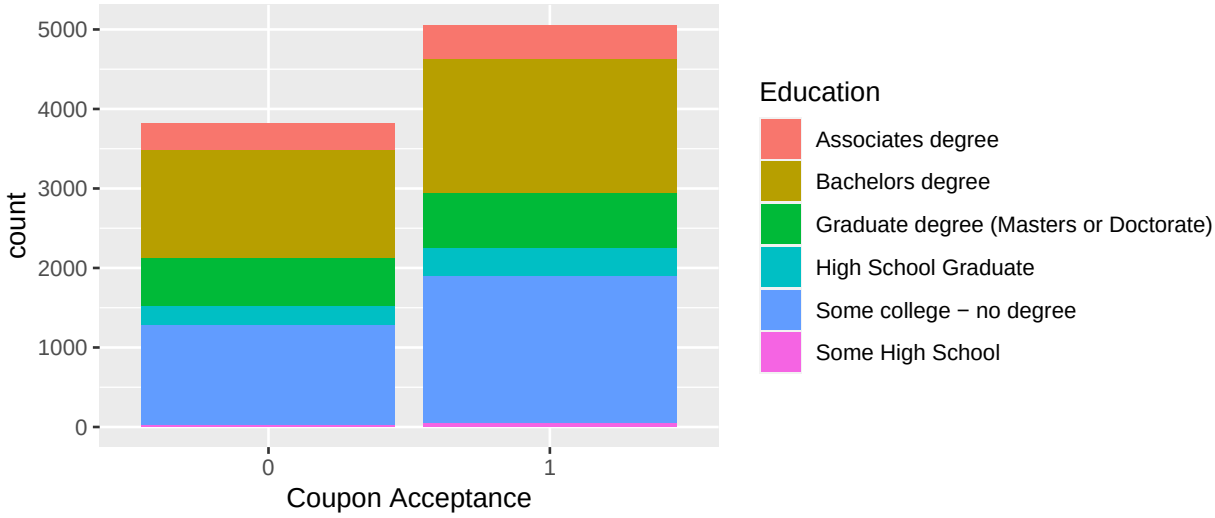

Restaurant 20 To 50
count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
1~3
4~8
gt8
less1
never

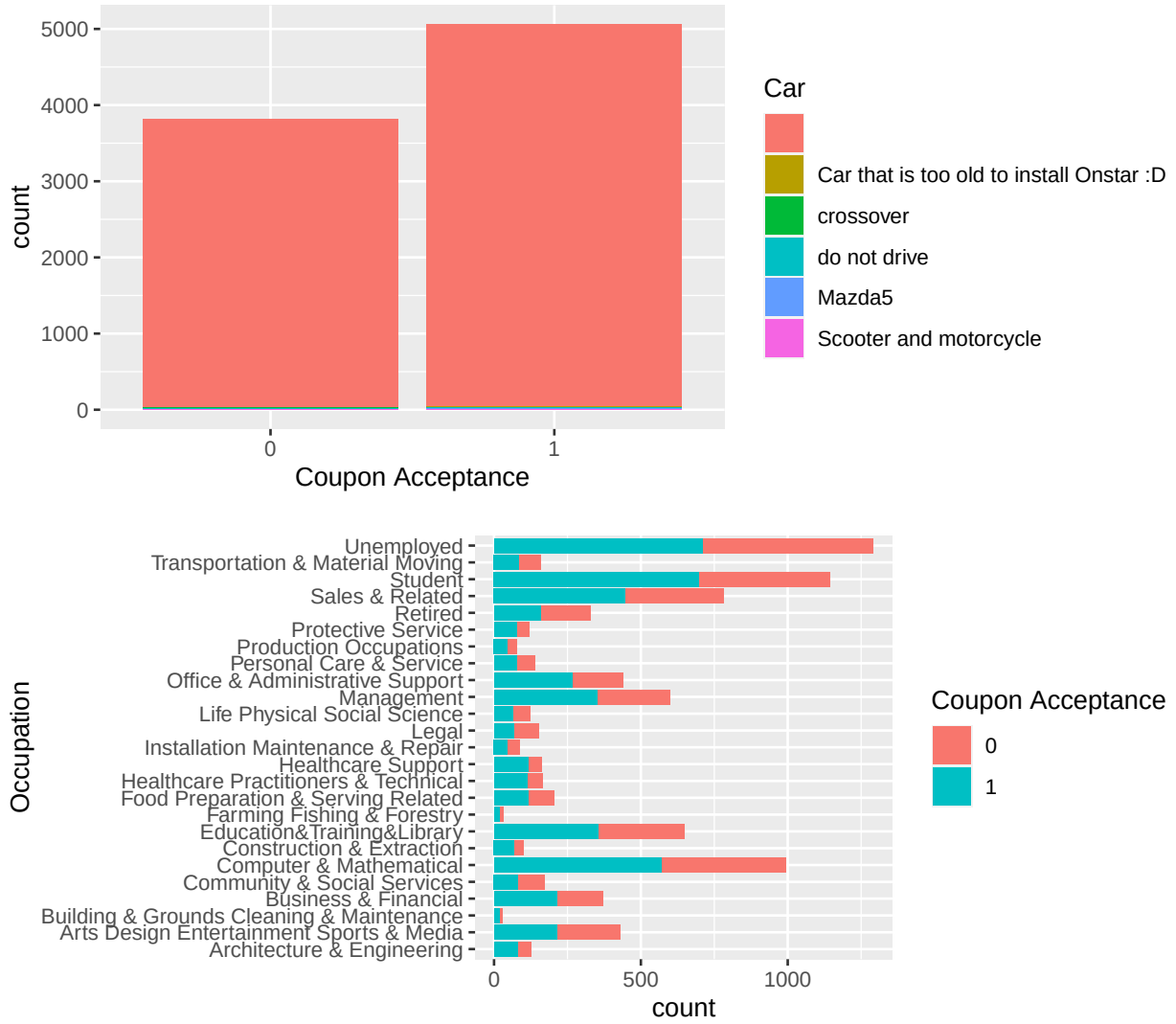

To Coupon GEQ 5min
count
5000
4000
3000
2000
1000
0
0
1
Coupon Acceptance
1


5000
4000
3000
2000
1000
0
count
0
1
Coupon Acceptance
To Coupon GEQ 15min
1


5000
4000
3000
2000
1000
0
count
0
1
Coupon Acceptance
To Coupon GEQ 25min
0
1


5000
4000
3000
2000
1000
0
count
0
1
Coupon Acceptance
Direction Same
0
1


5000
4000
3000
2000
1000
0
count
0
1
Coupon Acceptance
Education
Associates degree
Bachelors degree
Graduate degree (Masters or Doctorate)
High School Graduate
Some college − no degree
Some High School



We can see the following features that have no values on the 1st column of the table. The features with empty values comprised only of less than 2% which means imputation from the algorithm is feasible. Also, this is not the problem in this case, as the classification algorithms CART, Random Forest, and XGBoost are robust to missing values. However, this must be reported as one of the limitations of the dataset.

```
table(trainingset$RestaurantLessThan20)
```

```
##
##         1~3   4~8   gt8 less1 never
##    95  3734  2531   893  1464   161
```

```
table(trainingset$Restaurant20To50)
```

```
##
##         1~3   4~8   gt8 less1 never
##   139  2307   529   181  4225  1497
```

```
table(trainingset$CarryAway)
```

```
##
##         1~3   4~8   gt8 less1 never
##   116  3295  2995  1091  1282    99
```

```
table(trainingset$CoffeeHouse)
```

```
##
##         1~3   4~8   gt8 less1 never
##   158  2256  1239   774  2366  2085
```

```
table(trainingset$Bar)
```

```
##
##         1~3   4~8   gt8 less1 never
##    81  1734   741   253  2423  3646
```

2.2. F1 Score

It is often useful to have a one-number summary when we want to evaluate our prediction model, especially for optimization [7]. For that purpose, the F_1 score, is a widely used one-number summary for classification algorithms. The F_1 score is a measure of the model's accuracy, and is the harmonic average of the precision and recall:

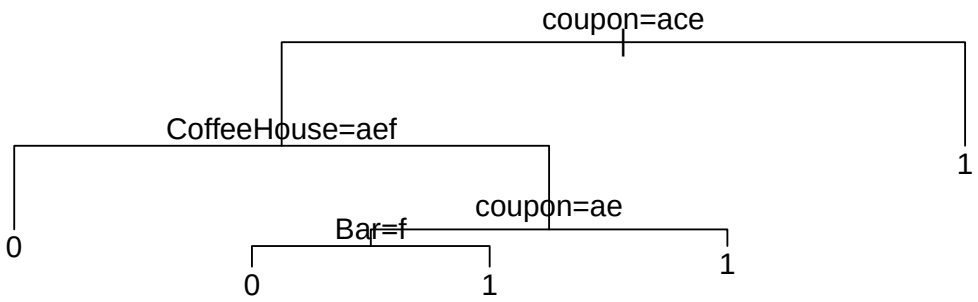
$$\frac{1}{\frac{1}{2}\left(\frac{1}{\text{recall}} + \frac{1}{\text{precision}}\right)}$$

In general, machine learning prediction models aims to maximize F_1 score as possible. The highest possible value of F_1 score is 1.0 indicating perfect precision and recall, while the lowest is 0.0, indicating zero precition and recall. This project's aim is to achieve $F_1 > 0.75$.

2.3. Classification and Regression Tree (CART)

2.3.1. Vanilla CART

The first algorithm I used in this project is the CART. It is also the simplest algorithm I used compared to the other two algorithms I used in this project. Before we set up the CART, we must mutate all features as a factor. Then I ran the model using 'rpart' function to predict coupon acceptance against all valid features in the training set. The model was then predicted on the test set. The results below showed that the vanilla CART was not able to achieve an $F_1 > 0.75$.

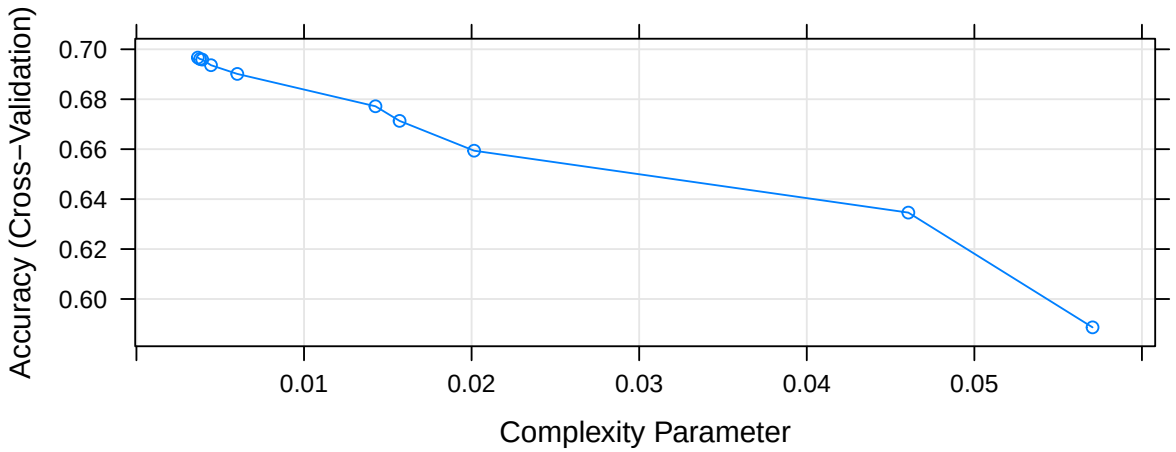



```
confusionMatrix(predicted.cases, testset$Y, mode = "everything", positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 408 252
##          1 401 842
##
##                Accuracy : 0.6569
##                  95% CI : (0.635, 0.6782)
##     No Information Rate : 0.5749
##     P-Value [Acc > NIR] : 1.573e-13
##
##                   Kappa : 0.2807
##
##  Mcnemar's Test P-Value : 6.968e-09
##
##             Sensitivity : 0.7697
##             Specificity : 0.5043
##          Pos Pred Value : 0.6774
##          Neg Pred Value : 0.6182
##               Precision : 0.6774
##                  Recall : 0.7697
##                      F1 : 0.7206
##              Prevalence : 0.5749
##          Detection Rate : 0.4425
##    Detection Prevalence : 0.6532
##       Balanced Accuracy : 0.6370
##
##        'Positive' Class : 1
##
```

2.3.2. Pruning the Tree

We can set the tuning parameter of the CART by pruning the tree and conducting the 10 times cross-validation. Below we can see the complexity parameter that produces the highest accuracy (Complexity parameter = 0.0036)



The new confusion matrix is shown below. In this case, the F_1 score improved. However, the objective of having $F_1 > 0.75$, was still not achieved.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 444 221
##          1 365 873
##
##                Accuracy : 0.6921
##                  95% CI : (0.6708, 0.7128)
##     No Information Rate : 0.5749
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.355
##
##  Mcnemar's Test P-Value : 3.478e-09
##
##             Sensitivity : 0.7980
##             Specificity : 0.5488
##          Pos Pred Value : 0.7052
##          Neg Pred Value : 0.6677
##               Precision : 0.7052
##                  Recall : 0.7980
##                      F1 : 0.7487
##              Prevalence : 0.5749
##          Detection Rate : 0.4587
##    Detection Prevalence : 0.6506
##       Balanced Accuracy : 0.6734
##
```

```
##        'Positive' Class : 1
##
```

2.3.3. Final Hold-Out Test for CART

Results in the final-hold out test using the Validation Set showed that CART almost achieved the objective of this project with a difference of 0.006 or 0.6%.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 480 216
##          1 364 843
##
##                Accuracy : 0.6952
##                  95% CI : (0.674, 0.7158)
##     No Information Rate : 0.5565
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3714
##
##  Mcnemar's Test P-Value : 1.035e-09
##
##             Sensitivity : 0.7960
##             Specificity : 0.5687
##          Pos Pred Value : 0.6984
##          Neg Pred Value : 0.6897
##               Precision : 0.6984
##                  Recall : 0.7960
##                      F1 : 0.7440
##              Prevalence : 0.5565
##          Detection Rate : 0.4430
##    Detection Prevalence : 0.6343
##       Balanced Accuracy : 0.6824
##
##        'Positive' Class : 1
##
```

2.4. Random Forest

2.4.1. Vanilla Random Forest

This time I utilized the random forest algorithm. I ran the algorithm using the 'randomForest' function to predict coupon acceptance against all valid features in the training set. Next, the model was used to predict the test set. The result of the prediction showed that the random forest surpassed the required F_1 score.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 516 208
```

```
##          1 293 886
##
##                Accuracy : 0.7367
##                  95% CI : (0.7163, 0.7564)
##     No Information Rate : 0.5749
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4539
##
##  Mcnemar's Test P-Value : 0.0001748
##
##             Sensitivity : 0.8099
##             Specificity : 0.6378
##          Pos Pred Value : 0.7515
##          Neg Pred Value : 0.7127
##               Precision : 0.7515
##                  Recall : 0.8099
##                      F1 : 0.7796
##              Prevalence : 0.5749
##          Detection Rate : 0.4656
##    Detection Prevalence : 0.6195
##       Balanced Accuracy : 0.7238
##
##        'Positive' Class : 1
##
```

2.4.2. Tuning the Hyperparameters

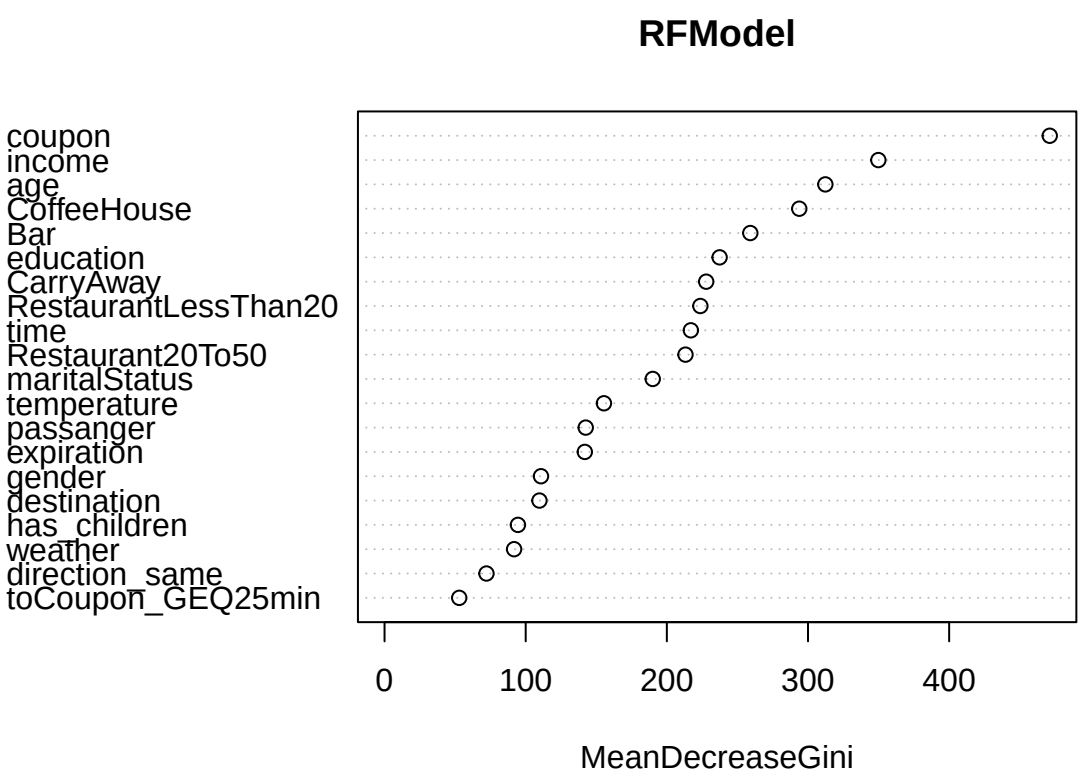
To tune the parameter we need to find the value of mtry the provides the least out of bag error (OOB). The out-of-bag (OOB) error is the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample [8]. We can set the initial mtry using this function:

```
floor(sqrt(ncol(trainingset) - 1))
```

```
## [1] 4
```

The'tuneRF' function was used to find the best mtry value. The results showed that an mtry value of 4 was the best tuning parameter.

The important features in the prediction model is shown in the figure below. The figure shows how much a decrease in Gini value would have a model if a feature is excluded. It shows that coupon, income, and age are the top three important features in the Random Forest Model.



I ran the again the 'randomForest' function with the new tuning parameter. The result indicates that the new tuning parameter increased the F_1 score on test set.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 515 206
##          1 294 888
##
##                Accuracy : 0.7373
##                  95% CI : (0.7169, 0.7569)
##     No Information Rate : 0.5749
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4547
##
##  Mcnemar's Test P-Value : 9.993e-05
##
##             Sensitivity : 0.8117
##             Specificity : 0.6366
##          Pos Pred Value : 0.7513
##          Neg Pred Value : 0.7143
##               Precision : 0.7513
##                  Recall : 0.8117
##                      F1 : 0.7803
```

```
##              Prevalence : 0.5749
##          Detection Rate : 0.4666
##    Detection Prevalence : 0.6211
##       Balanced Accuracy : 0.7241
##
##        'Positive' Class : 1
##
```

2.4.3 Final Hold-Out Test for Random Forest

Results in the final-hold out test using the Validation Set showed that Random Forest achieved the objective of this project, with an $F_1 = .7804$

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 526 168
##          1 318 891
##
##                Accuracy : 0.7446
##                  95% CI : (0.7244, 0.7641)
##     No Information Rate : 0.5565
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4731
##
##  Mcnemar's Test P-Value : 1.392e-11
##
##             Sensitivity : 0.8414
##             Specificity : 0.6232
##          Pos Pred Value : 0.7370
##          Neg Pred Value : 0.7579
##               Precision : 0.7370
##                  Recall : 0.8414
##                      F1 : 0.7857
##              Prevalence : 0.5565
##          Detection Rate : 0.4682
##    Detection Prevalence : 0.6353
##       Balanced Accuracy : 0.7323
##
##        'Positive' Class : 1
##
```

2.5. XGBoost

2.5.1. Vanilla XGBoost

Lastly, I utilized the XGBoost algorithm which is considered as the more powerful algorithm compared to CART and Random Forest. Before running the XGBoost algorithm, several mutations were done in the dataset such as the creating dummy variables for all features using 'sparse.model.matrix' function and setting parameters. After completing all conditions in the dataset, I ran the XGboost algorithm using the

'xgb.train' function. Next, the model was used to predict the test set. The result of the prediction showed that the XGBoost surpassed the required F_1 score.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 543 266
##          1 201 893
##
##                Accuracy : 0.7546
##                  95% CI : (0.7346, 0.7738)
##     No Information Rate : 0.609
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4926
##
##  Mcnemar's Test P-Value : 0.003061
##
##             Sensitivity : 0.7705
##             Specificity : 0.7298
##          Pos Pred Value : 0.8163
##          Neg Pred Value : 0.6712
##               Precision : 0.8163
##                  Recall : 0.7705
##                      F1 : 0.7927
##              Prevalence : 0.6090
##          Detection Rate : 0.4693
##    Detection Prevalence : 0.5749
##       Balanced Accuracy : 0.7502
##
##        'Positive' Class : 1
##
```

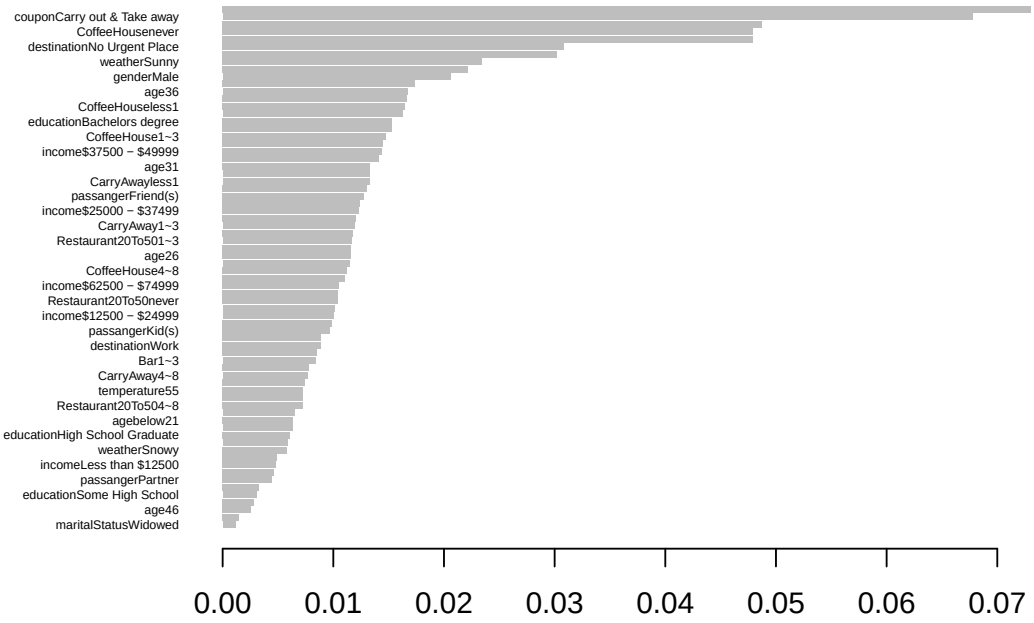
2.5.2 Final Hold-Out Test for XGBoost

The XGBoost algorithm has exceeded both CART and Random Forest hold-out test on Validation Set. XGBoost showed a final $F_1 = .7927$

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 545 299
##          1 165 894
##
##                Accuracy : 0.7562
##                  95% CI : (0.7362, 0.7753)
##     No Information Rate : 0.6269
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.498
##
```

```
##  Mcnemar's Test P-Value : 6.643e-10
##
##             Sensitivity : 0.7494
##             Specificity : 0.7676
##          Pos Pred Value : 0.8442
##          Neg Pred Value : 0.6457
##               Precision : 0.8442
##                  Recall : 0.7494
##                      F1 : 0.7940
##              Prevalence : 0.6269
##          Detection Rate : 0.4698
##    Detection Prevalence : 0.5565
##       Balanced Accuracy : 0.7585
##
##        'Positive' Class : 1
##
```

The important features in the predction model is shown below. The coupon, coffee house, destination, weather and gender were the top five important features in the XGBoost model.



3. Results

Results from the final-hold out tests indicate that XGboost provided the highest F_1 score, followed by Random Forest and CART. XGBoost has the highest number of correct predictions for both positive and

negative responses. However, Random Forest has the best recall among the algorithms. The random forest has less false-negative prevalence (customers want the coupon but are not given). We prefer better recall in this case because we would maximize our ability to identify potential customers rather than non-potential ones. Also, if in issue, the cost of losing a potential customer is higher than the cost of coupons.

Meanwhile, XGBoost has the best precision among the algorithms, which means it has less false-positive prevalence (customers do not want coupons but are given). Better precision means better cost efficiency (less waste on resources). When we have limited resources such as coupons, better precision is preferred.

Variable	F1	Precision	Recall
CART	0.7440	0.6984	0.7960
Random Forest	0.7857	0.7370	0.8414
XGBoost	0.7940	0.8442	0.7494

4. Conclusion

This report aimed to predict in-vehicle coupon acceptance using three classification algorithms namely, CART, Random Forest, and XGBoost. The results showed that XGBoost provides the highest F_1 score. Meanwhile, since the F_1 score is the harmonic mean of precision and recall, it is assumed that both precision and recall are equally valued. However, in reality, some companies prefer precision (cost efficiency) than recall (wider reach) or vice-versa.

While XGBoost is considered the best algorithm in the project, its downside is its interpretability. Due to its size, I cannot show the multi-trees in the R-Markdown that tell us the train-of-decision in its prediction. XGBoost's interpretability might do well using neatly prepared features.

Another limitation comes from the attributes of the dataset. I find that some features can still be tweaked for improved performance. For example, occupations can be categorized into different types, such as blue collar jobs, white collar jobs, etc. We also found empty values in some features. Fortunately, the algorithms used in this project are robust to missing values. Future works should consider these limitations.

Overall, the final result of this project is successful. All algorithms have showed promise in their prediction. Consumer prediction is considered to be one of the important task in machine learning. With this project, company's will be able to predict in-vehicle coupon acceptance at more than 75% accuracy (F_1) score.

References

[1] https://www.salecycle.com/blog/strategies/

[2] https://archive.ics.uci.edu/ml/datasets/in-vehicle+coupon+recommendation

[3] https://www.minitab.com/en-us/predictive-analytics/cart/

[4] https://www.ibm.com/cloud/learn/random-forest

[5] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[6] https://www.v7labs.com/blog/train-validation-test-set

[7] https://rafalab.github.io/dsbook/introduction-to-machine-learning.html#evaluation-metrics

[8] http://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html